Université Clermont Auvergne

École doctorale des
Sciences Pour l'Ingénieur de Clermont-Ferrand

Thèse

présentée par

David Brevet

pour obtenir le grade de

Docteur d'Université

Spécialité : Informatique

Conception de services pour la mobilité urbaine respectueux des données privées et assurant la sécurité des clients et des véhicules

Soutenue publiquement le 02/12/2019 devant le jury composé de :

Président :	
Xavier Delorme	Professeur à l'École des Mines de Saint-Etienne
Rapporteurs :	
Marc Sevaux	Professeur à l'Université de Bretagne Sud
Marie-Ange Manier	Maître de Conférences, HDR, Université de Technologie de Belfort-Montbéliard
Examinateurs :	
Nikolay Tchernev	Professeur à l'Université Clermont-Auvergne (UCA)
Wahiba Ramdane-Cherif	Maître de Conférences, École des Mines de Nancy
Directeurs de thèse :	
Philippe Lacomme	Maître de Conférences, HDR, Université Clermont Auvergne
Christophe Duhamel	Maître de Conférences, Université du Havre
Gérard Fleury	Maître de Conférences, HDR, Université Clermont Auvergne

IMOBS3
LABORATOIRE D'EXCELLENCE MOBILITÉ INNOVANTE

INVESTISSEMENTS D'AVENIR

Ce travail de recherche est financé par le LabEx IMobS3 de l'université Clermont Auvergne dans le cadre du projet Investissement d'Avenir.

Remerciements

Je souhaite tout d'abord remercier mes directeurs de thèse, Philippe Lacomme, Christophe Duhamel et Gérard Fleury pour leur présence et leur aide tout au long de ce travail de recherche. Avoir des directeurs de recherche présents quotidiennement est un luxe dont peu de doctorants peuvent bénéficier. Ils ont réussi à me guider et à me supporter (dans les deux sens du terme) au cours de ces trois années.

Je remercie également Marc Sevaux, Marie-Ange Manier, Xavier Delorme, Nicolay Tchernev et Wahiba Ramdane-Cherif pour avoir fait partie de mon jury de thèse, mais également pour leur lecture de ce manuscrit et ainsi contribuer à son amélioration.

Je remercie tout le personnel du LIMOS pour leur aide et leur réactivité : Nicolas et ses ordinateurs, Béatrice et ses bonbons,…

Merci à tous les doctorants (et maintenant docteurs pour la plupart !) pour leur accueil du début à la fin de cette thèse : arriver seul dans une ville inconnue en laissant ses proches au loin est difficile, mais grâce à vous, j'en garde des souvenirs mémorables (ah la Tête dans le Pion, le Beerland ou Delirium…).

Merci à Alexis pour ses rentrées de 9 à la casse, Kévin pour nos ballades sandwich et BergougnouX pour… lui-même. Co-bureaux de mon cœur (mais pas mormons !) et meilleurs chercheurs du B108 : lorsque quatre esprits issus de quatre sujets de thèse radicalement différents réfléchissent ensemble à un problème donné, la tournure des évènements est souvent mythique. Si vous n'aviez pas été présent, ces trois années auraient été beaucoup plus difficiles.

Merci à l'ensemble des enseignants qui m'ont aidé, de l'école primaire jusqu'à la fin du Master pour leur contribution à mon développement intellectuel et culturel. C'est grâce à des personnes comme vous que des élèves finissent par croire en leurs capacités et tentent l'impossible.

Merci à ma belle-famille, pour leur accueil (muscadet), leurs chants inimitables (ACDC, Queen, muscadet), et surtout pour leur muscadet (je l'ai déjà dit non ?).

Comment ne pas remercier également mes parents pour l'éducation qu'ils m'ont fournie. Grâce à la liberté que vous m'avez offerte, je peux aujourd'hui travailler dans un domaine exceptionnel. Merci également à mon petit frère Matisse, qui n'est maintenant plus si petit que ça.

Les derniers remerciements iront à la personne ayant accepté de partager sa vie avec la mienne depuis plus de sept ans maintenant. Agathe, je pense que nous rigolerons bien lorsque nous relirons ces remerciements dans 60 ans…

Résumé

Ce manuscrit aborde le problème du transport à la demande (*Dial-A-Ride Problem* – DARP) utilisant des véhicules privés et des sommets alternatifs (*Dial-A-Ride Problem using Private Vehicles and Alternative Nodes* – DARP-PV-AN). Le DARP consiste à créer des tournées de véhicules afin d'effectuer les transports de différents clients. Chaque requête correspond à un client ayant besoin d'être transporté de son point d'origine à son point de destination. Les différents coûts de transport doivent être minimisés tout en respectant un ensemble de contraintes comme les fenêtres de temps ou encore la durée maximale d'une tournée.

Dans le DARP classique, les véhicules partent d'un dépôt et doivent y revenir à la fin de leur tournée. Dans le DARP-PV-AN proposé, le service de transport à la demande peut être assuré par les véhicules classiques (appelés véhicules publics) ou par les clients utilisant leurs propres véhicules (appelés véhicules privés). L'utilisation de ces véhicules ajoute plus de flexibilité et permet aux clients d'organiser leurs propres transports. Cependant, cela soulève également des problèmes de confidentialité. Le DARP-PV-AN propose également une solution à ces problèmes et met l'accent sur la confidentialité des adresses, c'est-à-dire la possibilité d'empêcher des tiers de connaître la localisation des clients en préservant leurs lieux de départ et d'arrivée. Pour cela, plusieurs sommets d'origine/destination sont ajoutés pour chaque demande de transport.

Le DARP-PV-AN est tout d'abord traité dans un cadre mono-objectif. Un modèle linéaire est présenté et une métaheuristique de type ELS (*Evolutionary Local Search*) est proposée pour calculer des solutions de bonne qualité. Ces deux méthodes sont testées sur un ensemble d'instances classiques. Un nouvel ensemble d'instances dédiées au DARP-PV-AN est également fourni pour tester la méthode ELS.

Une seconde approche de résolution du DARP-PV-AN mono-objectif est proposée. Basée sur une métaheuristique hybride, elle est divisée en trois parties. Un algorithme génétique est couplé à une méthode de « séparation et évaluation » (*Branch and Bound*) ainsi qu'à l'utilisation de la programmation linéaire en nombre entier (PLNE) pour résoudre le problème de couverture par ensembles avec poids (*Weighted Set Cover Problem* – WSCP). Cette approche est testée sur les nouvelles instances et est comparée à l'ELS proposé précédemment.

Cette méthode hybride est ensuite adaptée pour résoudre le DARP-PV-AN dans un contexte bi-objectif. Pour cela, un algorithme génétique de type NSGA-II est utilisé. Ce dernier est associé à une méthode de *Branch and Bound* et à la résolution du WSCP bi-objectif par PLNE. Pour la résolution du WSCP bi-objectif, différentes méthodes d'agrégation ont été testées et comparées. Cette approche est également testée sur les nouvelles instances qui ont été modifiées pour les aborder dans un contexte bi-objectif.

Abstract

This manuscript addresses the Dial-A-Ride Problem (DARP) using Private Vehicles and Alternative Nodes (DARP-PV-AN). The DARP consists of creating vehicle routes in order to ensure a set of users' transportation requests. Each request corresponds to a client needing to be transported from his/her origin to his/her destination. Routing costs have to be minimized while respecting a set of constraints like time windows and maximum route length.

In the classical DARP, vehicles have to start from a depot and come back to it at the end of their route. In the DARP-PV-AN, the on-demand transportation service can be done either by a public fleet or by clients that use their private vehicles. The use of these vehicles adds more flexibility and unclog the public transportation fleet by allowing clients to organize their own transportation. However, it also raises some privacy concerns. The DARP-PV-AN addresses these concerns and focuses on location privacy, i.e. the ability to prevent third parties from learning clients' locations, by keeping both original and final location private. This is addressed by setting several pickup/delivery nodes for the transportation requests, thus masking the private address.

The DARP-PV-AN is first solved in a mono-objective context. A compact mixed integer linear model is presented and an Evolutionary Local Search (ELS) is proposed to compute solutions of good quality for the problem. These methods are benchmarked on a modified set of benchmark instances. A new set of realistic instances is also provided to test the ELS in a more realistic way.

A second approach solving the mono-objective DARP-PV-AN is proposed. Based on a hybrid metaheuristic, it combines three methods: a genetic algorithm, a Branch and Bound method, and Integer Linear Programming (ILP) to solve the Weighted Set Cover Problem (WSCP). This approach is tested on the new instances and compared to the previously proposed ELS.

This hybrid method is then tuned to solve the DARP-PV-AN in a bi-objective context. In order to do this, the multi-objective algorithm NSGA-II is used, always associated with a Branch and Bound method and bi-objective WSCP solving by ILP (Integer Linear Programming). For the bi-objective WSCP resolution, different aggregation methods were tested and compared. This approach is also tested on the new instances, modified to treat them in a bi-objective framework.

Sommaire

Introduction

Les problèmes de transport forment une classe de problèmes de recherche opérationnelle et d'optimisation combinatoire. Résoudre de tels problèmes consiste à déterminer un ensemble de tournées affectées à différents véhicules. Ces problèmes couvrent un large domaine théorique et possèdent également une forte application dans le monde réel : la livraison de biens dans le cadre de la grande distribution, les chaînes logistiques, les transports publics (avions, trains, métro, bus, etc) ou encore privés (BlaBlacar, Taxi, Uber, etc). Ces différents problèmes apparaissent également dans le service d'aide à la personne (transport de personnes handicapées ou âgées, visites médicales, etc).

L'augmentation globale des coûts de transport et de la densité du trafic routier (embouteillages) ont accentué l'attention portée aux moyens de transports alternatifs. Aujourd'hui, la plupart des véhicules privés sont sous-utilisés car ils ne transportent que le conducteur. La mise au point de nouvelles applications sur smartphone, basées sur la localisation géographique via les informations GPS, permettent d'utiliser les véhicules privés de manière partagée. Cependant, la vie privée des utilisateurs de tels systèmes peut être mise à mal : des logiciels malveillants peuvent accéder aux données privées des utilisateurs et faciliter l'identification de leur domicile pour organiser différentes agressions ou cambriolages. Une des propositions de cette thèse est un modèle de transport à la demande permettant l'utilisation conjointe de véhicules publics et privés tout en préservant la vie privée des utilisateurs.

Les critères d'optimisation des systèmes de transport sont très variés : minimisation des distances parcourues, minimisation de la date de fin du dernier élément visité (*makespan*), minimisation du coût financier ou encore maximisation du nombre de clients/biens transportés. La notion de qualité de service est également de plus en plus étudiée dans la littérature et s'articule autour de différentes notions : les temps d'attentes des clients dans les véhicules, les dates de début et de fin de la tournée, la durée totale de la tournée, les temps de trajet de chaque clients, etc.

En plus des critères d'optimisation précédemment présentés, les problèmes de transport doivent respecter un ensemble de contraintes. Parmi les contraintes des problèmes étudiés on peut citer :

- Les contraintes de capacité maximale des véhicules (quantité, poids, taille, etc) ;
- Les contraintes liées aux fenêtres de temps associées à chaque client, ces derniers devant être desservis dans l'intervalle de temps imposé ;
- Les contraintes de distance/durée maximale d'une tournée (carburant, temps de travail maximal pour le conducteur, etc) ;
- Les contraintes entre les ressources et les clients (par exemple dans le domaine médical où les médecins transportés dans les véhicules possèdent des compétences particulières qui ne correspondent qu'aux besoins de certains patients) ;
- Les contraintes de collecte et livraison, où les marchandises/clients doivent être déplacés des sites de collecte vers des sites de livraison.

Les problèmes de transport sont pour la plupart des problèmes d'optimisation combinatoires (ou optimisation discrète) : ils consistent à trouver dans un ensemble discret la solution réalisable maximisant (ou minimisant) une ou plusieurs fonctions objectifs. La difficulté réside dans la taille de l'ensemble discret, généralement exponentielle en la taille des données. Ainsi, parcourir l'ensemble des solutions pour trouver l'optimum est irréalisable en un temps raisonnable si la taille du problème à résoudre est importante. Au cours de cette thèse le problème du transport à la demande est traité avec différentes approches. Principalement

heuristiques, ces dernières permettent de fournir des solutions de bonne qualité en des temps de calculs acceptables, mais ne garantissent pas l'optimalité des résultats.

Une autre approche de résolution consiste en l'application de méthodes hybrides. Il est en effet possible de séparer un problème d'optimisation en différents sous-problèmes : certains peuvent être résolus à l'optimum en des temps raisonnables, et d'autres doivent être abordés par des méthodes heuristiques. Un couplage des méthodes de résolution exactes et heuristiques pour obtenir de meilleurs résultats sur le problème global est également proposé.

Ce manuscrit de thèse est structuré en quatre chapitres.

Le premier chapitre introduit les éléments nécessaires à la bonne compréhension des chapitres suivants. Il présente d'abord les problèmes de tournées de véhicules et du transport à la demande. Le concept de sécurité des clients et des véhicules est ensuite abordé. Les principes de complexité algorithmique et de classes de complexité sont également définis. Enfin, il présente les méthodes de résolution exactes et approchées les plus répandues dans la littérature pour les problèmes de transport.

Le second chapitre présente une nouvelle variante du problème de transport à la demande (*Dial-A-Ride Problem* – DARP) : le problème du transport à la demande avec véhicules privés et sommets alternatifs (*Dial-A-Ride Problem with Private Vehicles and Alternative Nodes* – DARP-PV-AN). Un modèle linéaire est proposé ainsi qu'une métaheuristique de type ELS, (*Evolutionary Local Search* – ELS). Basée sur les travaux de (Chassaing et al., 2016), cette dernière fournit une fonction d'évaluation efficace ainsi qu'une méthode de recherche locale basée sur un processus d'apprentissage. Ces deux propositions sont testées sur un ensemble d'instances de la littérature ainsi que sur un nouvel ensemble proposé dans ce chapitre.

Le troisième chapitre propose une seconde approche pour la résolution du DARP-PV-AN basée sur l'hybridation de méthodes de résolution exactes et approchées. Le problème de transport est ainsi divisé en trois sous-problèmes : l'affectation des clients dans les véhicules effectuée par un algorithme génétique, le calcul optimal de l'ordre de visite des clients dans chaque tournée effectué par un algorithme de *Branch and Bound,* et « l'assemblage » optimal des différentes tournées pour former des solutions réalisables par PLNE. Cet « assemblage » optimal des différentes tournées pour former des solutions est assimilé à la résolution du problème de couverture par ensembles avec poids (*Weighted Set Cover Problem* – WSCP). Cette approche est testée et comparée à la méthode ELS fournie dans le chapitre 2 et les résultats montrent l'intérêt d'une méthode hybride pour la résolution de ce problème.

Enfin, le quatrième chapitre propose une approche du DARP-PV-AN dans un contexte bi-objectif. La méthode proposée dans le chapitre 3 est ainsi adaptée en utilisant un algorithme NSGA-II pour l'affectation des clients dans les véhicules et différentes méthodes d'agrégation pour la résolution du WSCP bi-objectif par PLNE. Les algorithmes sont testés sur les instances proposées dans le chapitre 2 et permettent d'obtenir une bonne répartition des solutions sur les fronts de Pareto.

Les travaux de recherches réalisés durant cette thèse ont donné lieu aux publications présentées ci-après.

Publications dans le cadre de cette thèse

Brevet, D., Lacomme, P., & Duhamel, C. (2019). A Genetic Algorithm for the Dial-A-Ride Problem with private vehicles and privacy settings. In ROADEF 2019.

Brevet, D., Lacomme, P., Duhamel, C., & Iori, M. (2019). A Dial-A-Ride Problem using Private Vehicles and Alternative Nodes. Journal on Vehicle Routing Algorithms, 2019.

Brevet, D., Lacomme, P., & Duhamel, C. (2018). An ELS for the Dial-A-Ride Problem with private vehicles. In Congrès ROADEF 2018.

Brevet, D., Lacomme, P., Duhamel, C., & Iori, M. (2018). Modelling a Dial-A-Ride Problem with private vehicles. In Congrès ROADEF 2018.

Publications précédant cette thèse

Saber, T., Brevet, D., Botterweck, G., & Ventresque, A. (2018). Is seeding a good strategy in multi-objective feature selection when feature models evolve? Information and Software Technology, 95, 266-280.

Brevet, D., Saber, T., Botterweck, G., & Ventresque, A. (2016). Preliminary study of multi-objective features selection for evolving software product lines. In International Symposium on Search Based Software Engineering (pp. 274-280). Springer, Cham.

Références

Chassaing, M., Duhamel, C., Lacomme, P., 2016. 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. https://doi.org/10.1016/j.engappai.2015.10.002

CHAPITRE 1

Contexte général de l'étude

Objectifs du chapitre :

Dans ce premier chapitre, un rappel est proposé sur les problèmes de base des tournées de véhicules, et plus particulièrement sur le problème du transport à la demande qui est le sujet principal de cette thèse : ce problème consiste à transporter un ensemble de clients de leur point de départ jusqu'à leur point d'arrivée en utilisant une flotte de véhicules hétérogène.
La notion de sécurité lors de transports, et plus particulièrement lors de l'utilisation de moyens de covoiturage est également abordée : cette méthode de déplacement consiste à partager le même véhicule lorsque des clients ont des trajets similaires. Dans ce cas, le transport n'étant plus effectué par une entreprise publique mais par des personnes privées la notion de sécurité prend alors une importance particulière.
Enfin, des rappels sur les notions de complexité algorithmique ainsi que des différentes méthodes de résolution des problèmes de tournées de véhicules sont également proposés.

1.1 Les tournées de véhicules

Les problèmes de transport, représentent une classe de problèmes très étudiés dans le domaine de la recherche opérationnelle et de l'optimisation combinatoire. Dans ces problématiques, une flotte de véhicules (un ou plusieurs) doit visiter un ensemble de lieux avant de revenir à son point d'origine.

1.1.1 Problème du voyageur de commerce (TSP)

Parmi ces problèmes, le plus connu est celui du voyageur de commerce (*Traveling Salesman Problem* – TSP), problème classique d'optimisation combinatoire dont les premières publications sont antérieures à 1950 (Robinson, 1949). Ce problème consiste, étant donné une liste de villes ainsi que les distances les séparant, à déterminer le plus court chemin visitant chaque ville une unique fois et revenant au point d'origine. Une méthode efficace pour modéliser ce problème est l'utilisation des graphes. D'après la formulation de (Dantzig et al., 1954) une instance est un graphe complet $G = (V, A, C)$ avec V un ensemble de sommets, A un ensemble d'arrêtes, et C une fonction de coût associant à tout arc $(i, j) \in A$ la plus courte distance du sommet i au sommet j. La tournée optimale correspond au plus court cycle hamiltonien dans le graphe G. Malgré la simplicité de son énoncé, on ne connait pas d'algorithme permettant la résolution du problème à l'optimum en temps polynomial, et sa version décisionnelle associée fait partie des 21 problèmes NP-complets introduits par (Karp, 1972).

1.1.2 Problèmes de tournées de véhicules (VRP)

Les problèmes de tournées de véhicules (*Vehicle Routing Problem* – VRP) sont une extension du TSP introduisant la notion de flotte de véhicules. Dans cette variante, il est nécessaire de déterminer les tournées de chaque véhicule afin de livrer une liste de clients (ou encore d'effectuer des visites), tout en minimisant le coût des trajets. La définition introduite par (Dantzig et Ramser, 1959) comporte plusieurs nouvelles contraintes :

- Un ensemble de taille K de véhicules disponibles ;
- Chaque véhicule k possède une capacité maximale C_k ;
- Chaque sommet i à une demande $q_{i,j}$ de produit j à satisfaire.

Il est communément admis que les problèmes de tournées de véhicules se classent en deux catégories (Ramdane-Cherif, 2002) en fonction de la position des clients à livrer : si ces derniers se trouvent sur des sommets du graphe, il s'agit d'un problème de tournées sur sommets (TSP, VRP, …) et si les clients sont sur les arcs du graphe, on parle d'un problème de tournées sur arcs. Le plus connu des problèmes de tournées sur arcs est le problème du postier chinois (*Chinese Postman Problem* – CPP) (Guan, 1962) consistant en la minimisation de la tournée d'un facteur passant au moins une fois par chaque rue de son secteur, ou encore le problème de tournées sur arcs (*Capacitated Arc Routing Problem* – CARP) (Golden et Wong, 1981) ou pour chaque arc, une demande doit être satisfaite par l'un des véhicules de la flotte disponible.

Les travaux de cette thèse portent sur une sous classe particulière des problèmes de tournées de véhicules à savoir les problèmes de tournées avec des demandes localisées sur les sommets : les problèmes généraux de collecte et de livraison (*General Pickup and Delivery Problems* – GPDP).

1.1.3 Problèmes généraux de collecte et livraison (GPDP)

Définis par (Savelsbergh et Sol, 1995), les problèmes de tournées de véhicules avec collecte et livraison (GPDP) ne consistent plus en la simple visite des différents sommets par une flotte de véhicules. En effet, des objets ou personnes doivent être transportés des points d'origine aux points de destination. Ces différents transports sont représentés par des demandes (ou requêtes), chacune ayant trois caractéristiques :

- Un sommet j du graphe où la demande doit être validée ;
- Un produit $x_i \in X$ associé au sommet considéré, X étant l'ensemble contenant les types de produits ;
- Une quantité q_j^i, concernant la quantité de produit x_i associée au sommet j.

Lorsque la demande q_j^i du produit x_i sur le sommet j est positive, on parle de demande de collecte, tandis qu'une demande négative modélise une demande de livraison.

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

- Toutes les demandes sont satisfaites ;
- Le chargement du véhicule ne dépasse jamais sa capacité maximale ;
- La somme des coûts des tournées est minimale.

Dans la littérature, les GPDP sont modélisés sur un graphe orienté complet $G = (V, A)$, avec l'ensemble des sommets $V = \{0, \dots, n\}$, 0 représentant le dépôt où la flotte de véhicule commence et fini ses tournées. L'ensemble $A = \{(i, j) : i, j \in V, i \neq j\}$ définit les arcs et à

chacun d'eux est associée une valeur $c_{ij} \geq 0$ représentant le coût pour aller du sommet i au sommet j. Chaque sommet i, y compris le dépôt, peut avoir un ensemble de demandes de collecte ou de livraison : chacune de ces valeurs est représentée par q_i^j, la demande en produit x_j sur le sommet i. La *Figure 1.1* propose une solution au GPDP composée de deux tournées sur une instance de 6 sommets. Chacune des tournées créées démarre au dépôt, visite une liste ordonnée de clients et se termine au dépôt.

D'après (Berbeglia et al., 2007), la famille des problèmes de collecte et livraison est classée selon deux principaux critères : la structure, qui concerne la disposition des sommets de collecte et de livraison, et le type des visites effectuées par le véhicule sur les sommets du graphe.

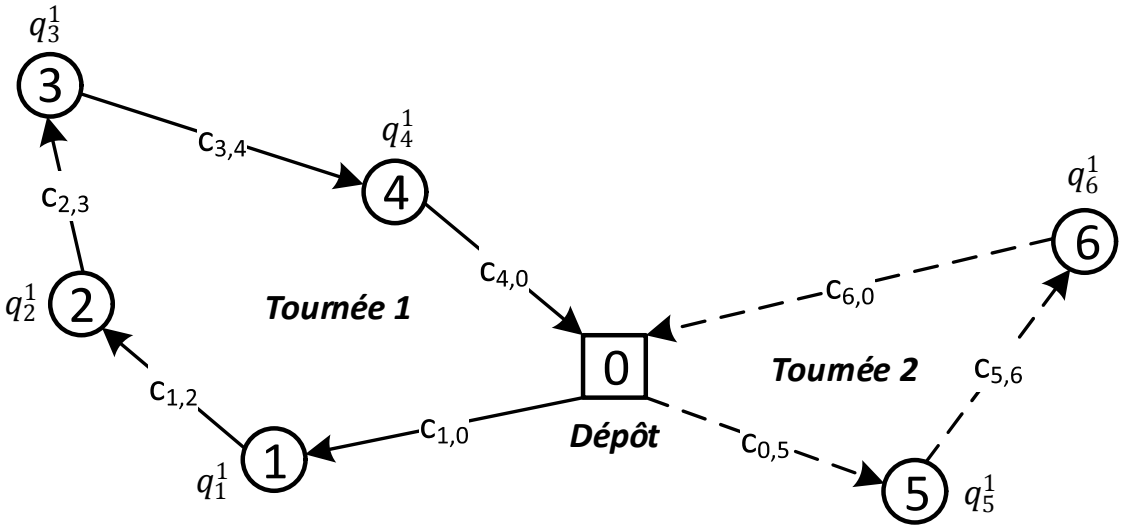
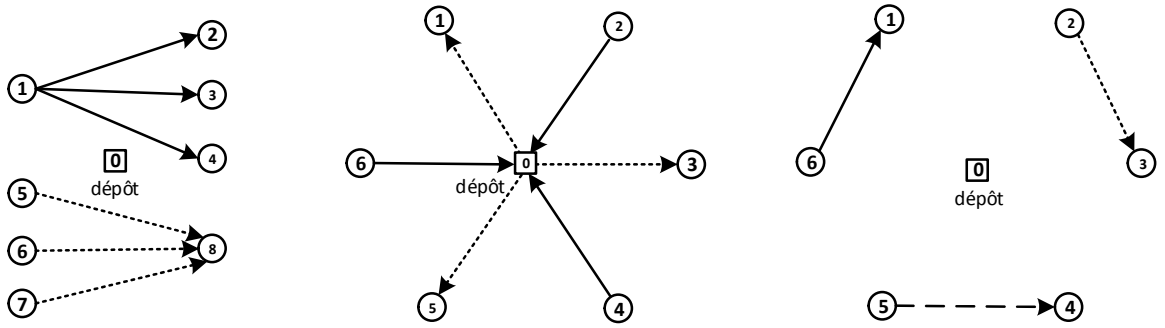


Figure 1.1 : Exemple de solution pour le GPDP avec 6 sommets et deux véhicules

a) La structure des demandes

La structure en plus de spécifier la disposition des sommets de collecte et de livraison, spécifie le nombre d'origines et de destinations liées à chaque demande. Elles sont généralement divisées en trois catégories représentées dans la *Figure 1.2* :

- *Many-to-Many* (M-M) : cette structure modélise les problèmes où il peut y avoir plusieurs sommets de collecte et plusieurs sommets de livraison pour un même produit ;
- *One-to-Many-to-One* (1-M-1) : cette structure de type centralisée comporte des demandes de collecte de différents sommets vers un sommet central (généralement le dépôt), et des demandes de livraison du sommet central vers d'autres sommets ;
- *One-to-One* (1-1) : cette dernière structure modélise des problèmes ou chaque demande de collecte est liée à une unique demande de livraison. Dans ce cas, on parle de requêtes de transport.



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

Figure 1.2 : Structures pour les GPDP, les arcs représentent les trajets que doivent effectuer les ressources (Ren, 2012)

b) Les types de visite

Le deuxième critère définit comment les opérations de collecte et de livraison sont effectuées à chaque sommet. Un problème de type PD indique que chaque sommet est visité exactement une fois pour un couple de collecte et de livraison. Si le problème est de type P-D, la demande combinée associée aux sommets peut être effectuée en plusieurs visites. Enfin, un problème de type P/D associe à chaque sommet une demande de type collecte ou livraison : à chaque sommet est donc associé une demande unique.

1.1.4 Les contraintes supplémentaires classiques

Il existe de nombreuses variantes des problèmes de collecte et livraison. Chacune de ces variantes implémente des contraintes supplémentaires afin de répondre à de nouvelles particularités. On peut notamment citer :

- Les fenêtres de temps ;
- Les flottes de véhicules hétérogènes ;
- La présence de plusieurs dépôts ;
- Le temps de trajet maximal pour chaque véhicule ;
- Le temps de trajet maximal pour chaque ressource.

a) Les fenêtres de temps

Les fenêtres de temps (*Time Windows* – TW) sont un ajout classique dans les problèmes de transports. Utilisées par (Dumas et al., 1991) dans un problème de collecte et livraison, elles imposent qu'à chaque sommet i du graphe soit associé un intervalle de temps $[e_i ; l_i]$ dans lequel le véhicule devra traiter la demande (*Figure 1.3*). L'intervalle est défini par deux valeurs : une date au plus tôt e_i et une date au plus tard l_i. Il est possible que le véhicule arrive avant la date au plus tôt et dans ce cas, ce dernier doit attendre avant d'effectuer son service. À l'inverse, si le véhicule arrive après la fin de la fenêtre de temps, la demande ne peut pas être satisfaite et la tournée n'est pas réalisable.

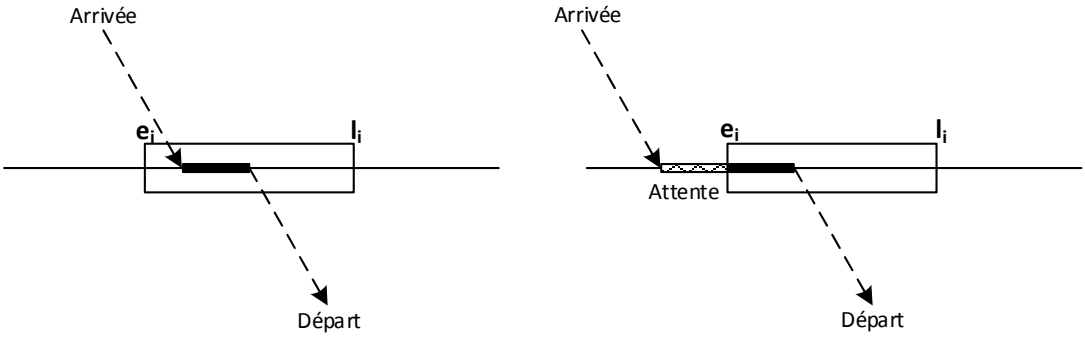


Figure 1.3 : Fenêtres de temps associées à des sommets

Dans la littérature, la variante avec fenêtres de temps souple (*Soft Time Windows* – STW) permet au véhicule de violer les fenêtres de temps mais une pénalité proportionnelle au retard du véhicule est appliquée dans la fonction objectif. Cette variante a été utilisée dans de nombreux problèmes de tournées de véhicules comme le TSP (Sexton et Choi, 1986), le VRP (Balakrishnan, 1993; Taillard et al., 1997) ou encore sur les problèmes de collecte et livraison (Bettinelli et al., 2014).

b) Les flottes de véhicules hétérogènes

Les problèmes avec flotte hétérogène impliquent de gérer un parc de véhicules avec des caractéristiques différentes. Suivant les variantes, les différences entre les véhicules peuvent être multiples :

- La charge maximale du véhicule, qui peut être éventuellement une charge maximale pour chaque type de produit suivant le type de véhicule utilisé ;
- Un coût fixe d'utilisation ;
- Un coût variable lié aux kilomètres parcourus ;
- Un type de produit lié à certains types de véhicules.

De nombreux travaux ont abordés les variantes des flottes hétérogènes. Initialement proposé par (Golden et al., 1984), la variante de base prend en compte une flotte de véhicules illimitée avec des capacités différentes (*Vehicle Fleet Mix Problem* – VFMP). Lorsque la flotte possède une taille fixe et des capacités variables selon les véhicules, l'extension est appelée (*Vehicle Routing Problem with Heterogeneous fleet* – VRPHE) et une méthode de résolution a été proposée par (Taillard, 1999). Enfin, si les véhicules possèdent à la fois des coûts fixes et variables et que la flotte est illimitée, les auteurs (Gendreau et al., 1999) parlent de (*Heterogeneous Vehicle Routing Problem* – HVRP). Sur ce dernier problème, on peut notamment citer (Li et al., 2007) ou encore (Brandão, 2011).

c) Les problèmes multi-dépôts

Les problèmes multi-dépôts sont des problèmes pour lesquels le graphe G comporte plusieurs dépôts d'où les véhicules commencent et finissent leur tournée (*Figure 1.4*). Le dépôt initial d'un véhicule n'est pas obligatoirement son dépôt final une fois la tournée terminée. Enfin la répartition des véhicules à chaque dépôt peut être différente au début de la journée. Ces problèmes font partie des sous problèmes de localisation et de tournées (*Location Routing Problem* – LRP), consistant à faire l'affectation des véhicules aux dépôts en plus de résoudre un problème de type VRP. Un état de l'art a été publié par (Prodhon et Prins, 2014).

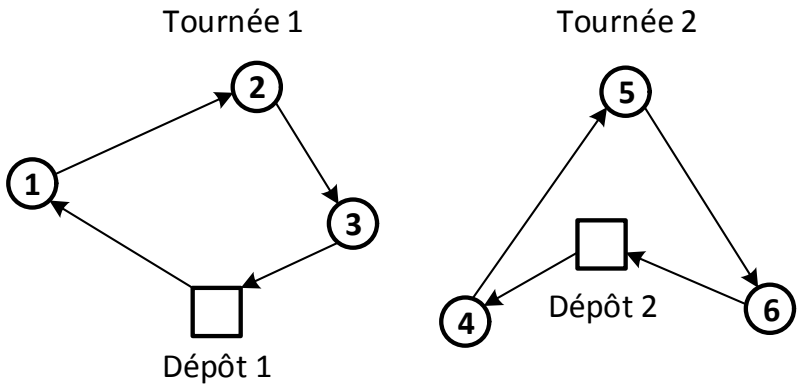


Figure 1.4 : Exemple de solution sur un problème de tournées de véhicules comportant deux dépôts

d) Le temps de trajet maximal pour chaque véhicule

L'ajout de cette contrainte oblige chaque véhicule à respecter un temps de trajet maximal. Ce temps de trajet est calculé comme la différence entre la date de départ du dépôt et la date de retour. Cette contrainte permet par exemple, de modéliser le temps de travail du chauffeur ou encore l'autonomie maximale d'un véhicule (essence ou électrique) avant une recharge potentielle (Felipe et al., 2014). Le temps de trajet maximal introduit par (Cordeau et Laporte, 2003a), est également présent dans le problème de transport de clients prenant en compte la qualité de service (*Dial-A-Ride Problem* – DARP).

e) Le temps de trajet maximal pour chaque ressource

Les ressources transportées par les véhicules sont très diverses. Certaines peuvent être périssables (dans le cas de produits frais), ou encore doivent être transportées rapidement de leur point de départ jusqu'à leur point d'arrivée (transport hospitalier). Lorsque de telles contraintes apparaissent, une durée maximale de transport est affectée à chaque marchandise (ou personne) à transporter. Ainsi, après avoir collecté le produit, le véhicule dispose d'un délai maximal pour atteindre le sommet de livraison et effectuer sa livraison. Cette contrainte permet également de modéliser la notion de qualité de service dans le domaine de transport de personnes (DARP) afin d'éviter de trop importants détours.

1.1.5 Les problèmes classiques de type GPDP

Un rappel des problèmes principaux de type collecte et livraison (GPDP) est proposé par (Chassaing, 2015) sur la *Figure 1.5*. Cette représentation est divisée en trois branches principales : *Many-to-Many* (M-M), *One-to-Many-to-One* (1-M-1) et *One-to-One* (1-1). Chaque arc représente des contraintes supplémentaires présentées dans la section précédente (section 1.1.4)

Le problème du transport à la demande (DARP) fait partie de la famille *One-to-One* et est présenté dans la section suivante (section 1.2).

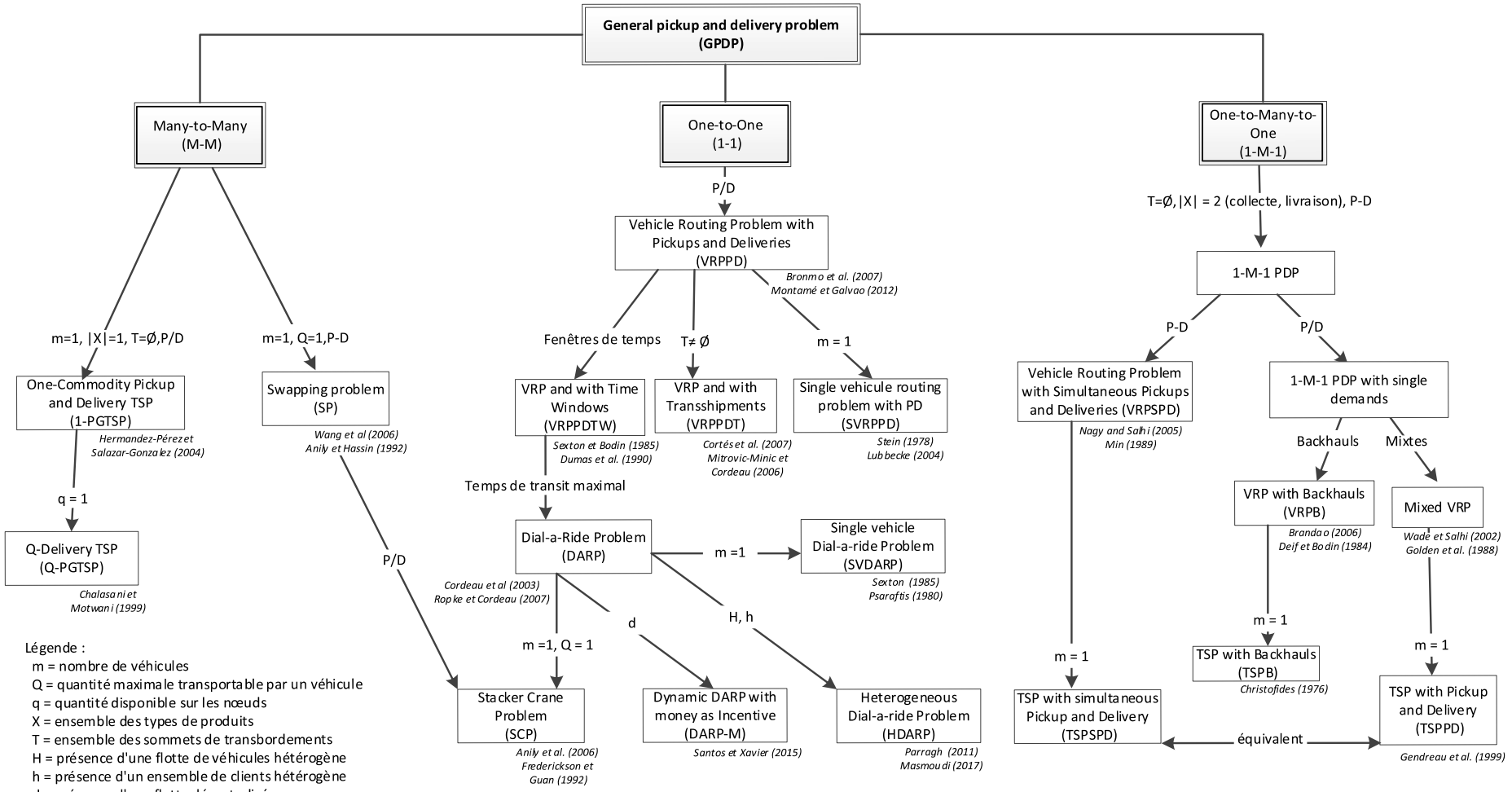


Figure 1.5 : Résumé des problèmes de type GPDP issue de (Chassaing, 2015)

1.2 Le problème du transport à la demande

1.2.1 Les particularités du DARP

Le problème du transport à la demande (DARP) est un dérivé du problème général de collecte et livraison. De type *One-to-One*, c'est une variante du problème VRPPDTW ajoutant principalement une durée de transport maximale pour chaque requête. Les différents éléments à transporter étant des clients, un ensemble de contraintes responsables de la qualité de service a été ajouté. Initialement, la contrainte de durée de transport maximale pour chaque requête définit la qualité de service du point de vue du client. Tout comme un grand nombre de problème de tournées de véhicules, l'objectif est de minimiser la distance totale parcourue par la flotte (Stein, 1978).

Le problème du transport à la demande est originalement associé au domaine hospitalier ainsi qu'aux personnes à mobilité réduite (personnes âgées ou handicapées). D'autres contraintes modifiant le DARP sont apparues afin de modéliser différemment la qualité de service (Paquette et al., 2009) :

- Limitation du temps d'attente des clients dans les véhicules immobiles ;
- Diminution du temps de trajet maximal du véhicule ;
- Respect des fenêtres de temps ;
- Temps de trajet entre les lieux de collecte/livraison du véhicule et les lieux de départ/arrivé des clients (marche à pied par exemple) ;
- Présence assurée d'une place pour fauteuil roulant ;
- Etc…

Ces différents critères peuvent être ajoutés au modèle sous la forme de contraintes à respecter (Cordeau et Laporte, 2007) ou bien intégrés dans la fonction objectif comme une pénalité en cas de violation (Jorgensen et al., 2007; Parragh et al., 2010).

1.2.2 Les premières apparitions du DARP

Une des premières apparitions du DARP est dans le livre « *Scheduling Algorithms for a dial-a-ride system* » proposé par (Wilson et al., 1971). Deux modèles, résolus par une approche heuristique, sont proposés par (Stein, 1978) : un modèle statique ou l'ensemble des demandes est connu à l'avance, et un modèle dynamique ou certaines requêtes apparaissent alors que certains véhicules sont déjà sur leurs trajets. (Psaraftis, 1980) propose un algorithme exacte de programmation dynamique sur un DARP mono-véhicule : leur méthode par pondération minimise le temps nécessaire à servir l'ensemble des clients et maximise la satisfaction des clients en minimisant leur temps d'attente et leur temps de trajet. Dans cet article, les clients n'ont pas de fenêtres de temps associées.

(Jaw et al., 1986), propose la première méthode de résolution du DARP multi-véhicules. Le modèle et la méthode de résolution proposés sont plus applicables dans le monde réel :

- Flotte hétérogène ;
- Fenêtres de temps ;
- Fonction objectif optimisant la distance parcourue par les véhicules et la qualité de service associée aux clients ;
- Heuristique d'insertion permettant des solutions sur de très grandes instance (>2000 clients).

1.2.3 Les extensions du DARP

Un grand nombre de publications consacrées au DARP se concentrent sur la minimisation des coûts d'exploitation des fournisseurs (y compris, mais sans s'y limiter, au temps de transport et à la durée des tournées) et seuls quelques-uns traitent des émissions polluantes (CO2) des véhicules, comme (Atahran et al., 2014).

La plupart des publications proposent des approches avec un unique objectif à optimiser. Cependant, certains articles sont consacrés au DARP multi-objectif, et utilisent le concept de pondération pour optimiser plusieurs objectifs au sein d'un même critère. L'optimisation lexicographique des fonctions objectifs est utilisée dans (Garaix et al., 2010) et convient bien aux problèmes où un premier critère est définitivement plus important que le second. Les approches d'optimisation d'un front de Pareto pour le DARP fournissent un ensemble complet de solutions non dominées et permettent d'analyser les compromis entre les objectifs (Paquette et al., 2013).

Quels que soient les objectifs à optimiser, le DARP a d'abord été introduit dans un contexte déterministe, puis étendu à la version stochastique, afin de permettre de prendre des décisions lorsque des données sont représentées par des variables aléatoires (Chassaing, 2015). Le DARP en version dynamique consiste à traiter des demandes qui sont reçues tout au long de la journée en maximisant le nombre de requêtes de transport traitées comme expliqué par (Attanasio et al., 2004).

La variante multi-dépôts a été introduite par (Detti et al., 2017) en résolvant un problème de transport issu d'un système de santé réel : dans ce problème, plusieurs dépôts sont disponibles ainsi qu'une flotte de véhicules hétérogènes.

L'homogénéité de la flotte de véhicules est une hypothèse classique du DARP. Elle peut être étendue à des flottes hétérogènes lorsque les clients ont des exigences et des attentes de services spécifiques. Ces considérations peuvent être modélisées par une matrice de compatibilité véhicule-client et ont été abordées dans plusieurs publications, notamment par (Parragh et al., 2012) ou (Detti et al., 2017).

Enfin, l'utilisation de différents moyens de transport au sein d'un même trajet est de plus en plus fréquente. Les transferts de clients entre véhicules nécessitent une synchronisation entre ces derniers. Cela offre une certaine flexibilité et peut conduire à de meilleures performances du système (Reinhardt et al., 2013).

1.2.4 Les méthodes actuelles de résolution du DARP

Dans l'étude proposée par (Cordeau et Laporte, 2003a), les auteurs définissent un modèle du DARP avec les critères de qualité de service suivant :

- Le temps total d'attente ;
- Le temps total de trajet des clients ;
- La durée totale des tournées.

Dans leur méthode de résolution, les auteurs utilisent une fonction d'évaluation permettant de savoir si une tournée respecte les contraintes. De plus, cette fonction minimise la durée totale de la tournée tout en minimisant les temps d'attentes des clients ainsi que les temps passés dans les véhicules. Enfin, un ensemble d'instances de référence utilisé par l'ensemble de la communauté a été fourni.

En 2003 et 2007, les mêmes auteurs ont publié un état de l'art des différentes variantes du DARP (Cordeau et Laporte, 2003b; Cordeau et Laporte, 2007).

Le *Tableau 1.1* regroupe les publications principales du domaine des transports à la demande depuis l'article de (Cordeau et Laporte, 2003a). Les méthodes de résolution sont comparées sur les instances de (Cordeau et Laporte, 2003a) notées [1] et celles de (Ropke et al., 2007), notées [2]. Ce tableau est construit sur le même modèle que celui proposé par (Cordeau et Laporte, 2003a). Il est composé de 7 colonnes :

- *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.

L'article de (Cordeau et Laporte, 2003a)[1] présente un ensemble d'instances avec une méthode de résolution approchée de type recherche tabou. Puis, (Ropke et al., 2007)[2], proposent une méthode exacte de type *Branch and Cut* ainsi qu'un ensemble d'instances de tailles plus réduites afin de les résoudre à l'optimum.

Plus récemment, certaines publications ont amélioré les résultats obtenus sur les jeux d'instances classiques de la littérature. (Braekers et al., 2014) ont proposé deux algorithmes : une méthode exacte de type *Branch and Cut* ainsi qu'une métaheuristique *Deterministic Annealing* (DA). Puis, (Gschwind et Irnich, 2014), ont proposé une méthode exacte de type *Branch and Price* résolvant toutes les instances [2] avec des temps moyens de quelques secondes. Enfin, (Gschwind et Drexl, 2019) ont proposé une métaheuristique de type *Adaptative Large Neighborhood Search* (ALNS) sur les instances [1].

1.3 La sécurité des clients et des véhicules

L'émergence des téléphones portables et des équipements connectés a renforcé la notion d'un monde sous surveillance : chaque information personnelle comme les données de santé, l'historique d'achats ou encore les données de mobilité peuvent être récupérés par des sociétés de services et vendues à des sociétés tierces comme les compagnies d'assurance. L'utilisation de plus en plus importante des mobiles a également permis le développement de nouvelles sociétés de services dans le domaine du transport, et plus particulièrement le covoiturage, tels qu'*Uber* ou *Blablacar*. Ce moyen de transport offre un certain nombre d'avantages pour le conducteur et les utilisateurs :

- Amortir les coûts de transport (essence, autoroute) du conducteur ;
- Proposer des trajets à prix réduit pour les utilisateurs ;
- Proposer des trajets plus rapides lorsque certaines routes dédiées à ces véhicules sont disponibles dans certains pays ;
- Réduire les émissions de CO_2 (Caulfield, 2009).

Cependant, les services de covoiturage actuels sauvegardent et utilisent potentiellement les données d'utilisateurs comme le lieu de résidence ou des informations financières. Ces données peuvent être utilisées pour déduire certaines informations supplémentaires sur les utilisateurs (Gambs et al., 2012), comme la prédiction de leurs déplacements. La société de covoiturage *Uber* a subit de telles attaques : ses chauffeurs ont été victimes d'agressions par des chauffeurs de taxi, tout comme certains utilisateurs. Ces agressions ont été possibles à cause de la capacité de localisation précise du véhicule.

Tableau 1.1 : Etat de l'art sur le DARP depuis l'article de (Cordeau et Laporte, 2003a)

Références	**Type de flotte**	**Fonction objectif**	**Fenêtres de temps**	**Autres contraintes**	**Algorithmes**	**Taille des instances**
[1](Cordeau et Laporte, 2003a)	Homogène	Min distance	Sommets d'origine ou de destination	max. route max. transport	Métaheuristique *recherche Tabou*	24 ≤ n ≤ 295 3 ≤ m ≤ 13
[2](Ropke et al. 2007)	Homogène	Min distance	Sommets d'origine ou de destination	définition de [1]	Méthode exacte *Branch and Cut*	16 ≤ n ≤ 96 2 ≤ m ≤ 8
(Parragh et al. 2009)	Homogène	Min distance Min temps trajet	Sommets d'origine ou de destination	définition de [1]	Métaheuristique *Deux phases : VNS + PR*	Instances [2]
(Parragh et Schmidt, 2013)	Homogène	Min distance	Sommets d'origine ou de destination	définition de [1]	Métaheuristique *Génération de colonnes + LNS*	Instances [1], [2]
(Masson et al. 2014)	Homogène	Min distance	Sommets d'origine et de destination	définition de [1] transbordement	Métaheuristique *ALNS*	Instances [1], [2]
(Braekers et al. 2014)	Hétérogène	Min distance	Sommets d'origine et de destination	définition de [1] multi-dépôts	Méthode exacte *Branch and Cut* Métaheuristique *Deterministic Annealing*	Instances [1], [2]
(Gschwind et Irnich, 2014)	Homogène	Min distance	Sommets d'origine ou de destination	définition de [1]	Méthode exacte *Branch and Price*	Instances [2]
(Chassaing et al. 2016)	Homogène	Min distance	Sommets d'origine ou de destination	définition de [1]	Métaheuristique *ELS*	Instances [1], [2]
(Gschwind et Drexl, 2019)	Homogène	Min distance	Sommets d'origine ou de destination	définition de [1]	Métaheuristique *ALNS*	Instances [1]

Les auteurs (Beresford et Stajano, 2003) définissent la notion de localisation confidentielle (*location privacy*) comme l'impossibilité pour une entité extérieure d'apprendre et prédire les positions géographiques passées, présentes et futures d'un utilisateur. Les méthodes de préservation de la vie privée dans le cadre des problèmes de transports ont été classées en deux catégories principales par (Cottrill, 2009) :

- Les méthodes basées sur les lois interdisant à un tiers de commettre une violation de la vie privée en raison de la menace de sanctions. ;
- Les méthodes techniques basées sur l'utilisation d'algorithmes de protections de la vie privée (*Tableau 1.2*). Ces dernières visent à réduire le taux de réussite d'un tiers au moyen de plusieurs approches comme l'utilisation de cryptographie avancée, ou l'anonymisation des utilisateurs.

Tableau 1.2 : Aperçu des principales méthodes de protection par (Aïvodji et al., 2016)

Méthode	**Description**
Aggregation	Regroupe les différentes traces de déplacement afin de rendre plus difficile l'indentification des utilisateurs. (Castelluccia, 2007; Cohen et al., 2001)
Anonymity	k-anonymat requiert que chaque ensemble de traces de déplacement qui ne se distinguent pas les unes de autres, contienne au moins k enregistrements. (Chen et al., 2013; Sweeney, 2002),
Cloaking	Brouille les traces de déplacement d'un utilisateur. (Cheng et al., 2006; Gedik et Ling Liu, 2006)
Encryption	Chiffre les données des utilisateurs avant de les diffuser. (Bilogrevic et al., 2014; Xi et al., 2014)
Geographical Masking	Perturbe l'emplacement d'origine en ajoutant du bruit. (Armstrong et al., 1999; Kwan et al., 2004)
Mix zone	Les *Mix-zones* sont des régions dans lesquelles les emplacements des utilisateurs ne sont pas enregistrés. De plus, le pseudonyme d'un utilisateur entrant diffère de celui qu'il aura lors de sa sortie. (Beresford et Stajano, 2003; Freudiger et al., 2007)
Pseudonyms	Remplace les identifiants des utilisateurs par des pseudonymes. (Bettini et al., 2009; Buttyán et al., 2007)

1.4 Complexité algorithmique

Dès les débuts de l'informatique moderne et la résolution des premiers problèmes par des algorithmes dédiés, les scientifiques se sont rendu compte de la nécessité d'une méthode permettant de comparer l'efficacité des algorithmes entre eux (temps, espace). En effet, les mesures des années 50 étaient souvent dépendantes de différents facteurs : le processeur, les différents temps d'entrées-sorties (moyen de stockage, affichage), le compilateur et le langage

de programmation. Il était donc nécessaire de trouver une nouvelle approche indépendante des différents facteurs extérieurs. L'un des premiers à appliquer systématiquement ce genre de méthode fut (Knuth, 1997) dès les premières versions de sa série *The Art of Computer Programming* datant de 1968. L'approche utilisée est le comportement asymptotique de l'algorithme dans le pire des cas, lorsque la taille des entrées tend vers l'infini : pour cela les notations de Landau sont utilisées.

1.4.1 Complexité en temps et notation de Landau

L'estimation de la complexité en temps d'un algorithme cherche à donner une borne supérieure sur le nombre d'opérations élémentaires, exprimé par un ordre de grandeur. Par exemple, pour la résolution du TSP par énumération, on parle d'une complexité en $O(n!)$ ce qui signifie qu'il existe une constante k telle que pour toute entrée de taille n, le nombre d'étapes sera inférieur à $k \times n!$, lorsque n temps vers l'infini (Knuth, 1976).

Plus formellement, une fonction $|f|$ est bornée par $|g|$ asymptotiquement, à un facteur près si et seulement si :

$$f(n) = O(g(n)) \quad \Leftrightarrow \quad \exists k > 0, \exists n_0, \quad tel\ que\ \forall n > n_0, \quad |f(n)| \leq |g(n)| \times k$$

Cette écriture, appelée notation de Landau, permet d'estimer et de comparer l'efficacité de plusieurs algorithmes sans la perturbation de facteurs extérieurs. Les notations Ω et Θ sont également très utilisées et permettent respectivement de déterminer une borne inférieure et un encadrement de l'algorithme.

Le *Tableau 1.3* représente une échelle de comparaison des différentes fonctions couramment utilisées en analyse, de la plus rapide à la plus lente (lorsque n tend vers l'infini et avec c, une constante supérieure à 1).

Tableau 1.3 : Échelle de comparaison

Notation	Grandeur
$O(1)$	Constante
$O(\log(n))$	Logarithmique
$O(n)$	Linéaire
$O(n^c)$	Polynomiale
$O(c^n)$	Exponentielle
$O(n!)$	Factorielle

Lors de l'étude des différents problèmes algorithmiques, une observation a été faite sur la facilité à résoudre certains problèmes, ou au contraire, la très grande difficulté à trouver des solutions. Cette analyse a permis de classer les problèmes en différentes catégories : les classes de complexité.

1.4.2 Classes de complexité

La complexité d'un problème peut être spatiale (les algorithmes qui le résolvent nécessitent une place importante) ou bien temporelle (le temps nécessaire à la résolution est démesuré). Dans ce manuscrit, la complexité temporelle est étudiée en priorité. L'étude de la complexité d'un problème permet de déterminer si sa résolution est « facile », c'est-à-dire s'il existe un algorithme en temps polynomial en la taille des données pour le résoudre. Le but est de comprendre la difficulté des différents problèmes algorithmiques, de les organiser par classes et d'étudier les relations entre elles. Une classe de complexité est donc un ensemble de problèmes dont la résolution nécessite une même quantité de ressources (spatiale ou temporelle) : deux problèmes issus d'une même classe de complexité seront aussi difficiles à résoudre l'un que l'autre. Parmi les différentes classes existantes, P et NP sont importantes lors de l'étude des problèmes de tournées de véhicules.

a) La classe P

La classe P contient tous les problèmes de décision pouvant être résolus sur une machine de Turing déterministe en temps polynomial par rapport à la taille des données d'entrée. Ainsi, un problème appartient à la classe P si on peut écrire un algorithme de complexité au pire polynomiale ($O(n^c)$) en la taille des données pour le résoudre. On peut citer le problème de l'arbre couvrant de poids minimal (*Minimum Spanning Tree* – MST) dans sa version décisionnelle consistant à trouver dans un graphe connexe un arbre connectant tous les sommets et dont la somme du poids des arrêtes est inférieure à k. De nombreux algorithmes existent pour résoudre ce problème comme (Borůvka, 1926; Prim, 1957) ou encore (Chazelle, 2000).

b) La classe NP

La classe NP contient tous les problèmes de décision pouvant être résolus sur une machine de Turing non déterministe en temps polynomial par rapport à la taille des données d'entrée. Une autre définition consiste à dire que l'on peut vérifier la solution en temps polynomial sur une machine de Turing déterministe. Cette classe à la particularité de contenir la plupart des problèmes d'optimisation combinatoire dans leur version décisionnelle. Par exemple, le TSP appartient à la classe NP : on peut vérifier en temps polynomial sur une machine de Turing déterministe que la solution est effectivement un cycle Hamiltonien d'une longueur inférieure à k.

c) Les problèmes NP-complets et NP-difficiles

La notion de complétude a été introduite par (Cook, 1971) et permet de déterminer dans une classe C l'ensemble des problèmes les plus difficiles à résoudre de cette classe. Plus formellement, un problème est C-difficile s'il est au moins aussi difficile que tous les problèmes situés dans C. Si un problème est C-difficile et appartient à la classe C, alors ce problème est C-complet.

Les problèmes NP-complets correspondent à une classe particulière : elle est formée de l'ensemble des problèmes appartenant à NP telle que tout problème appartenant à NP peut être transformé en eux via une réduction polynomiale. Ainsi, la résolution d'un problème NP-complet par un algorithme polynomial permettrait la résolution de tous les problèmes de la classe NP en temps polynomial. Trouver un tel algorithme confirmerai la conjecture $P = NP$. Valider (ou réfuter) cette conjecture est un des plus importants problèmes mathématiques contemporains et de nombreux travaux cherchent à résoudre en temps polynomial un

problème NP-complet. Le premier problème prouvé comme étant NP-complet est le problème SAT (*satisfiability*) par le théorème de Cook-Levin.

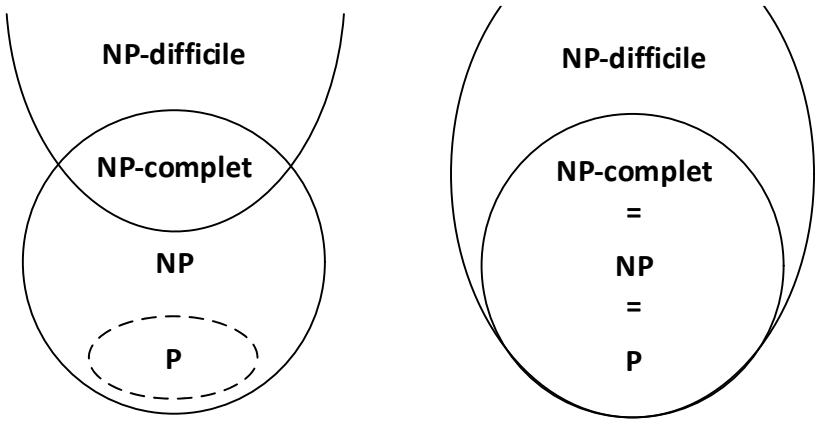


Figure 1.6 : Inclusion des classes si P≠NP (gauche) et si P=NP (droite)

Les problèmes NP-difficiles forment une classe vers laquelle tous les problèmes de la classe NP se ramènent par réduction polynomiale, mais ne sont pas nécessairement dans la classe NP. Les problèmes NP-difficiles peuvent donc être dans une classe plus large et donc plus difficile que la classe NP.

Enfin, les classes de complexité concernent les problèmes de décision, c'est-à-dire les questions dont la réponse est soit « oui » soit « non ». Les problèmes d'optimisation sont cités comme appartenant à une classe de complexité par abus de langage : il est possible de transformer tout problème d'optimisation en son problème de décision correspondant. En effet, à tout problème d'optimisation « trouver le minimum de $f(x)$ dans l'ensemble E » peut être associé le problème décisionnel : « Pour k donné, existe-t-il x tel que $f(x) \leq k$? ». Si le problème décisionnel est résoluble en temps polynomial alors le problème d'optimisation l'est également par recherche dichotomique sur k.

1.5 Méthodes de résolution

Les méthodes de résolution des problèmes de recherche opérationnelle, et plus particulièrement des problèmes de transport, peuvent être divisées en deux catégories principales :

- Les méthodes exactes ;
- Les méthodes approchées.

La particularité des méthodes exactes est de fournir la solution optimale à un problème donné. En revanche comme précisé dans la section 1.4.2, la plupart des problèmes d'optimisation appartiennent à la classe $NP \backslash P$ et on ne connait pas d'algorithme exact en temps polynomial pour les résoudre. Les algorithmes exacts actuels résolvant ces problèmes sont donc de complexité exponentielle et ne peuvent résoudre que des instances de taille raisonnable. Par exemple, les plus grosses instances de type DARP résolues actuellement ne dépassent pas les 100 clients, soit 200 sommets (Gschwind et Irnich, 2014).

Contrairement aux méthodes exactes, les méthodes approchées ne garantissent pas l'optimalité de la solution. En revanche, elles permettent de calculer des solutions beaucoup plus rapidement, et donc de traiter des plus grandes instances.

Un aperçu des différentes méthodes de résolution pour les problèmes de transport est proposé dans la *Figure 1.7*. Ces dernières sont divisées en deux catégories : les méthodes exactes décrites dans la section 1.5.1 et les méthodes approchées dans la section 1.5.2.

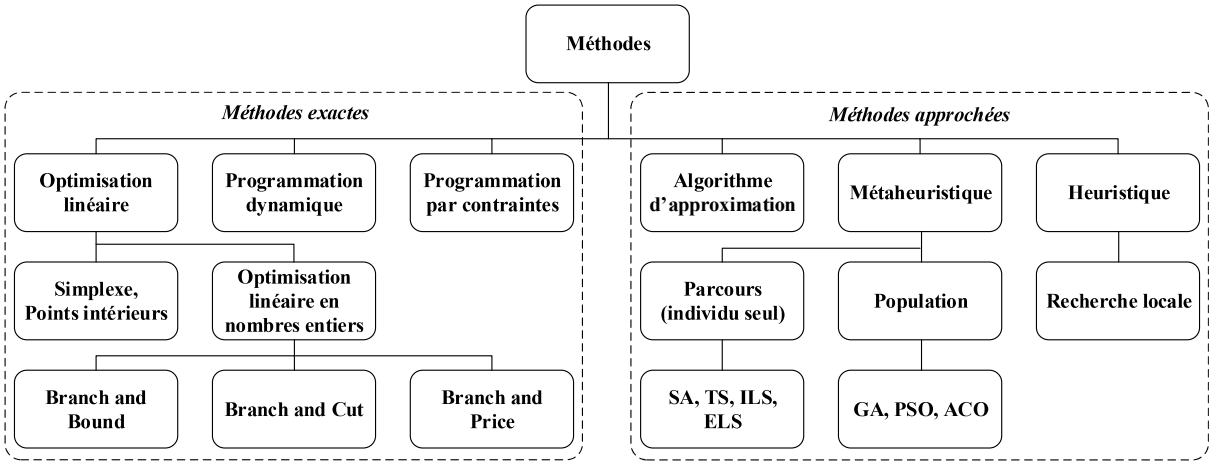


Figure 1.7 : Les différentes méthodes de résolutions pour les problèmes de transport

1.5.1 Les méthodes exactes

Les algorithmes exacts sont des méthodes fournissant une solution optimale à un problème donné. Pour un problème de décision appartenant à $NP \backslash P$, aucun algorithme de complexité polynomiale n'a encore été découvert : une première méthode consiste en l'énumération de toutes les possibilités, puis de sélectionner la meilleure. Cependant, le nombre de solutions étant exponentiel en la taille des données, cette première approche est inutilisable en pratique. Par exemple, pour un problème de type SAT, une telle méthode devrait tester 2^n solutions. Des méthodes différentes de l'énumération exhaustive existent et permettent d'accélérer la résolution exacte de certains problèmes (qu'ils appartiennent à la classe P ou $NP \backslash P$).

a) Programmation dynamique

Utilisée la première fois par (Bellman, 1954) dans les années 50, la programmation dynamique consiste à résoudre un problème initial donné en le décomposant en sous-problèmes. Une fois ces derniers résolus, la réponse à des problèmes plus grands est trouvée jusqu'à donner la solution optimale du problème initial.

Par exemple, pour calculer le $n^{ème}$ terme de la suite de Fibonacci, définie de la façon suivante :

$$F_n = \begin{cases} F_0 = 0, \\ F_1 = 1, \\ F_{n>1} = F_{n-1} + F_{n-2} \end{cases}$$

on peut utiliser un algorithme récursif classique. Un tel algorithme va effectuer plusieurs fois les mêmes calculs comme montré dans l'arbre d'appels récursif (*Figure 1.8*).

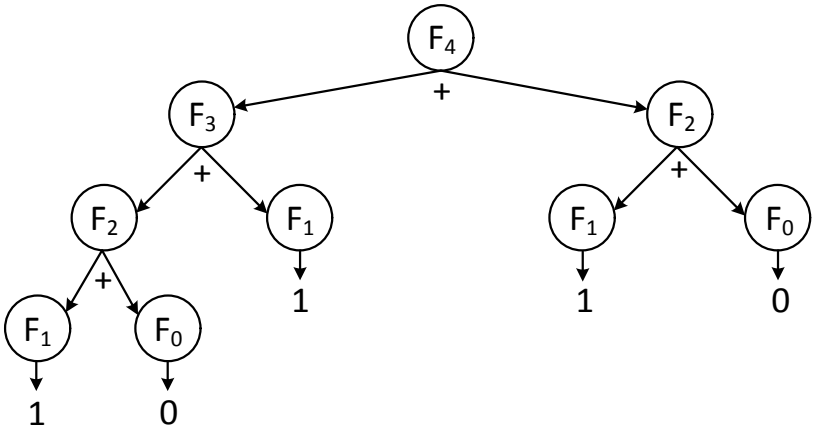


Figure 1.8 : Arbre d'appels de fonctions pour calculer le 4ème terme de Fibonacci. F_0, F_1 et F_2 sont calculées plusieurs fois

Avec la programmation dynamique, on sauvegarde chaque résultat intermédiaire afin d'éviter de le calculer à nouveau : ainsi, lors d'un appel récursif de la fonction de Fibonacci, on regarde dans un tableau T en position i le résultat de F_i : si ce dernier a déjà été calculé alors on retourne le résultat stocké, sinon on poursuit les appels récursifs. Grâce à cette méthode, on utilise et stocke les résultats des sous problèmes afin d'éviter de faire des calculs identiques à plusieurs reprises.

Les méthodes de résolution par programmation dynamique existent dans les problèmes de transports. Citons notamment l'algorithme de (Held et Karp, 1962) permettant de résoudre le TSP en $O(n^2 2^n)$: l'algorithme est toujours exponentiel mais reste plus rapide qu'une méthode énumérative de ce problème en $O(n!)$.

b) Programmation par contraintes

La programmation par contraintes (PPC) est apparue dans les années 70-80 (Haralick et Elliott, 1980; Montanari, 1974). C'est une technique de résolution de problèmes combinatoires consistant à modéliser un problème par un ensemble de relations logiques reliant des variables, chacune possédant un domaine sur lequel elle est définie. Ces contraintes ne sont pas obligatoirement linéaires et permettent une grande liberté dans la modélisation du problème.

Les algorithmes de résolution ont pour particularité d'utiliser activement les contraintes afin de réduire les domaines de définition des variables. Ces algorithmes utilisent la propagation de contraintes afin de filtrer le domaine d'une variable et supprimer les valeurs ne pouvant pas apparaitre dans une solution réalisable : ainsi, l'espace de recherche des solutions est fortement réduit. Les méthodes de filtrages sont ensuite utilisées dans des algorithmes de recherche arborescente afin d'élaguer les branches n'amenant qu'à des violations de contraintes. (Rossi et al., 2006) détaillent la programmation par contraintes dans leur livre "*Handbook of Constraint Programming"*.

c) Programmation linéaire

La programmation linéaire est un outil permettant de modéliser et résoudre toute une classe de problèmes d'optimisation. Elle consiste en la minimisation d'une fonction linéaire sur un polyèdre convexe défini par un ensemble de contraintes linéaires. On pourra décrire plus généralement un problème d'optimisation linéaire par la notation suivante :

$$\begin{cases} \min c^T x \\ Ax \leq b \end{cases}$$

avec $x \in \mathbb{R}^n$ un vecteur de variables réelles, et les données représentées par les vecteurs $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et la matrice $A \in \mathbb{R}^{n \times m}$.

Les problèmes linéaires sont simples à résoudre : l'algorithme du simplexe (dont les bases ont été découvertes en 1947) proposé par (Dantzig et al., 1955) ou encore la méthode des points intérieurs avec l'algorithme de (Karmarkar, 1984) sont très efficaces. En revanche, lorsque l'on ajoute des contraintes d'intégrité, c'est-à-dire que l'on oblige les variables à être entières, les problèmes deviennent beaucoup plus complexes : la programmation linéaire en nombre entier (PLNE) est NP-difficile.

Pour résoudre des problèmes de PLNE, une méthode appelée « séparation et évaluation » (*Branch and Bound)* proposée en 1960 par (Land et Doig, 2010), peut être appliquée.

Comme expliqué précédemment, une méthode naïve pour résoudre les problèmes d'optimisation combinatoire minimisant une fonction consiste en l'énumération de toutes les solutions, puis de sélectionner celle donnant le coût minimal. Grâce à une méthode de type *Branch and Bound*, il est possible d'éviter d'énumérer certaines solutions qui seront obligatoirement sous-optimales. Cette méthode est basée sur les deux phases suivantes :

1. La phase de séparation consistant en la division du problème initial en sous-problèmes formant une partition. Chaque sous-problème fourni une solution, et la comparaison de toutes les solutions des différents sous-problèmes permet de garder la meilleure trouvée, qui est celle du problème initial. Puisque cette phase de séparation est applicable récursivement, la séparation du problème initial formera un arbre de recherche.
2. La phase d'évaluation consistant en l'affectation d'une valeur à un nœud de l'arbre de recherche. Dans le cas d'un problème de minimisation, cette valeur est généralement une borne inférieure (minorant) du problème et est obtenue en relâchant certaines contraintes du problème initial. Trois cas sont possibles :
    - Le nœud n'apporte pas de solution respectant les contraintes conservées, ou bien son minorant est supérieur à la solution courante. Dans ce cas, il est inutile de poursuivre la séparation de la branche contenant ce nœud.
    - Le nœud contient une solution réalisable du problème initial et respecte donc toutes les contraintes. Comme précédemment, la séparation est inutile.
    - Le nœud contient un minorant meilleur que la solution courante mais n'étant pas une solution réalisable du problème initial. Dans ce cas, une phase de séparation sera à nouveau effectuée.

Dans le cas de la programmation linéaire en nombre entier, la méthode fonctionne comme indiqué dans l'exemple suivant :

Soit P un problème linéaire en nombre entier naturels à deux variables x_1 et x_2 maximisant une fonction f et soumis à deux contraintes, C_1 et C_2. Lors de la phase d'évaluation, la méthode de relaxation continue va être utilisée sur un sous-problème P_0 : celle-ci relâche les contraintes d'intégrité. Ainsi, un algorithme très efficace peut être utilisé afin de maximiser la fonction objectif, comme l'algorithme du simplexe. La solution trouvée (S_1) peut ainsi être composée de variables entières, auquel cas c'est la meilleure solution réalisable de P accessible dans P_0, ou bien être fractionnaire (*Figure 1.9*).

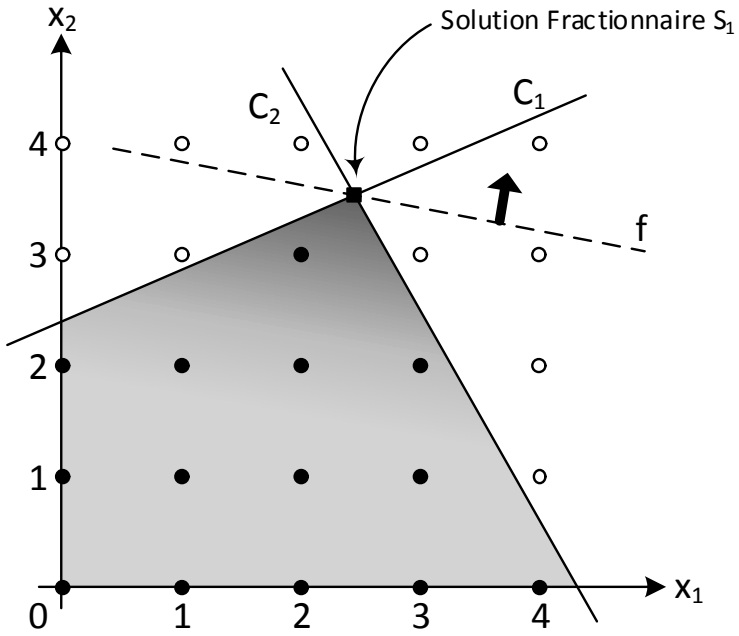


Figure 1.9 : Opération d'évaluation donnant une solution fractionnaire S_1

Une phase de séparation est donc nécessaire puisque les deux variables x_1 et x_2 ne sont pas entières. On va donc choisir une variable fractionnaire, par exemple x_1 ($x_1 = 2{,}44$), et fixer deux nouvelles contraintes $x_1 \leq 2$ et $x_1 \geq 3$, chacune apparaissant dans un sous-problème P_1 et P_2 (*Figure 1.10*).

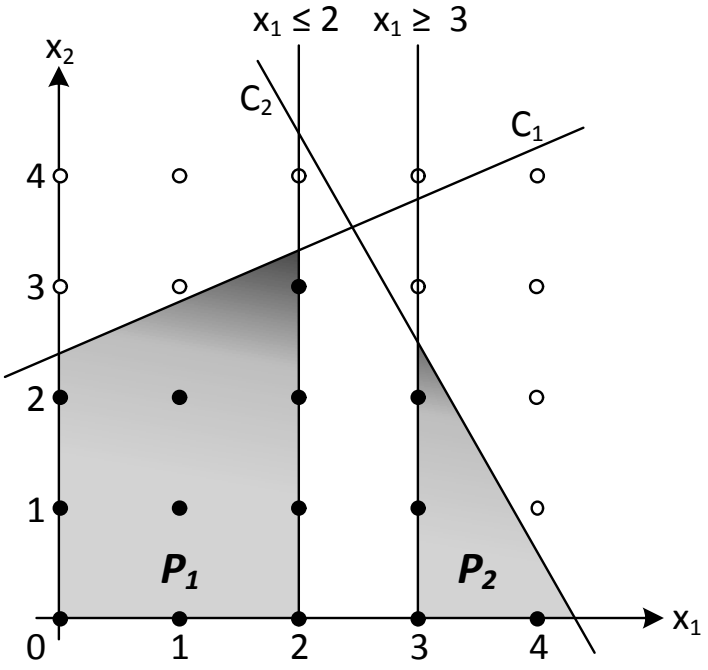


Figure 1.10 : Phase de séparation en deux sous-problèmes P_1 et P_2

Les liens entre les différents sous-problèmes sont modélisés sous la forme d'un arbre de recherche (aussi appelé arbre de décision). Dans cet arbre, le problème P_0 est la racine et les sous-problèmes P_1 et P_2 sont les premières branches créées (*Figure 1.11*).

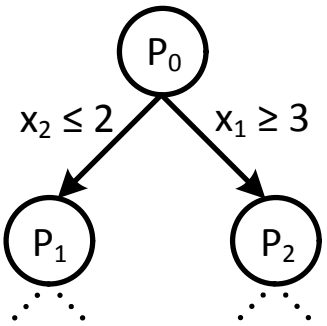


Figure 1.11 : Arbre de décision lors de la séparation de P_0

À chaque itération de l'algorithme, un sous-problème d'optimisation est évalué et une solution est obtenue : si l'ensemble des variables sont entières, alors la solution est valide, sinon le processus de séparation est appelé à nouveau sur une variable non entière. La méthode est répétée jusqu'à ce que toutes les feuilles de l'arbre aient été traitées. La solution optimale du problème P est la meilleure solution réalisable (avec variables entières) obtenue dans les feuilles de l'arbre.

D'autres méthodes de type *Branch and Bound* existent permettant un filtrage de l'arbre de décision plus poussé évitant ainsi certaines phases de séparations et accélérant la résolution des problèmes. Par exemple l'algorithme de *Branch and Cut* (Padberg et Rinaldi, 1991) couple le *Branch and Bound* avec la méthode des plans sécants ou encore l'algorithme de *Branch and Price* (Desrochers et Soumis, 1989) utilisant la génération de colonnes.

1.5.2 Les méthodes approchées

Les méthodes de résolution approchées ont la particularité de pouvoir donner de bonnes solutions beaucoup plus rapidement et ainsi traiter des instances de plus grandes tailles.

a) Algorithme d'approximation

Les algorithmes d'approximation sont des méthodes très souvent associées aux problèmes NP-difficiles et garantissant la qualité des solutions trouvées. Cette qualité est assurée par la capacité de ces méthodes à borner la solution découverte.

Plus généralement, un algorithme est une c-approximation si, pour un problème de minimisation ayant pour solution optimale S^*, la méthode fournie une solution $S \leq cS^*$. La variable c est appelée le facteur d'approximation. Un bon algorithme d'approximation donnera donc une méthode avec c le plus proche de 1 possible, pour un problème donné.

L'algorithme de (Christofides, 1976) fournit une 3/2-approximation du TSP dans le cas métrique, basé sur la résolution de deux problèmes plus simples appartenant à la classe de complexité P : le problème de l'arbre couvrant de poids minimal et le problème de couplage de poids minimum. Actuellement, le facteur 3/2 est le meilleur facteur d'approximation connu pour ce problème.

b) Heuristiques

Contrairement aux méthodes exactes ou aux algorithmes d'approximation, les méthodes heuristiques ne garantissent pas la qualité des solutions qu'elles retournent.

Une heuristique est généralement une stratégie de résolution empirique qui exploite la structure particulière du problème qu'elle cherche à résoudre. Par exemple, les algorithmes gloutons sont des heuristiques, comme la méthode du plus proche voisin pour le TSP. La plupart des méthodes de recherche locales sont des heuristiques car dédiées à une opération particulière sur un problème donné. Par exemple, la méthode 2-opt proposée par (Croes, 1958) est un algorithme itératif appliqué à la base au TSP.

Les principales limites des heuristiques sont qu'elles tombent assez facilement dans des minimas locaux et la qualité des résultats est fortement influencée par la solution initiale ainsi que les différents paramètres de réglage de l'heuristique.

c) Métaheuristiques

Très utilisées de nos jours, les métaheuristiques sont des schémas globaux, applicables et adaptables aux différents problèmes rencontrés. La plupart sont des algorithmes stochastiques itératifs ayant des méthodes de recherche favorisant la convergence vers un optimum global. Certains travaux (Fleury, 1993; Hajek, 1986) permettent de déterminer les conditions nécessaires et suffisantes démontrant la convergence de certaines métaheuristiques vers les solutions optimales. Elles intègrent toutes des mécanismes de diversifications, qui alternent avec des phases d'intensification. Une des qualités principale des métaheuristiques est leur capacité à se combiner à des méthodes heuristiques ou exactes déjà existantes afin d'améliorer leur vitesse de résolution et la qualité des solutions proposées. Elles sont principalement divisées en deux catégories :

- Les métaheuristiques à parcours (individu seul) consistent à faire évoluer une seule solution dans l'espace de recherche à chaque itération. On peut citer la méthode de recuit simulé (*Simulated Annealing* – SA), de recherche tabou (*Tabu Search* – TS) ou encore la méthode GRASP (*Greedy Randomized Adaptive Search Procedure*).
- Les métaheuristiques à population font évoluer un groupe de solutions en parallèle, ces dernières s'échangeant des informations afin de converger vers un optimum global. Les algorithmes génétiques (*Genetic Algorithm* – GA), les méthodes de colonies de fourmis (*Ant Colony Optimization* – ACO) ou encore l'optimisation par essaims particulaires (*Particle Swarm Optimization* – PSO) sont les plus connus.

Un état de l'art est proposé par (Gendreau et Potvin, 2010) expliquant chaque méthode ainsi que leurs domaines d'application et réussites.

1.6 Conclusion

Ce chapitre donne un aperçu des problèmes de tournées de véhicules en abordant leurs différentes variantes, contraintes et objectifs. Puis le problème particulier du transport à la demande (DARP), sujet central de cette thèse, est présenté. La notion de sécurité dans le domaine des transports est introduite, ainsi que les différentes méthodes de résolution. Les notions de base en complexité algorithmique sont rappelées permettant d'introduire les différentes méthodes de résolution. Divisées en deux catégories, chacune présente des avantages et inconvénients : les méthodes exactes fournissent des solutions optimales mais ne peuvent être utilisées sur des instances de grande taille à cause du principe d'explosion

combinatoire. En revanche, les métaheuristiques fournissent des solutions sur des grandes instances très rapidement, mais ne peuvent en garantir l'optimalité.

La particularité du DARP est d'inclure des contraintes de qualité de service modélisées différemment suivant ses variantes. Ces contraintes, spécifiques au DARP, permettent d'implémenter des problèmes plus réalistes car la qualité des services proposés est un facteur primordial dans le domaine des transports, et plus particulièrement celui des personnes âgées ou à mobilité réduite.

Les véhicules privés sont souvent utilisés, dans le domaine du transport, à titre personnel. De nombreux véhicules sont en effet occupés uniquement par le conducteur lors du trajet. La mise en place de systèmes de transport à la demande couplant les moyens de transports privés aux moyens plus classiques permettrait d'améliorer la qualité de service proposé ainsi que le nombre de clients transportés. L'utilisation de moyen de transports privés nécessite en revanche une approche sécurisant les usagers. Une des premières approches de cette thèse sera de fournir un modèle du DARP couplant les moyens de transports privés et publics et sécurisant l'utilisation de transports privés. Le deuxième point concernera l'étude économique de ces moyens de transports par un problème multi-objectif.

Références

Aïvodji, U.M., Gambs, S., Huguet, M.-J., Killijian, M.-O., 2016. Meeting points in ridesharing: A privacy-preserving approach. Transportation Research Part C: Emerging Technologies 72, 239‑253. https://doi.org/10.1016/j.trc.2016.09.017

Armstrong, M.P., Rushton, G., Zimmerman, D.L., 1999. Geographically masking health data to preserve confidentiality 29.

Atahran, A., Lenté, C., T'kindt, V., 2014. A Multicriteria Dial-a-Ride Problem with an Ecological Measure and Heterogeneous Vehicles: MULTICRITERIA DIAL-A-RIDE PROBLEM. J. Multi-Crit. Decis. Anal. 21, 279 ‑ 298. https://doi.org/10.1002/mcda.1518

Attanasio, A., Cordeau, J.-F., Ghiani, G., Laporte, G., 2004. Parallel Tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Computing 30, 377‑387. https://doi.org/10.1016/j.parco.2003.12.001

Balakrishnan, N., 1993. Simple Heuristics for the Vehicle Routeing Problem with Soft Time Windows. Journal of the Operational Research Society 44, 279 ‑ 287. https://doi.org/10.1057/jors.1993.53

Bellman, R., 1954. The theory of dynamic programming. Bull. Amer. Math. Soc. 60, 503‑516. https://doi.org/10.1090/S0002-9904-1954-09848-8

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

Beresford, A.R., Stajano, F., 2003. Location privacy in pervasive computing. IEEE Pervasive Comput. 2, 46‑55. https://doi.org/10.1109/MPRV.2003.1186725

Bettinelli, A., Ceselli, A., Righini, G., 2014. A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows. Math. Prog. Comp. 6, 171‑197. https://doi.org/10.1007/s12532-014-0064-0

Bettini, C., Mascetti, S., Wang, X.S., Freni, D., Jajodia, S., 2009. Anonymity and Historical-Anonymity in Location-Based Services, in: Bettini, C., Jajodia, S., Samarati, P., Wang, X.S. (Éd.), Privacy in Location-Based Applications. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 1‑30. https://doi.org/10.1007/978-3-642-03511-1_1

Bilogrevic, I., Jadliwala, M., Joneja, V., Kalkan, K., Hubaux, J.-P., Aad, I., 2014. Privacy-Preserving Optimal Meeting Location Determination on Mobile Devices. IEEE Trans.Inform.Forensic Secur. 9, 1141 - 1156. https://doi.org/10.1109/TIFS.2014.2318435

Borůvka, O., 1926. O jisíém problému minimálním. 24.

Braekers, K., Caris, A., Janssens, G.K., 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. Transportation Research Part B: Methodological 67, 166‑186. https://doi.org/10.1016/j.trb.2014.05.007

Brandão, J., 2011. A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. Computers & Operations Research 38, 140 - 151. https://doi.org/10.1016/j.cor.2010.04.008

Buttyán, L., Holczer, T., Vajda, I., 2007. On the Effectiveness of Changing Pseudonyms to Provide Location Privacy in VANETs, in: Stajano, F., Meadows, C., Capkun, S., Moore, T. (Éd.), Security and Privacy in Ad-Hoc and Sensor Networks. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 129‑141. https://doi.org/10.1007/978-3-540-73275-4_10

Castelluccia, C., 2007. Securing Very Dynamic Groups and Data Aggregation in Wireless Sensor Networks, in: 2007 IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems. Présenté à 2007 IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, IEEE, Pisa, Italy, p. 1 - 9. https://doi.org/10.1109/MOBHOC.2007.4428625

Caulfield, B., 2009. Estimating the environmental benefits of ride-sharing: A case study of Dublin. Transportation Research Part D: Transport and Environment 14, 527‑531. https://doi.org/10.1016/j.trd.2009.07.008

Chassaing, M., 2015. Problèmes de transport à la demande avec prise en compte de la qualité de service. Clermont-Ferrand.

Chazelle, B., 2000. A minimum spanning tree algorithm with inverse-Ackermann type complexity. J. ACM 47, 1028‑1047. https://doi.org/10.1145/355541.355562

Chen, R., Fung, B.C.M., Mohammed, N., Desai, B.C., Wang, K., 2013. Privacy-preserving trajectory data publishing by local suppression. Information Sciences 231, 83‑97. https://doi.org/10.1016/j.ins.2011.07.035

Cheng, R., Zhang, Y., Bertino, E., Prabhakar, S., 2006. Preserving User Location Privacy in Mobile Data Management Infrastructures, in: Danezis, G., Golle, P. (Éd.), Privacy Enhancing Technologies. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 393‑412. https://doi.org/10.1007/11957454_23

Christofides, N., 1976. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem 11.

Cohen, N.H., Purakayastha, A., Turek, J., Wong, L., Yeh, D., 2001. Challenges in Flexible Aggregation of Pervasive Data 15.

Cook, S.A., 1971. The complexity of theorem-proving procedures, in: Proceedings of the Third Annual ACM Symposium on Theory of Computing - STOC '71. Présenté à the third annual ACM symposium, ACM Press, Shaker Heights, Ohio, United States, p. 151‑158. https://doi.org/10.1145/800157.805047

Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Ann Oper Res 153, 29‑46. https://doi.org/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. Transportation Research Part B: Methodological 37, 579‑594. https://doi.org/10.1016/S0191-2615(02)00045-0

Cordeau, J.-F., Laporte, G., 2003b. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. 4OR 1. https://doi.org/10.1007/s10288-002-0009-8

Cottrill, C.D., 2009. Approaches to Privacy Preservation in Intelligent Transportation Systems and Vehicle–Infrastructure Integration Initiative. Transportation Research Record 2129, 9‑15. https://doi.org/10.3141/2129-02

Croes, G.A., 1958. A Method for Solving Traveling-Salesman Problems. Operations Research 6, 791‑812. https://doi.org/10.1287/opre.6.6.791

Dantzig, G., Fulkerson, R., Johnson, S., 1954. Solution of a Large-Scale Traveling-Salesman Problem. OR 2, 393‑410. https://doi.org/10.1287/opre.2.4.393

Dantzig, G., Orden, A., Wolfe, P., 1955. The generalized simplex method for minimizing a linear form under linear inequality restraints. Pacific J. Math. 5, 183‑195. https://doi.org/10.2140/pjm.1955.5.183

Dantzig, G.B., Ramser, J.H., 1959. The Truck Dispatching Problem. Management Science 6, 80‑91. https://doi.org/10.1287/mnsc.6.1.80

Desrochers, M., Soumis, F., 1989. A Column Generation Approach to the Urban Transit Crew Scheduling Problem. Transportation Science 23, 1 - 13. https://doi.org/10.1287/trsc.23.1.1

Detti, P., Papalini, F., Lara, G.Z.M. de, 2017. A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. Omega 70, 1‑14. https://doi.org/10.1016/j.omega.2016.08.008

Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. European Journal of Operational Research 54, 7 - 22. https://doi.org/10.1016/0377-2217(91)90319-Q

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. Transportation Research Part E: Logistics and Transportation Review 71, 111‑128. https://doi.org/10.1016/j.tre.2014.09.003

Fleury, G., 1993. Méthodes stochastiques et déterministes pour les problèmes NP-difficiles. Clermont-Ferrand.

Freudiger, J., Raya, M., Félegyházi, M., Papadimitratos, P., Hubaux, J.-P., 2007. Mix-Zones for Location Privacy in Vehicular Networks 7.

Gambs, S., Killijian, M.-O., del Prado Cortez, M.N., 2012. Next place prediction using mobility Markov chains, in: Proceedings of the First Workshop on Measurement,

Privacy, and Mobility - MPM '12. Présenté à the First Workshop, ACM Press, Bern, Switzerland, p. 1‑6. https://doi.org/10.1145/2181196.2181199

Garaix, T., Artigues, C., Feillet, D., Josselin, D., 2010. Vehicle routing problems with alternative paths: An application to on-demand transportation. European Journal of Operational Research 204, 62‑75. https://doi.org/10.1016/j.ejor.2009.10.002

Gedik, B., Ling Liu, 2006. MobiEyes: A Distributed Location Monitoring Service Using Moving Location Queries. IEEE Trans. on Mobile Comput. 5, 1384‑1402. https://doi.org/10.1109/TMC.2006.153

Gendreau, M., Laporte, G., Musaraganyi, C., Taillard, É.D., 1999. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. Computers & Operations Research 26, 1153‑1173. https://doi.org/10.1016/S0305-0548(98)00100-2

Gendreau, M., Potvin, J.-Y. (Éd.), 2010. Handbook of metaheuristics, 2. ed. ed, International series in operations research & management science. Springer, New York, NY.

Golden, B., Assad, A., Levy, L., Gheysens, F., 1984. The fleet size and mix vehicle routing problem. Computers & Operations Research 11, 49‑66. https://doi.org/10.1016/0305-0548(84)90007-8

Golden, B.L., Wong, R.T., 1981. Capacitated arc routing problems. Networks 11, 305‑315. https://doi.org/10.1002/net.3230110308

Gschwind, T., Drexl, M., 2019. Adaptive Large Neighborhood Search with a Constant-Time Feasibility Test for the Dial-a-Ride Problem. Transportation Science 53, 480‑491. https://doi.org/10.1287/trsc.2018.0837

Gschwind, T., Irnich, S., 2014. Effective Handling of Dynamic Time Windows and Its Application to Solving the Dial-a-Ride Problem. Transportation Science 49, 335‑354. https://doi.org/10.1287/trsc.2014.0531

Guan, M., 1962. Graphic programming using odd and even points. Chinese Math 237‑277.

Hajek, B., 1986. Optimization by simulated annealing: a necessary and sufficient condition for convergence, in: Institute of Mathematical Statistics Lecture Notes - Monograph Series. Institute of Mathematical Statistics, Hayward, CA, p. 417‑427. https://doi.org/10.1214/lnms/1215540316

Haralick, R.M., Elliott, G.L., 1980. Increasing tree search efficiency for constraint satisfaction problems. Artificial Intelligence 14, 263‑313. https://doi.org/10.1016/0004-3702(80)90051-X

Held, M., Karp, R.M., 1962. A Dynamic Programming Approach to Sequencing Problems. Journal of the Society for Industrial and Applied Mathematics 10, 196‑210. https://doi.org/10.1137/0110015

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. Transportation Research Part B: Methodological 20, 243‑257. https://doi.org/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. Journal of the Operational Research Society 58, 1321‑1331. https://doi.org/10.1057/palgrave.jors.2602287

Karmarkar, N., 1984. A new polynomial-time algorithm for linear programming, in: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing - STOC '84. Présenté à the sixteenth annual ACM symposium, ACM Press, Not Known, p. 302‑311. https://doi.org/10.1145/800057.808695

Karp, R.M., 1972. Reducibility among Combinatorial Problems, in: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (Éd.), Complexity of Computer Computations. Springer US, Boston, MA, p. 85‑103. https://doi.org/10.1007/978-1-4684-2001-2_9

Knuth, D.E., 1997. The art of computer programming. Pearson Education.

Knuth, D.E., 1976. Big Omicron and big Omega and big Theta. SIGACT News 8, 18‑24. https://doi.org/10.1145/1008328.1008329

Kwan, M.-P., Casas, I., Schmitz, B., 2004. Protection of Geoprivacy and Accuracy of Spatial Information: How Effective Are Geographical Masks? Cartographica: The International Journal for Geographic Information and Geovisualization 39, 15‑28. https://doi.org/10.3138/X204-4223-57MK-8273

Land, A.H., Doig, A.G., 2010. An Automatic Method for Solving Discrete Programming Problems, in: Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (Éd.), 50 Years of Integer Programming 1958-2008. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 105 ‑ 132. https://doi.org/10.1007/978-3-540-68279-0_5

Li, F., Golden, B., Wasil, E., 2007. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. Computers & Operations Research 34, 2734‑2742. https://doi.org/10.1016/j.cor.2005.10.015

Montanari, U., 1974. Networks of constraints: Fundamental properties and applications to picture processing. Information Sciences 7, 95‑132. https://doi.org/10.1016/0020-0255(74)90008-5

Padberg, M., Rinaldi, G., 1991. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. SIAM Rev. 33, 60 ‑ 100. https://doi.org/10.1137/1033004

Paquette, J., Cordeau, J.-F., Laporte, G., 2009. Quality of service in dial-a-ride operations. Computers & Industrial Engineering 56, 1721 ‑ 1734. https://doi.org/10.1016/j.cie.2008.07.005

Paquette, J., Cordeau, J.-F., Laporte, G., Pascoal, M.M.B., 2013. Combining multicriteria analysis and tabu search for dial-a-ride problems. Transportation Research Part B: Methodological 52, 1‑16. https://doi.org/10.1016/j.trb.2013.02.007

Parragh, S.N., Cordeau, J.-F., Doerner, K.F., Hartl, R.F., 2012. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. OR Spectrum 34, 593‑633. https://doi.org/10.1007/s00291-010-0229-9

Parragh, S.N., Doerner, K.F., Hartl, R.F., 2010. Variable neighborhood search for the dial-a-ride problem. Computers & Operations Research 37, 1129 ‑ 1138. https://doi.org/10.1016/j.cor.2009.10.003

Prim, R.C., 1957. Shortest connection networks and some generalizations. The Bell System Technical Journal 1389‑1401.

Prodhon, C., Prins, C., 2014. A survey of recent research on location-routing problems. European Journal of Operational Research 238, 1 - 17. https://doi.org/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. Transportation Science 14, 130-154. https://doi.org/10.1287/trsc.14.2.130

Ramdane-Cherif, W., 2002. Problèmes d'optimisation en tournées sur arcs. Troyes.

Reinhardt, L.B., Clausen, T., Pisinger, D., 2013. Synchronized dial-a-ride transportation of disabled passengers at airports. European Journal of Operational Research 225, 106-117. https://doi.org/10.1016/j.ejor.2012.09.008

Ren, L., 2012. Conception et évaluation d'outils décisionnels pour des systèmes réactifs d'aide à la mobilité. Clermont-Ferrand.

Robinson, J., 1949. On the Hamiltonian game (a traveling salesman problem). Rand project air force arlington va.

Ropke, S., Cordeau, J.-F., Laporte, G., 2007. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks 49, 258 - 272. https://doi.org/10.1002/net.20177

Rossi, F., Van Beek, P., Walsh, T., 2006. Handbook of Constraint Programming, Foundations of Artificial Intelligence. Elsevier. https://doi.org/10.1016/S1574-6526(06)X8001-X

Savelsbergh, M.W.P., Sol, M., 1995. The General Pickup and Delivery Problem. Transportation Science 29, 17-29. https://doi.org/10.1287/trsc.29.1.17

Sexton, T.R., Choi, Y.-M., 1986. Pickup and Delivery of Partial Loads with "Soft" Time Windows. American Journal of Mathematical and Management Sciences 6, 369-398. https://doi.org/10.1080/01966324.1986.10737200

Stein, D.M., 1978. Scheduling Dial-a-Ride Transportation Systems. Transportation Science 12, 232-249. https://doi.org/10.1287/trsc.12.3.232

Sweeney, L., 2002. k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY. Int. J. Unc. Fuzz. Knowl. Based Syst. 10, 557 - 570. https://doi.org/10.1142/S0218488502001648

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y., 1997. A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. Transportation Science 31, 170-186. https://doi.org/10.1287/trsc.31.2.170

Taillard, E.D., 1999. A heuristic column generation method for the heterogeneous fleet VRP. RAIRO-Oper. Res. 33, 1-14. https://doi.org/10.1051/ro:1999101

Wilson, N.H., Sussman, J., Wong, H.-K., Higonnet, T., 1971. Scheduling Algorithms for a dial-a-ride system. Massachusetts Institute of Technology. Urban Systems Laboratory.

Xi, Y., Schwiebert, L., Shi, W., 2014. Privacy preserving shortest path routing with an application to navigation. Pervasive and Mobile Computing 13, 142-149. https://doi.org/10.1016/j.pmcj.2013.06.002

CHAPITRE 2

Résolution du problème de transport à la demande avec véhicules privés et sommets alternatifs

Objectifs du chapitre :

Ce chapitre aborde le problème du transport à la demande (*Dial-A-Ride Problem* – DARP) utilisant des véhicules privés et des sommets alternatifs (*Dial-A-Ride Problem with Private Vehicles and Alternative Nodes* – DARP-PV-AN). Tout d'abord, une définition détaillée du problème et de ses particularités est proposée.
Une formulation linéaire en nombre entier est présentée ainsi qu'une métaheuristique de type « recherche locale évolutive » (*Evolutionary Local Search* – ELS). Ces méthodes sont comparées à celles de la littérature sur les instances classiques du DARP. Un nouvel ensemble d'instances pour le DARP-PV-AN est également proposé afin de tester la méthode ELS de manière plus réaliste.

2.1 Définition du problème

Le problème du transport à la demande (DARP) est dérivé du problème général de collecte et livraison et a été abordé la première fois en 1971 par (Wilson et al., 1971). La définition moderne du problème est attribuée à (Cordeau et Laporte, 2003). Le contexte du problème est défini dans le chapitre précédent.

Une instance du DARP est constituée de n clients et m véhicules. Un client n'est pas constitué d'un seul sommet : c'est une requête de transport d'un sommet d'origine (*pickup*) à un sommet de destination (*delivery*). Une requête est également constituée du nombre de personnes à transporter.

Chaque tournée est effectuée par un véhicule et doit respecter un ensemble de contraintes comme la capacité maximale du véhicule ou encore les fenêtres de temps correspondant au début de service, liées à chaque sommet.

Le but est de minimiser la somme des distances parcourues par les véhicules pour desservir la totalité des clients tout en respectant l'ensemble des contraintes.

2.1.1 Problème du transport à la demande (DARP)

Formellement, le DARP est défini sur un graphe orienté complet $G = (N, A)$, avec une flotte hétérogène F de K véhicules et un ensemble $R = \{1, \dots, n\}$ de requêtes de transport. $N = \{0, 1, \dots, 2n, 2n + 1\}$ est l'ensemble des sommets du graphe. Le dépôt est séparé en deux copies n et $2n + 1$, pour respectivement le début et la fin des tournées. Pour chaque

requête $i \in R$, son *pickup* est i et son *delivery* est $n + i$ Ainsi, $N^P = \{1, \ldots, n\}$ et $N^D = \{n + 1, \ldots, 2n\}$ sont respectivement, les sous-ensembles de N des sommets *pickup* et *delivery*.

Pour chaque sommet $i \in N$:

- $[e_i\ ;\ l_i]$ est sa fenêtre de temps telle que e_i est la date de début de service au plus tôt et l_i la date de début de service au plus tard ;
- d_i est la durée du service ;
- q_i est le nombre de personnes entrant ou sortant, tel que $q_i > 0$ et $q_{n+i} = -q_i$.

Pour tout arc $(i, j) \in A$, t_{ij} correspond à la durée de transport du sommet i à j, et c_{ij} correspond au coût de transport : il est courant d'utiliser c_{ij} comme étant égal à t_{ij} ce qui est le cas dans cette thèse. Enfin, chaque véhicule $k \in F$ a une capacité maximale Q_k. Pour simplifier la notation, i^+ est utilisé pour représenter un sommet de *pickup* i, et i^- pour représenter un sommet de *delivery* $n + i$.

En suivant la notation proposée par (Cordeau et Laporte, 2003), cinq types de variables peuvent être utilisés pour modéliser l'aspect temporel du DARP. Illustrées dans la *Figure 2.1*, elles décrivent pour chaque sommet i :

- A_i la date d'arrivée d'un véhicule ;
- B_i la date de début de service ;
- D_i la date de départ telle que $D_i = B_i + d_i$;
- W_i le temps d'attente tel que $W_i = B_i - A_i$;
- R_i (*Riding Time*) le temps de trajet du client correspondant à la durée entre la fin de service de son *pickup* D_{i^+} et le début de service de son *delivery* B_{i^-}. Soit $R_i = B_{i^-} - D_{i^+}$.

Une tournée peut donc être représentée comme une séquence de sommets, commençant au sommet 0 et finissant au sommet $2n + 1$, telle que le sommet de *pickup* de chaque requête est situé avant le sommet de *delivery* associé. Une solution s est donc l'assignement d'une tournée à chaque véhicule, telle que chaque requête de transport est effectuée exactement une fois. Pour chaque route liée à un véhicule, l'affectation des variables A_i, D_i, et B_i est calculée pour respecter l'ensemble des contraintes de temps.

En résumé, une solution doit satisfaire l'ensemble des contraintes suivantes :

- Le *pickup* et le *delivery* d'une requête i doivent être dans la même tournée, le sommet de *pickup* étant visité avant le sommet de *delivery* ;
- À n'importe quel moment d'une tournée, le nombre de personnes dans le véhicule $k \in F$ ne doit pas excéder Q_k ;
- Pour chaque sommet i, la date de début de service B_i concorde avec la fenêtre de temps, i.e. $e_i \leq B_i \leq l_i$;
- La durée de trajet R_i d'un client ne doit pas excéder une limite RT_i ;
- La durée d'une tournée d'un véhicule k ne doit pas excéder une limite T ;
- Au maximum K véhicules peuvent être utilisés.

L'objectif du DARP est donc de trouver une solution réalisable s de durée de trajet minimal :

$$\min s = \sum_{k=1}^{K} \sum_{i=0}^{|n(k)|-1} c_{n_i(k) n_{i+1}(k)}$$

où $n(k)$ est la séquence de sommets pour le véhicule k et $n_i(k)$ est le sommet en position i dans $n(k)$.

Proposée par (Cordeau et Laporte, 2003), la qualité d'une solution est mesurée à travers les trois critères suivants :

- *Total Riding Time* : $TRT = \sum_{i \in R} B_{i^-} - D_{i^+}$
- *Total Waiting Time* : $TWT = \sum_{i \in N^P \cup N^D} W_i$
- *Total Duration* : $TD = \sum_{k \in F} A_{|n(k)|} - D_{n_0(k)}$

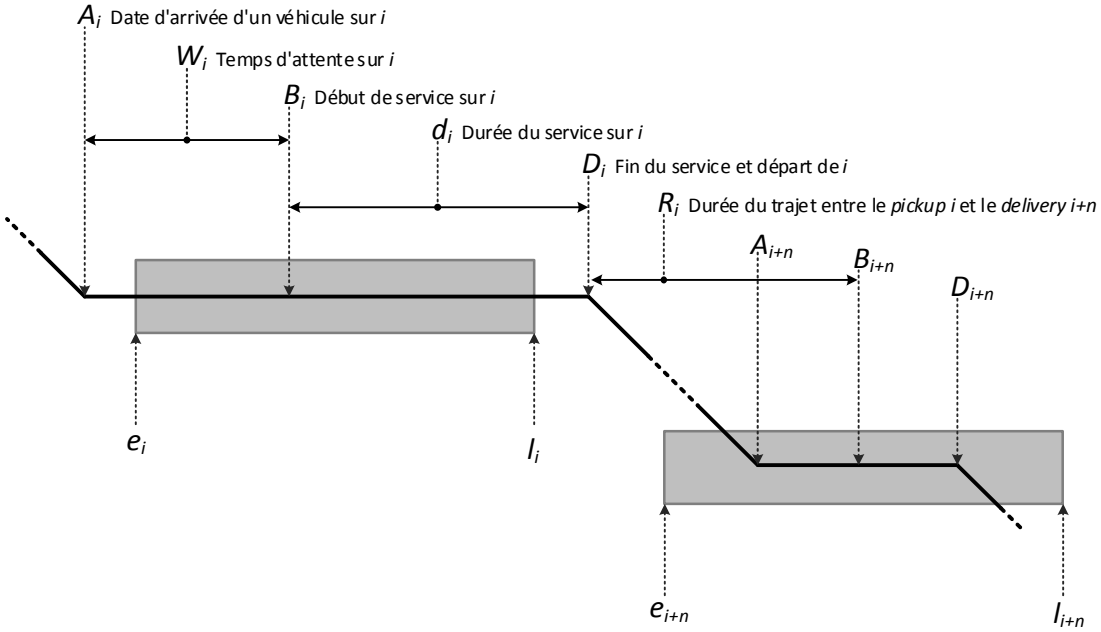


Figure 2.1 : Notations temporelles pour le DARP

2.1.2 DARP avec véhicules privés et sommets alternatifs (DARP-PV-AN)

Les embouteillages ou encore l'augmentation du prix de l'essence ont relancé l'intérêt porté aux moyens de transports alternatifs afin de réduire les durées de trajet et les effets environnementaux. Actuellement, la plupart des véhicules privés sont sous-utilisés et contiennent uniquement un seul voyageur (le conducteur) durant le trajet, laissant le reste des sièges inutilisés. La disponibilité croissante de systèmes de localisation géographique, principalement basés sur les informations GPS, permet la création de nouveaux types de services de transports personnalisés dans lesquels un véhicule privé peut être utilisé comme un véhicule privé partagé.

Par exemple, les services basés sur la localisation (*Location-Based Services* – LBS) (Artigues et al., 2012; Quercia et al., 2010) et les services de covoiturage (Agatz et al., 2012; Bruck et al., 2017; Furuhata et al., 2013) sont de plus en plus populaires. Au lieu d'être embauchés par une entreprise, les conducteurs d'un système de covoiturage peuvent être considérés comme des entités privées indépendantes. Ces nouveaux types de transport visent à réunir des voyageurs ayant des trajets et horaires similaires. Cependant, la vie privée des utilisateurs est une question importante dans les systèmes actuels de covoiturage (Cottrill et Thakuriah, 2015; Furuhata et al., 2013). Étant donné que ces services dépendent fortement d'applications Web, ils peuvent être ciblés par des logiciels malveillants afin d'accéder aux données privées de

utilisateurs. Parmi ces données, celles contenant des informations sur la localisation permettraient de deviner le domicile de l'utilisateur (Gambs et al., 2010) et ainsi permettre de potentiels cambriolages ou agressions. (Aïvodji et al., 2016) présentent un aperçu des principales techniques permettant de renforcer la confidentialité liée à la localisation.

Ces différents problèmes ont été examinés afin de proposer un modèle de DARP permettant l'utilisation de véhicules privés et publics de manière conjointe, tout en respectant la vie privée des utilisateurs.

Dans le DARP avec véhicules privés et sommets alternatifs (DARP-PV-AN), la flotte centralisée de véhicules publics est utilisée conjointement à une flotte décentralisée de véhicules privés. Cette flotte de véhicules privés est située dans un sous-ensemble de sommets $R' \subset R$, chaque sommet représentant un client ayant la possibilité d'utiliser son propre véhicule pour atteindre sa destination, éventuellement en transportant d'autres clients. Le système de covoiturage de sommet à sommet est une alternative aux moyens de transports en commun composé d'une flotte de véhicules centralisée en un dépôt. Les véhicules publics fournissent un système de transport porte à porte classique grâce à une plateforme centralisée. Cela peut modéliser des situations dans lesquelles une entreprise organise les transports de ses clients au moyen d'un parc de véhicules dédiés : les clients peuvent toujours organiser le transport eux-mêmes, comme c'est le cas s'ils sont des employés de l'entreprise.

Dans le cas où un client $i \in R'$ utilise son véhicule privé, la tournée de ce véhicule commence au sommet i^+ et se termine au sommet i^- : ces sommets correspondent aux points de départ et d'arrivé du client i. Un client i utilisant son véhicule privé ne peut pas faire de détours trop importants en faisant du covoiturage avec d'autres clients : cette limite de détour maximum est prise en compte par la limite du *riding time* RT_i du client (*Figure 2.2*).

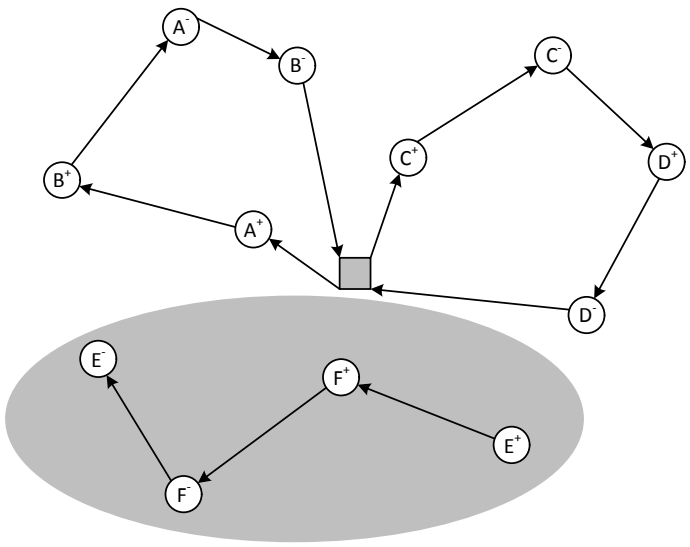


Figure 2.2 : Exemple de solution avec 2 véhicules publics et 1 véhicule privé. Le véhicule privé a une zone de détour maximale limitée par le riding time *maximal (zone grise) de son conducteur E*

La vie privée d'un client i est assurée par un ensemble de sommets de *pickup* alternatifs N_{i+}^S en plus de son sommet initial de *pickup* i^+ et par un ensemble de sommets de *delivery* alternatifs N_{i-}^S en plus de son sommet initial de *delivery* i^-. De cette manière, il est plus difficile de deviner l'emplacement exact de l'utilisateur. De plus, il est également plus difficile de prévoir où le client sera pris en charge et déposé. Si le client i est transporté par un

véhicule privé, un sommet $\lambda^+ \in N_{i^+}^S$ et un sommet $\lambda^- \in N_{i^-}^S$ doivent être sélectionnés, sinon les sommets de *pickup* et *delivery* initiaux i^+ et i^- sont utilisés.

Puisque deux sous-ensembles N_i^S et N_j^S peuvent partager certains sommets (*Figure 2.3*), ce problème permet l'utilisation de sommets communs (*meeting nodes*). Ainsi, deux (ou plus) clients i et j peuvent être pris en charge au même endroit si leurs sous-ensembles de sommets alternatifs possèdent des sommets communs. Ces derniers seront automatiquement utilisés chaque fois que cela améliorera le coût de la solution courante. Le DARP-PV-AN se réduit donc au DARP classique lorsque R' est vide et est par conséquent un problème NP-difficile.

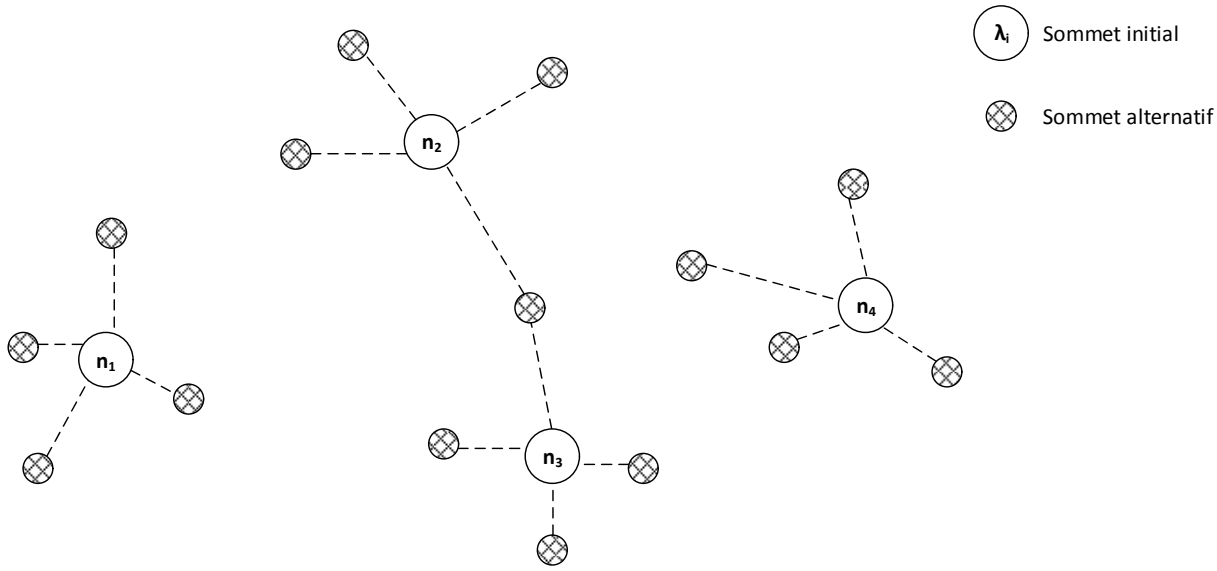


Figure 2.3 : Sommets initiaux avec sommets alternatifs associés, n_2 et n_3 ayant un sommet alternatif commun

2.2 Proposition d'un modèle linéaire pour l'optimisation exacte

Une formalisation linéaire du DARP-PV-AN est proposée dans cette section. Ce modèle combine :

- L'utilisation de véhicules publics et privés ;
- La manipulation de sommets alternatifs pour chaque client utilisant un véhicule privé ;
- L'emploi de sommets communs à plusieurs clients (*meeting nodes*).

2.2.1 Données du modèle

N ensemble des sommets initiaux, $N = N^P \cup N^D \cup \{0\} \cup \{2n+1\}$

N^P ensemble des sommets de *pickup* initiaux, $N^P \subset N$

N^D ensemble des sommets de *delivery* initiaux, $N^D \subset N$

N_i^S ensemble des sommets alternatifs liés au sommet initial i, $i \in N$

N_i^T ensemble des sommets pour i, contenant le sommet initial i et ses sommets alternatifs N_i^S, i.e. $N_i^T = N_i^S \cup \{i\}$, $i \in N$

N^T ensemble de tous les sommets initiaux et alternatifs, i.e. $N^T = (\bigcup_{i \in N} N_i^S) \cup N$

K ensemble de véhicules publics

K' ensemble de véhicules privés

K^T ensemble des véhicules, $K \cup K' = K^T$

n nombre de clients

d_i durée du service au sommet i, $i \in N^T$

$[e_i; l_i]$ fenêtre de temps au sommet i, $i \in N^T$

q_i demande du client au sommet i, $i \in N$

RT_i *riding time* maximum du client i, $i \in \{1, \dots, n\}$

$c_{i,j}^k$ durée entre le sommet i et le sommet j pour le véhicule k, $i, j \in N, k \in K^T$

g_k position initiale du sommet pour le véhicule k, $k \in K^T$

Q_k capacité maximum du véhicule k, $k \in K^T$

T durée maximale d'une tournée

M grand nombre positif (*big M*)

2.2.2 Définition des variables

Le modèle utilise 6 ensembles de variables continues ($A_i, B_i, D_i, A_{dpt}^k, D_{dpt}^k$ et v_i^k) et deux ensembles de variables binaires ($x_{i,j}^k$, $y_{l,m}$). Les variables continues sont liées aux dates d'arrivées des véhicules A_i et à leurs dates de départs D_i ainsi qu'aux dates de débuts de services B_i et à la charge des véhicules v_i^k. Les dates de départs du dépôt et les dates de retours sont aussi continues. Les variables de décision $x_{i,j}^k$ et $y_{l,m}$ sont responsables de la sélection des arcs pour les différentes tournées.

A_i date d'arrivée d'un véhicule au sommet i, $i \in N^P \cup N^D$, $A_i \geq 0$

B_i date de début de service au sommet i, $i \in N^P \cup N^D$, $B_i \geq 0$

D_i date de départ d'un véhicule du sommet i, $i \in N^P \cup N^D$, $D_i \geq 0$

A_{dpt}^k date d'arrivée d'un véhicule k au sommet final, $k \in K^T$, $A_{dpt}^k \geq 0$ (le sommet final est le dépôt pour un véhicule public et le sommet $g_k + n$ pour un véhicule privé)

D_{dpt}^k date de départ d'un véhicule k du sommet initial, $k \in K^T$, $D_{dpt}^k \geq 0$ (le sommet initial est le dépôt pour un véhicule public et le sommet g_k pour un véhicule privé)

v_i^k chargement du véhicule k lorsqu'il arrive au sommet i, $i \in N, k \in K^T$, $v_i^k \geq 0$

$$x_{i,j}^k = \begin{cases} 1 \text{ si un arc est utilisé entre } N_i^T \text{ et } N_j^T \text{ par le véhicule } k, i, j \in N, k \in K^T \\ 0 \text{ sinon} \end{cases}$$

$$y_{l,m} = \begin{cases} 1 \text{ si l'arc } (l, m) \text{ est utilisé}, l, m \in N^T \\ 0 \text{ sinon} \end{cases}$$

2.2.3 Contraintes et fonction objectif

L'ensemble des contraintes peut être partitionné en différents sous-ensembles : chacun représente un type de contraintes.

a) Contraintes de temps

Le premier ensemble de contraintes assure que les dates d'arrivées, de débuts de services et de départs respectent les fenêtres de temps du sommet correspondant. Les contraintes de précédence sont également prises en compte.

La contrainte (1) force la date de service B_i du sommet i à commencer dans la fenêtre de temps $[e_\alpha, l_\alpha]$ de son sommet visité α :

$$\sum_{\alpha \in N_i^S} e_\alpha \sum_{j,k} x_{\alpha,j}^k \leq B_i \leq \sum_{\alpha \in N_i^S} l_\alpha \sum_{j,k} x_{\alpha,j}^k, \forall i \in N^P \cup N^D \quad (1)$$

La contrainte (2) force la date de service B_i à commencer après la date d'arrivée A_i :

$$B_i \geq A_i, \forall i \in N^P \cup N^D \quad (2)$$

La contrainte (3) force le véhicule k à quitter le dépôt après son ouverture e_0 et à revenir avant sa fermeture l_0. Il oblige également la date de départ du dépôt à être avant la date de retour. Les véhicules privés commencent et finissent leur tournée au dépôt, mais leur durée de trajet depuis/vers ce dernier est égale à 0 :

$$e_0 \leq D_{dpt}^k \leq A_{dpt}^k \leq l_0, \forall k \in K^T \quad (3)$$

La contrainte (4) force la date de départ D_i du sommet i à être après la date de début de service B_i plus la durée du service d_α, le tout dépendant du sommet α visité :

$$D_i \geq B_i + \sum_{\alpha \in N_i^S} d_\alpha \sum_{j,k} x_{\alpha,j}^k, \forall i \in N^T \quad (4)$$

La contrainte (5) force la date de début de service B_{i+n} d'un sommet de type *delivery* à ne pas commencer avant la date de départ de son sommet de *pickup* associé D_i plus le temps de trajet entre eux (par exemple la durée la plus courte possible entre le *pickup* et son *delivery* associé). Dans le DARP-PV-AN, chaque sommet initial de *pickup* et *delivery* a un ensemble de sommets alternatifs associé. $\gamma_i = \min c_{l,m}^k, \forall l \in N_i^T, \forall m \in N_{i+n}^T, \forall k \in K^T$ est défini comme l'arc (l, m) de longueur minimum entre le *pickup* de i et son *delivery* associé avec le véhicule k. Il est ainsi possible de poser :

$$B_{i+n} \geq D_i + \gamma_i, \forall i \in N^P \quad (5)$$

La contrainte (6) traite les dates d'arrivée. A_j est définie après la date de départ D_i plus l'arc de taille $c_{l,m}$ si (l, m) est utilisé, avec $l \in N_i^T$ et $m \in N_j^T$. La technique du *big M* est utilisée ici : si l'arc (l, m) est utilisé, alors $y_{l,m} = 1$ et $A_j = D_i + c_{l,m}$. Sinon, A_j n'est pas contraint par D_i :

$$\left(D_i + c_{l,m}\right) + \left(y_{l,m} - 1\right) \times M \leq A_j \leq \left(D_i + c_{l,m}^k\right) - \left(y_{l,m} - 1\right) \times M \quad (6)$$
$$\forall i, j \in N^P \cup N^D, \forall k \in K^T$$

Les contraintes (7) et (8) traitent les dates d'arrivées au dépôt et depuis le dépôt de la même manière que la contrainte (6) :

$$\left(D_i + c_{l,m}^k\right) + \left(y_{l,m} - 1\right) \times M \leq A_{dpt}^k \leq \left(D_i + c_{l,m}^k\right) - \left(y_{l,m} - 1\right) \times M \quad (7)$$

$$\forall i \in N^P \cup N^D, \forall\, k \in K^T, \forall l \in N_i^T, \forall\, m \in N_0^T$$

$$\left(D_{dpt}^k + c_{l,m}^k\right) + \left(y_{l,m} - 1\right) \times M \leq A_i \leq \left(D_{dpt}^k + c_{l,m}^k\right) - \left(y_{l,m} - 1\right) \times M \quad (8)$$

$$\forall i \in N^P \cup N^D, \forall\, k \in K^T, \forall\, l \in N_0^T, \forall\, m \in N_i^T$$

La contrainte (9) garantit que la durée totale du voyage est limitée par la borne supérieure T :

$$A_{dpt}^k - D_{dpt}^k \leq T, \forall k \in K \quad (9)$$

La contrainte (10) limite le *riding time* maximum du client :

$$B_{i+n} - D_i \leq RT_i, \forall i \in \{1, \ldots, n\} \quad (10)$$

b) Contraintes de capacité

Le deuxième ensemble de contraintes garantit que la charge des véhicules ne dépasse pas leur capacité au niveau de chaque sommet des tournées.

La contrainte (11) est basée sur la formulation MTZ (Miller, Tucker et Zemlin) où la charge du véhicule est mise à jour à chaque sommet i. Il n'est pas nécessaire de définir ces contraintes par une égalité car la charge du véhicule sera automatiquement définie pour correspondre au nombre de place restantes lors de l'optimisation :

$$v_j^k \geq \left(v_i^k + q_i\right) + \left(x_{i,j}^k - 1\right) \times M, \forall i, j \in N, \forall k \in K^T \quad (11)$$

La contrainte (12) force les véhicules à être vides lorsqu'ils sont situés sur le dépôt :

$$v_0^k = 0, \forall k \in K^T \quad (12)$$

Les contraintes (13) et (14) obligent la charge d'un véhicule k à être toujours positive et à ne pas excéder sa capacité limite Q_k :

$$v_i^k \geq 0, \forall i \in N^P \cup N^D, \forall k \in K^T \quad (13)$$

$$v_i^k \leq Q_k, \forall i \in N^P \cup N^D, \forall\, k \in K^T \quad (14)$$

c) Contraintes de flot

Le troisième ensemble définit les contraintes de flot que doit respecter chaque véhicule.

La contrainte (15) garantit qu'un véhicule privé k ne peut pas arriver à son sommet de départ g_k depuis un sommet différent du dépôt, puisque l'arc « artificiel » $(0, g_k)$ est obligatoirement utilisé pour commencer la tournée privée du véhicule k. Ainsi, le véhicule k commence "virtuellement" à partir du dépôt avec un coût de déplacement nul ; tous les arcs entre le dépôt et un sommet différent de g_k sont également interdits pour le véhicule k. Ceci est garanti par la contrainte (16). Les deux contraintes impliquent qu'un véhicule privé k utilisera l'arc $(0, g_k)$ de coût 0 entre le dépôt et sa vraie position de départ si k est utilisé :

$$x_{i,g_k}^k = 0, \forall i \in N^P \cup N^D, \forall\, k \in K' \quad (15)$$

$$x_{0,i}^{k} = 0, \forall i \in (N^P \cup N^D) \backslash \{g_k\}, \forall k \in K' \quad (16)$$

Les contraintes (17) et (18) forcent les véhicules privés à terminer leur tournée sur leur sommet *delivery* correspondant, puis "virtuellement" sur le dépôt, de la même manière que les contraintes (15) et (16) :

$$x_{g_k+n,i}^{k} = 0, \forall i \in N^P \cup N^D, \forall k \in K' \quad (17)$$

$$x_{i,0}^{k} = 0, \forall i \in (N^P \cup N^D) \backslash \{g_k + n\}, \forall k \in K' \quad (18)$$

La contrainte (19) interdit à un véhicule de parcourir des arcs ayant la même origine et destination :

$$x_{i,i}^{k} = 0, \forall i \in N^P \cup N^D, \forall\, k \in K^T \quad (19)$$

La contrainte (20) assure qu'un arc (i, j) est utilisé par un seul véhicule au maximum :

$$\sum_{k \in K^T} x_{i,j}^{k} \leq 1, \forall i, j \in N^P \cup N^D \quad (20)$$

Les contraintes (21), (22) et (23) forcent un et un seul arc entrant et sortant à être utilisés $\forall N_i^T$:

$$\sum_{i \in N} \sum_{k \in K^T} x_{i,j}^{k} \leq 1\,, \forall j \in N^P \cup N^D \quad (21)$$

$$\sum_{j \in N} \sum_{k \in K^T} x_{i,j}^{k} \leq 1\,, \forall i \in N^P \cup N^D \quad (22)$$

$$\sum_{j \in N} x_{i,j}^{k} = \sum_{j \in N} x_{j,i}^{k}, \forall i \in N, \forall k \in K^T \quad (23)$$

La contrainte (24) oblige les sommets de *pickup* et *delivery* d'un client i à être desservis par le même véhicule :

$$\sum_{i \in N} x_{i,j}^{k} = \sum_{i \in N} x_{i,j+n}^{k}, \forall j \in N^P \cup N^D, \forall k \in K^T \quad (24)$$

La contrainte (25) force les véhicules à être utilisés au maximum une seule fois :

$$\sum_{i \in N^P \cup N^D} x_{0,i}^{k} \leq 1, \forall k \in K^T \quad (25)$$

La contrainte (26) force l'utilisation d'un seul et unique arc de N_i^T à N_j^T si ces derniers sont connectés :

$$\sum_{l \in N_i^T} \sum_{m \in N_j^T} y_{l,m} = \sum_{k \in K^T} x_{i,j}^{k}, \forall i, j \in N^P \cup N^D \quad (26)$$

Les contraintes (27) et (28) assurent que le sommet initial d'un client i est utilisé quand i est pris en charge par un véhicule public (27). En revanche, si un véhicule privé visite le client i, un des sommets alternatifs de l'ensemble N_i^S est utilisé (28). Cela renforce la vie privée des utilisateurs :

$$y_{i,j} = \sum_{k \in K} x_{i,j}^{k}, \forall i, j \in N \quad (27)$$

$$\sum_{l \in N_i^S} \sum_{m \in N_j^S} y_{l,m} = \sum_{k \in K'} x_{i,j}^{k}, \forall i, j \in N \quad (28)$$

La contrainte (29) interdit les liens entre les sommets alternatifs d'un même client :

$$\sum_{l\in N_i^T}\sum_{m\in N_i^T} y_{l,m} = 0, \forall i \in N \quad (29)$$

d) Contraintes de réduction de graphe

Les contraintes suivantes correspondent à des coupes pour l'arbre de recherche réduisant ainsi les temps de calcul.

Les contraintes (30) et (31) interdisent les connexions depuis un sommet de *pickup* vers le sommet du dépôt, et depuis le sommet du dépôt vers un sommet de *delivery* :

$$x_{i,0}^k = 0, \forall i \in N^P, \forall k \in K^T \quad (30)$$

$$x_{0,i}^k = 0, \forall i \in N^D, \forall k \in K^T \quad (31)$$

La contrainte (32) interdit la connexion depuis un sommet *delivery* vers un sommet *pickup* du même client :

$$x_{i+n,i}^k = 0, \forall i \in N^P, \forall k \in K^T \quad (32)$$

Pour les tournées privées, les contraintes (33) et (34) interdisent au second sommet d'être un sommet de *delivery* et l'avant dernier sommet d'être un *pickup* (à l'exception des sommets *pickup* et *delivery* du client possédant le véhicule privé) :

$$x_{i,g_k+n}^k = 0, \forall k \in K', \forall i \in N^P \backslash \{g_k\} \quad (33)$$

$$x_{g_k,i}^k = 0, \forall k \in K', \forall i \in N^D \backslash \{g_k\} \quad (34)$$

L'ensemble des variables sont positives et les variables binaires doivent être associées à la valeur 0 ou 1.

e) Fonction objectif

L'objectif du modèle est de minimiser la somme des distances parcourues par les véhicules lors de leurs tournées :

$$\min \sum_{i\in N^T}\sum_{j\in N^T}\sum_{k\in K^T} c_{i,j}^k \times x_{i,j}^k \quad (35)$$

Les contraintes (5), (15), (16), (17), (18), (28), (29), (33) et (34) sont dédiées au DARP-PV-AN et permettent d'aborder le problème des véhicules privés et de la vie privée des clients.

Grâce à ce modèle, différentes versions du problème de transport à la demande peuvent être traitées :

- Dans le cadre ou $K' = 0$, alors $R' = \emptyset$, il n'y a pas de véhicules privés. Le problème se réduit donc au DARP classique, pouvant être résolu par le modèle proposé ;
- Si $K = 0$ et $R' \neq \emptyset$, alors il n'y a pas de flotte de véhicules publics, seulement des véhicules privés. Dans ce cas, si $N_i^S = i$ alors l'ensemble des sommets alternatifs se réduit au sommet initial et le modèle résout un problème de covoiturage (*Ridesharing Problem*) avec fenêtre de temps ;
- Enfin, si $K \neq 0$ et $K' \neq 0$, le problème étudié est celui proposé dans le cadre de cette thèse, le DARP-PV-AN.

2.3 Proposition d'une métaheuristique de type ELS

La métaheuristique de type « recherche locale évolutive » (*Evolutionary Local Search* – ELS) a été proposée pour la première fois par (Wolf et Merz, 2007). Elle améliore les méthodes de type « recherche locale itérative » (*Iterated Local Search* – ILS) proposées par (Lourenço et al., 2003) et a été appliquée avec succès au problème de tournées de véhicules (*Vehicle Routing Problem* – VRP) par (Prins, 2009). À chaque itération de l'ELS, plusieurs copies de la solution courante sont effectuées. Chaque copie est modifiée par un ou plusieurs opérateurs de mutation avant d'être améliorée par différentes méthodes de recherche locale : la meilleure obtenue est conservée en tant que nouvelle solution courante pour la prochaine itération. La logique de l'algorithme ELS est d'examiner le voisinage de l'optimum local actuel avant de le quitter. Il est souvent associé à un mécanisme d'exploration plus large, comme dans le GRASP (*Greedy Randomized Adaptive Search Procedure*) par exemple. Il peut même être intégré dans un simple démarrage multiple, afin de gérer la diversité lors de l'exploration de l'espace des solutions. Un schéma global de l'ELS est proposé en *Figure 2.4*.

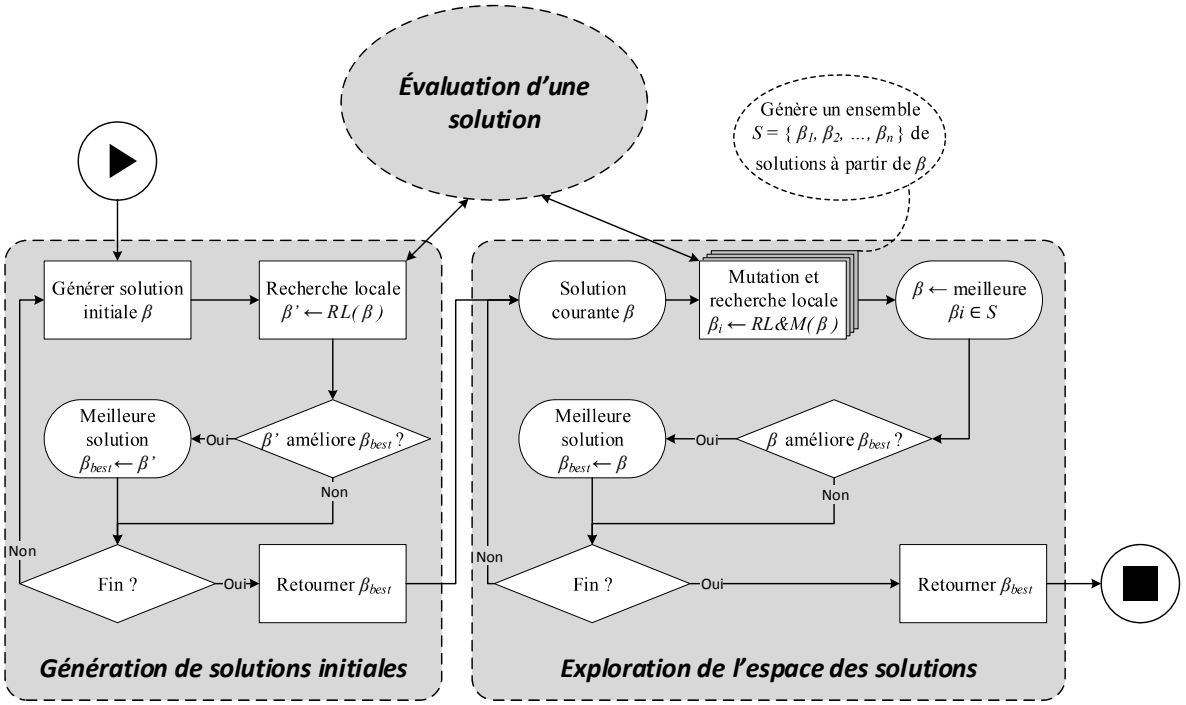


Figure 2.4 : Schéma global de l'algorithme ELS

La méthode de résolution proposée reprend les principes proposés par (Chassaing et al., 2016). La prise en compte de véhicules privés ainsi que le respect de la vie privée des clients par l'utilisation de sommets alternatifs ont été intégrés. De plus, deux nouveaux opérateurs de recherche locale dédiés aux tournées privées ont été développés et ajoutés.

Le schéma de l'algorithme ELS proposé est décrit dans l'*Algorithme 1*. L'algorithme commence par initialiser les différentes probabilités d'appels des opérateurs (ligne 1) puis poursuit par la création d'une solution initiale avec une heuristique (ligne 4). Cette solution initiale est ensuite améliorée par différentes méthodes de recherche locale (ligne 5). Ensuite, chacune des *ne* itérations de l'ELS consiste à :

- Créer en ensemble de solutions correspondant au voisinage de la solution courante par copies et mutations (ligne 11) ;

- Améliorer les solutions ainsi créées en utilisant des méthodes de recherche locale (ligne 12) ;
- Conserver la meilleure solution créée par les deux modifications ci-dessus et l'affecter comme nouvelle solution courante pour la prochaine itération (ligne 14) ;
- Mettre à jour les probabilités d'activations des différentes méthodes utilisées dans la recherche locale (ligne 16).

Algorithme 1 ***ELS***

Input parameters

nd : Number of neighbors
ne : Number of ELS iterations
nr : Number of iterations with same probabilities P

Output variable

S : Solution

Begin

```
1   initialization of P: P[i] = 0.25, ∀i = 1..8 violationS := false
2
3   // Step 1. Creation of the initial solution
4   s := randomized_constructive_heuristic()
5   s := local_search(s,P)
6
7   // Step 2.Improvement by ELS
8   for i := 1 to ne do
9   |   s' := s
10  |   for k := 1 to nd do
11  |   |   s" := mutation(s')
12  |   |   s" := local_search(s", P)
13  |   end for
14  |   s := best of all s"
15  |   if (i mod nr)=0 then
16  |   |   update(P)
17  |   end if
18  end for
19
20  return s
End
```

La procédure de recherche locale est appelée aux lignes 5 et 12 afin d'améliorer la solution (s et s"). Elle utilise plusieurs opérateurs décrits par (Braekers et al., 2014; Lin et Kernighan, 1973; Masson et al., 2014; Potvin et Rousseau, 1993). Ces opérateurs s'appuient sur des mouvements de base et correspondent à différentes manières d'explorer localement l'espace des solutions. Une probabilité d'activation P_i est associée à chaque opérateur i et est mise à jour toutes les nr itérations de l'ELS.

Les principales caractéristiques de la méthode proposée sont les suivantes :

- Une représentation indirecte des solutions sous forme d'une séquence de requêtes et une fonction de décodage permettant la création d'une solution réalisable à partir de cette séquence ;
- Une heuristique constructive pour générer des solutions initiales de bonne qualité, en fonction des étapes suivantes :
    - Pour chaque couple de clients, une vérification sur l'ordre est effectuée : si un

client doit être servi avant un autre, cela est modélisé par l'ajout d'une contrainte de précédence dans un graphe de précédence ;
    - Le graphe de précédence est ensuite partitionné par niveau ;
    - Une fois le graphe de partition créé, ce dernier est utilisé pour générer un ordre d'insertion des différents clients. Cet ordre, couplé à une heuristique d'insertion, permet de générer des solutions initiales.
- Une méthode supervisant la recherche locale et reposant sur des probabilités dynamiques. Une probabilité est associée à chaque méthode de recherche : à chaque itération, un sous-ensemble de ces méthodes est sélectionné en fonction de ces probabilités. Ces dernières sont ensuite mises à jour en fonction de leur réussite (ou non) à améliorer la solution courante ;
- Une procédure efficace pour évaluer le coût d'une tournée, prenant en compte le plus court chemin entre les différents sommets initiaux et alternatifs et minimisant de manière itérative trois critères (*Total Duration*, *Total Riding Time* et *Total Waiting Time*).

2.3.1 Évaluation d'une solution

La méthode d'évaluation d'une solution consiste à évaluer indépendamment chaque tournée. Cette méthode prend en compte un ordre de visite des clients ainsi que le type de véhicule (public ou privé). Le respect de la vie privée des utilisateurs par l'utilisation de sommets alternatifs du sommet initial, si le client voyage dans un véhicule privé, est également considéré. L'algorithme de base permet l'évaluation d'une tournée et a été introduit par (Firat et Woeginger, 2011) : il propose une méthode de complexité linéaire par rapport au nombre de clients. Cette méthode améliore l'algorithme original de complexité quadratique proposé par (Cordeau et Laporte, 2003) mais contrairement à ce dernier, ne minimise ni la durée totale de la tournée (*Total Duration*) ni le temps de trajet minimal des clients (*Total Riding Time*). La proposition d'une méthode d'évaluation d'une tournée du DARP-PV-AN consiste en deux étapes principales :

- Étape 1 : permettre l'évaluation d'une tournée privée en ajoutant des arcs à coût nul partant du dépôt vers le sommet de *pickup* correspondant au véhicule privé, ainsi que du sommet de *delivery* correspondant au véhicule privé jusqu'au dépôt ;
- Étape 2 : quand un véhicule visite le sommet d'un client (*pickup* ou *delivery*), l'algorithme doit choisir où le client sera pris en charge ou déposé. Si le transport est effectué par un véhicule public, alors le sommet initial du client sera utilisé. En revanche, si le client est transporté par un véhicule privé, un des sommets alternatifs doit être choisi par l'algorithme.

Les modifications apportées à l'algorithme classique lors de la première étape permettent une fonction d'évaluation commune pour les tournées effectuées par des véhicules publics ou des véhicules privés. Cette fonctionnalité est permise grâce à l'ajout d'arcs à coût nul lors de l'évaluation d'une tournée privée. Comme souligné sur la *Figure 2.5*, les arcs à coût nul $c_{DPT,1^+}$ et $c_{1^-,DPT}$ sont ajoutés depuis le dépôt vers le *pickup* du client conduisant le véhicule privé, et depuis le *delivery* vers le dépôt (1). Ensuite, la tournée est évaluée comme une tournée publique (2). Une fois l'évaluation terminée, les arcs supplémentaires sont supprimés (3). Cette modification permet à l'algorithme d'évaluer les déplacements publics et privés en temps linéaire de façon identique.

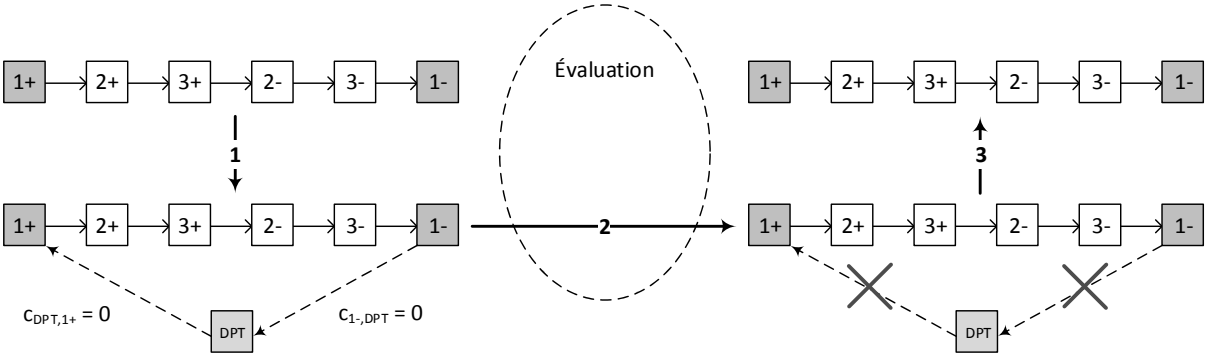


Figure 2.5 : Évaluation d'une tournée effectuée par un véhicule privé en utilisant des arcs à coût nul

Lors de la seconde étape, la vie privée d'un utilisateur est prise en compte par l'utilisation de sommets alternatifs. Chaque client i est associé à un sommet initial de *pickup*/*delivery*, i^+/i^- et à un ensemble de sommets alternatifs de *pickup*/*delivery*, $N_{i^+}^S/N_{i^-}^S$. Ainsi, un sommet de pickup et un sommet de delivery doivent être sélectionnés en fonction du type de véhicule utilisé pour le transporter (public ou privé). Pour un véhicule public, les sommets de *pickup* et *delivery* initiaux i^+/i^- sont utilisés. Dans le cas d'un véhicule privé, il faut choisir un sommet de *pickup* $\alpha^+ \in N_{i^+}^S$ et un sommet de *delivery* $\alpha^- \in N_{i^-}^S$. Cette sélection est effectuée de manière optimale en construisant un graphe par niveau : chaque niveau est constitué d'un ensemble de sommets alternatifs sans aucune arête entre eux et est relié au niveau précédent et suivant sous la forme d'un graphe biparti complet. Chaque niveau correspond soit aux sommets alternatifs correspondant au *pickup*/*delivery* d'un client, soit au départ/retour au dépôt avec un coût nul. Une fois le graphe construit, un plus court chemin peut être calculé, en utilisant par exemple l'algorithme de (Dijkstra, 1959) afin de minimiser la distance parcourue par le véhicule privé.

L'exemple de la *Figure 2.6* illustre la sélection de sommets alternatifs lorsqu'un véhicule privé est utilisé en exploitant un graphe par niveau. Ici, le véhicule du client 1 est utilisé car la tournée commence par le sommet de *pickup* 1^+ et termine par le sommet de *delivery* 1^-. Des arcs de coût 0 sont ajoutés entre le dépôt et $N_{i^+}^S/N_{i^-}^S$. Pour cet exemple, le véhicule effectue également la demande du client 2 lors de son trajet. Étant donné qu'un véhicule privé effectue

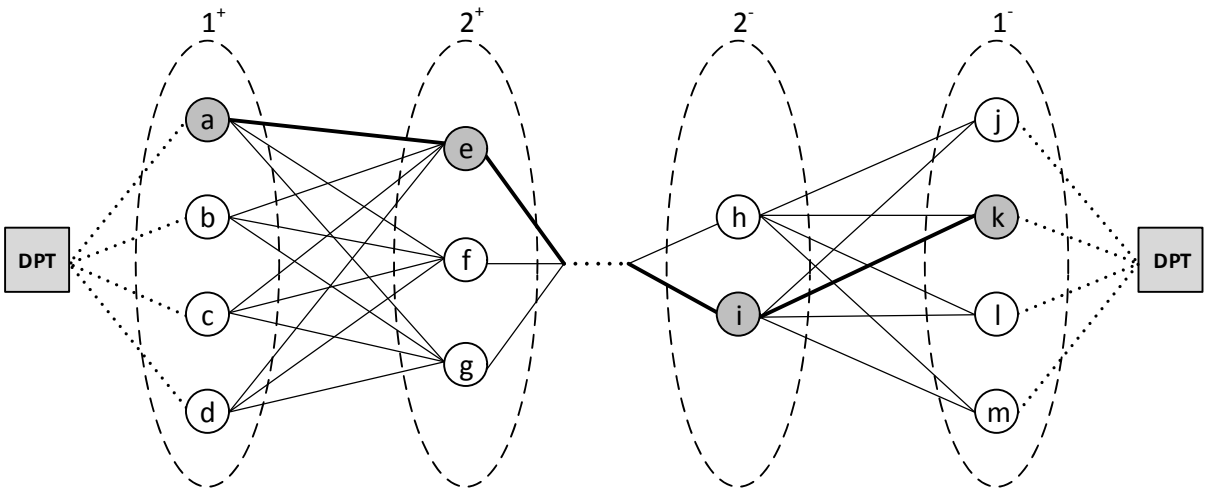


Figure 2.6 : Sélection de sommets alternatifs lors de l'évaluation d'une tournée privée

le trajet, les sommets initiaux ne peuvent pas être utilisés afin de préserver la confidentialité du client. Ainsi, un sommet alternatif doit être sélectionné pour chaque sommet de *pickup* et *delivery* : ici, les sommets alternatifs $\{a, e, \dots, i, k\}$ sont visités puisqu'ils forment le plus court chemin.

La méthode d'évaluation est présentée dans l'*Algorithme 2*. Il prend en entrée une solution potentielle S représentant un ensemble de tournées. Pour chaque tournée $S[i]$, l'algorithme appelle une procédure $evaluation\_T$ (ligne 5) permettant d'obtenir le coût de la tournée et de vérifier sa validité. La somme des coûts de chaque tournée est effectuée (ligne 6), et si une seule tournée n'est pas valide, alors $S\_violation$ est assignée à $true$ (ligne 7). L'algorithme retourne la somme des coûts des tournées ainsi que la validité ou non de la solution S.

Algorithme 2 $\boldsymbol{Evaluation\_S}$

Input parameter

$S[\,]$: Set of trips. Each trip contains variables such departure time, charge, etc. that will be modified during the algorithm

Output variables

S_cost : Sum of the cost of all trips in the solution

$S\_violation$: Boolean that determine if the solution is valid ($false$) or violate at least one constraint ($true$)

Begin

1 S_cost := 0
2 $S\_violation$:= false
3
4 **for** i := 1 **to** $|S|$ **do**
5 | $\{T\_cost,\ T\_violation\}$:= evaluation_T(S[i])
6 | S_cost += T_cost
7 | $S\_violation$ |= $T\_violation$
8 **end for**
9
10 **return** $\{S\_cost, S\_violation\}$

End

L'évaluation d'une tournée $trip$ est décrite dans l'*Algorithme 3* et est effectuée en trois étapes :

- Si le véhicule effectuant la tournée est privé, ajouter les arcs à coût zéro (ligne 2) pour simuler un départ et un retour au dépôt n'ayant aucun effet sur le coût de la tournée ou les valeurs affectées aux variables de décision $(A_i, B_i, v_i, \dots)$;
- Si le véhicule effectuant la tournée est un véhicule privé, il faut alors sélectionner les sommets alternatifs sur lesquels les clients vont être pris/déposés (ligne 3) ;
- Les modifications sur la tournée ayant été effectuées, l'appel de la méthode $evaluate\_T$ permet l'évaluation de la tournée en temps linéaire. Cette procédure est une implémentation de l'algorithme initialement proposé par (Firat et Woeginger, 2011) et utilisé par exemple par (Chassaing et al., 2016).

Algorithme 3 $\boldsymbol{Evaluation\_T}$

Input parameter

$trip$: The trip that need to be evaluated. It contains several variables that will be modified during the algorithm

Output variables

T_cost : Cost of the trip

$T\_violation$: Boolean that determine if the trip is valid ($false$) or violate at least one constraint ($true$)

Begin

```
1   if trip is Private do
2   |  add_zero_cost_arcs(trip)
3   |  compute_alternative_node(trip)
4   end if
5
6   return evaluate_T(trip)
End
```

2.3.2 Mutation

La mutation est une fonction initialement introduite dans le contexte des algorithmes génétiques. Elle est utilisée dans l'ELS et consiste à effectuer de manière aléatoire une petite modification de la solution courante S afin d'obtenir une nouvelle solution S' partageant des similitudes avec S. Deux opérateurs de mutation ont été étudiés :

- Le premier opérateur supprime une requête de transport d'une tournée et l'insère dans une nouvelle tournée, différente de la première ;
- Le second opérateur consiste en un mouvement de « *cut and paste* ».

Ces deux opérateurs n'ont pas la même probabilité d'être utilisés. La première mutation génère un voisin proche de la solution courante avec une probabilité de 0,7. La deuxième mutation génère un voisin plus différent mais est plus coûteuse en terme de calcul : la probabilité de l'utiliser est de 0,3. Ces probabilités définissent un équilibre entre les temps de calculs et la diversification.

Le second opérateur fonctionne de la manière suivante : on choisit de manière aléatoire dans la solution courante, deux tournées λ_1 et λ_2 de longueur respective n_{λ_1} et n_{λ_2}. Deux positions p_1 et p_2 sont choisies aléatoirement dans λ_1 et λ_2, telles que $1 \leq p_k \leq n_{\lambda_k}, k \in \{1,2\}$. Toutes les requêtes de transport pour lesquelles la position p_i est entre leur *pickup* et leur *delivery*, sont marqués *true* dans les tournées λ_1 et λ_2. Tous les sommets de λ_1 situés avant p_1 associés à des requêtes de transport non marquées, sont insérés de manière itérative dans la nouvelle tournée t_1. Ensuite, tous les sommets de λ_2 situés après p_2 et associés aux requêtes de transport non marquées, sont insérés de manière itérative dans la nouvelle tournée t_1. Les rôles de λ_1 et λ_2 sont ensuite échangés pour générer la tournée t_2. Chaque requête de transport marquée *true* est ensuite insérée à la meilleure position possible dans une des deux tournées choisies aléatoirement. Si une tournée comporte un véhicule privé, les sommets représentant son propriétaire sont insérés dans une des deux tournées créées afin de continuer à utiliser son propre véhicule. Ainsi, le nombre de tournées privées et publiques est identique avant et après l'exécution de cet opérateur de mutation. Un exemple du déroulement de cette mutation est donné ci-après.

Considérons les tournées λ_1 (privée) et λ_2 (publique) de la *Figure 2.7* avec les positions de coupe $p_1 = 3$ et $p_2 = 3$. Les sommets $5^+, 3^+, 5^-, 3^-$ sont marqués *true* car leur *pickup* et leur *delivery* sont séparés par une coupe. Les sommets 1^+ et 1^- sont mis de côté puisqu'ils correspondent au point de départ et d'arrivée du client possédant le véhicule privé. Enfin les sommets correspondant au dépôt sont également mis de côté.

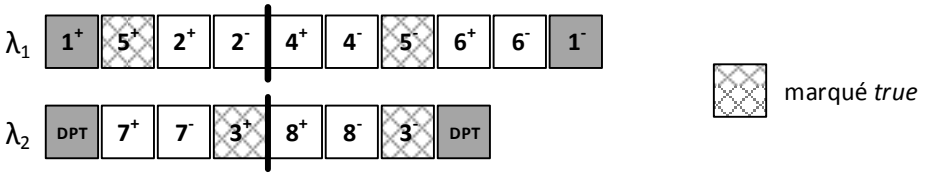


Figure 2.7 : Tournées initiales sélectionnées λ_1 et λ_2 et positions des coupes

Dans la *Figure 2.8*, la tournée t_1 est créée avec le début de la tournée λ_1 et la fin de la tournée λ_2 : les sommets $2^+, 2^-, 8^+, 8^-$ sont insérés dans t_1 car ils ne sont pas marqués. En revanche, les requêtes 5 et 3 ne sont pas encore prises en compte et sont mises de côté.

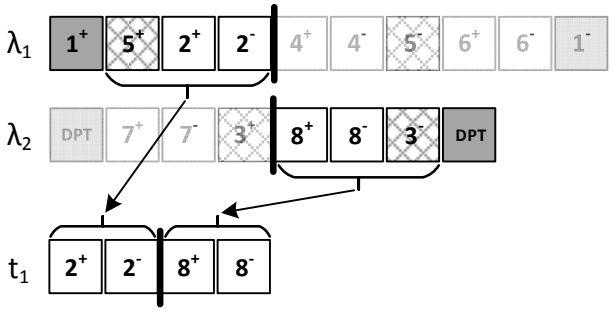


Figure 2.8 : Création de la tournée t_1

De manière similaire, la tournée t_2 est créée avec la fin de la tournée λ_1 et le début de la tournée λ_2 : les sommets $7^+, 7^-, 4^+, 4^-, 6^+, 6^-$ sont insérés dans t_2 (*Figure 2.9*).

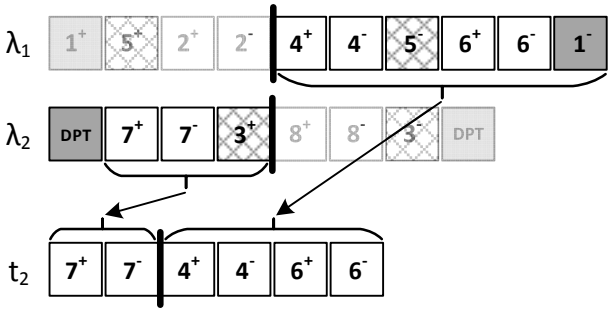


Figure 2.9 : Création de la tournée t_2

L'étape suivante consiste à insérer les sommets marqués ($3^+, 3^-, 5^+, 5^-$ sur la *Figure 2.10*). Les clients non insérés sont choisis dans un ordre aléatoire. Pour chaque client, une des deux tournées créées est choisie aléatoirement. Les sommets de *pickup* et *delivery* correspondant au client sont ensuite insérés dans la tournée choisie à la meilleure position possible.

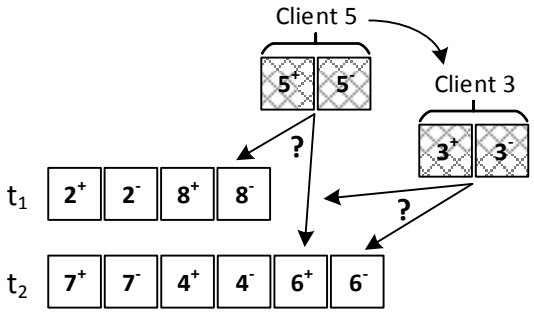


Figure 2.10 : Choix de l'ordre d'insertion des clients marqués (5 puis 3) ainsi que leur tournée

La dernière étape est illustrée sur la *Figure 2.11* : les sommets 1^+ et 1^- correspondant aux positions de départ et d'arrivée du véhicule privé doivent être insérés tout comme le départ et l'arrivée au dépôt pour la tournée publique. Les deux possibilités sont évaluées et la meilleure est gardée.

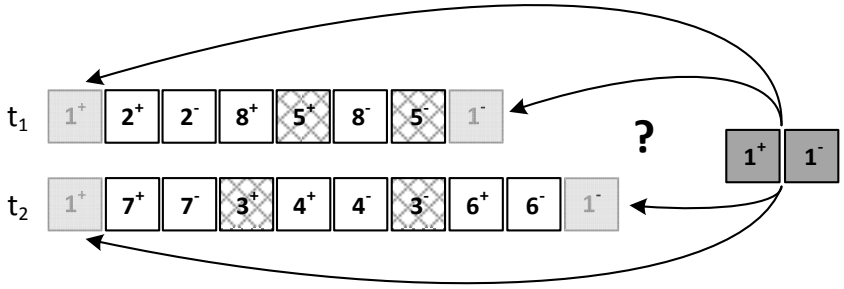


Figure 2.11 : Insertion du client 1 possédant le véhicule privé une fois les sommets marqués insérés

Une fois l'ensemble des étapes effectuées, les deux tournées forment par exemple le résultat final de la *Figure 2.12*.

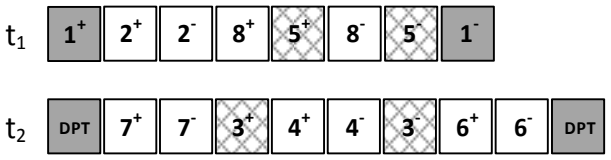


Figure 2.12 : Tournées finales

2.3.3 Recherche locale

La recherche locale est une méthode heuristique itérative qui tente de réduire le coût de la solution en explorant plusieurs voisinages de la solution actuelle. Un ensemble de six mouvements de base, avec des probabilités d'être appelées différentes qui évoluent au cours des itérations, est proposé. Ces mouvements sont les suivants :

- Un mouvement de type « 4-*Opt* » proposé par (Lin et Kernighan, 1973), où 4 arcs sont supprimés dans une même tournée et toutes les manières possibles de reconnecter les segments restants sont examinées, en gardant la meilleure solution ;
- Le mouvement « *cut and paste* » également utilisé dans la partie mutation de l'algorithme ;

- Un mouvement de suppression du pire client d'une tournée : le client responsable du plus grand détour est retiré de sa tournée puis réinséré à la meilleure position dans une tournée aléatoire ;
- Un mouvement « *relocate* » mettant en œuvre l'algorithme proposé par (Braekers et al., 2014), dans lequel une tournée est choisie aléatoirement. Chaque client dans cette tournée est successivement retiré avant d'être réinséré dans une autre tournée (sélectionnée aléatoirement) ;
- Le mouvement « 2-*Opt** » proposé par (Potvin et Rousseau, 1993), où un arc est supprimé dans deux tournées choisies aléatoirement. Les différentes manières de reconnecter les segments restants sont examinées et la meilleure solution est sélectionnée ;
- Une méthode d'échange de requêtes proposée par (Braekers et al., 2014), échangeant deux requêtes dans des tournées différentes.

Ces différents algorithmes de recherche locale sont complétés par deux nouvelles méthodes présentées dans les sections suivantes. Elles concernent la nature de la tournée, qu'elle soit publique ou privée :

- Le mouvement « *extract private trip* » créant une tournée privée à partir d'une requête traitée par un véhicule public ;
- Le mouvement « *remove private trip* » supprimant une tournée privée et insérant ses requêtes de transport dans les autres tournées.

a) Extract private trip

Dans la méthode « *extract private trip* », une tournée publique λ et les positions d'un sommet *pickup* (p_i^+) et d'un sommet *delivery* (p_i^-) d'un client i pouvant effectuer une tournée privée sont fournies en entrée de l'algorithme. Ensuite, i est retiré de λ et une nouvelle tournée privée λ' constituée du seul client i est créée. Toutes les requêtes de transport dans λ dont le *pickup* et le *delivery* se situent entre p_i^+ et p_i^- sont marquées *true*. Enfin, ces requêtes sont séquentiellement insérées dans λ' si possible.

Par exemple, considérons la tournée publique λ de la *Figure 2.13* avec le client 5 en position $p_5^+ = 2$ et $p_5^- = 7$ possédant un véhicule privé. Les sommets 1^+ et 1^- sont marqués (rayés sur la figure) puisqu'ils correspondent à une requête de transport située entre p_5^+ et p_5^-.

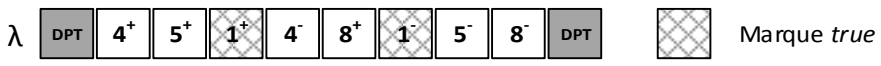


Figure 2.13 : Tournée publique avant l'appel de la méthode d'extraction de tournée privée

Ensuite, le client 5 est extrait de λ et une nouvelle tournée privée λ' est créée (*Figure 2.14*).

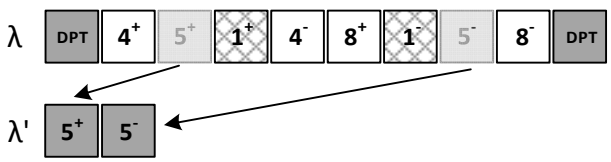


Figure 2.14 : Création de la tournée privée

Enfin, chaque client marqué *true* (rayé sur la figure) est inséré si possible dans la tournée privée (*Figure 2.15*). L'ordre d'insertion de clients dépend de l'ordre de leur *pickup* dans la tournée publique λ. Le résultat final est montré dans la *Figure 2.16*.

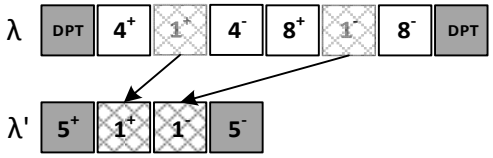


Figure 2.15 : Insertion des différentes requêtes marquées true

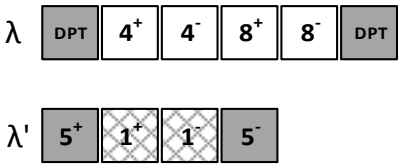


Figure 2.16 : Tournées privée et publique à la fin de la méthode « extract private turn *»*

b) Remove private trip

Ce mouvement est illustré dans l'*Algorithme 4* et consiste en la suppression d'une tournée privée et la réinsertion de ses différentes requêtes dans d'autres tournées. Elle est composée de quatre étapes principales. Tout d'abord, un tour privé doit être sélectionné dans S' (ligne 6). Si la solution ne comporte aucun tour privé, alors l'algorithme ne fait aucune modification. Ensuite, une copie de la solution courante S dans S' est effectuée afin de revenir à S si S' n'est pas valide (ligne 10). La liste des requêtes de transport des clients issue de la tournée privée supprimée est mélangée pour un ordre d'insertion aléatoire (ligne 15). Chaque requête est ensuite insérée en limitant le détour qu'elle causait précédemment (ligne 19 et 20) : la méthode « *insert_best* » essayant d'insérer la requête dans chaque tournée à la meilleure position possible, et sélectionnant la meilleure tournée. La variable « *maxDev* » permet, lors de l'insertion d'une requête dans une tournée, d'autoriser un détour maximum supérieur au précédent détour que faisait le véhicule privé. Ainsi, quelques détours plus grands pour certaines requêtes peuvent autoriser des détours beaucoup plus courts pour d'autres requêtes et avoir donc une solution avec un coût global inférieur.

2.4 Instances

Le modèle linéaire ainsi que la métaheuristique ELS ont été testés tous les deux sur trois ensembles d'instances :

- Le premier ensemble contient 20 instances modifiées à partir de celles fournies par (Ropke et al., 2007). La modification par rapport aux instances d'origine consiste à ajouter un ensemble de véhicules privés, plus précisément les clients proposant l'utilisation potentielle de leur véhicule. De plus, des sommets alternatifs liés aux sommets initiaux ont été créés et ajoutés dans chaque instance. Certains clients ont également été supprimés afin de garder les instances suffisamment petites pour la résolution par le modèle linéaire. Les tests ont montré que 8 requêtes constituaient une limite pratique pour une résolution exacte. En incluant les sommets alternatifs, on obtient des instances avec un maximum de 47 sommets ;

Algorithme 4 $remove\_private\_trip$

Input parameter

S : The solution where a private trip will be removed

Output variables

S_cost : Cost of the solution
$S\_violation$: Boolean that determine if the solution is valid ($false$) or not ($true$)

Local variables

t : Private trip that will be removed
S' : Copy of S
$clients[\,]$: Set of clients in t
$coef$: Maximum deviation coefficient
$maxDev$: Maximum deviation

Begin

1 S_cost := $S.cost$
2 $S\_violation$:= false
3 $coef$:= 1.5
4
5 **// Step 1. Selection of the private turn**
6 t := select_private_turn(S)
7
8 **if** $t \neq \emptyset$ **do**
9 | **// Step 2. Creation of a temporary solution S' that will be modified**
10 | S' := S
11 | S' := remove_trip(S', t)
12 | $clients$:= extract_client_from_trip(t)
13 |
14 | **// Step 3. Shuffle the set of client for their future insertion order**
15 | shuffle_Fisher_Yates($clients$)
16 |
17 | **// Step 4. Try to reinsert all client in trip t into other trips in S'**
18 | **for** i := 0 **to** $clients.length$ **do**
19 | | $maxDev$:= detour_of_client($clients[i]$) * $coef$
20 | | { S_cost , $S\_violation$ } := insert_best(S', $clients[i]$, $maxDev$)
21 | **end for**
22 |
23 | **if** $S\_violation$ = false **do**
24 | | S := S'
25 | **end if**
26 **end if**
27
28 **return** { S_cost , $S\_violation$ }
End

- Le second ensemble est composé de 20 instances prolongeant l'ensemble proposé par (Cordeau et Laporte, 2003). Ces instances sont dédiées à l'évaluation de la métaheuristique ELS. Comme pour le premier ensemble, ces instances ont été modifiées pour ajouter des véhicules privés et des positions de sommets alternatifs liés aux sommets initiaux. Aucun client n'a été supprimé car l'ELS est capable de gérer des instances contenant jusqu'à 120 requêtes de transport ;
- Le dernier ensemble est constitué de 35 instances plus réalistes qui ont été créées sur la base de graphes orientés générés par (Duhamel et al., 2009). Ces instances contiennent jusqu'à 60 requêtes de transport. Cet ensemble est également dédié à l'évaluation de la performance de la métaheuristique ELS.

La génération du troisième ensemble d'instances est basée sur l'utilisation de deux graphes proposés par (Duhamel et al., 2009) : les graphes 126 et 224. Ces graphes représentent de manière réaliste un potentiel réseau routier : le graphe est asymétrique (la distance d'un sommet i à un sommet j est différente de celle de j à i) et les distances utilisées sont non euclidiennes. La génération de chaque instance à partir de ces deux graphes est divisée en 4 étapes principales :

1. Choix de la position des sommets initiaux des clients dans le graphe ;
2. Choix de la position des sommets alternatifs pour chaque client ;
3. Choix de la position des véhicules privés ;
4. Choix des fenêtres de temps.

2.4.1 Position des sommets initiaux

Chaque instance générée contient entre 10 et 60 requêtes de transport. Chaque requête est constituée d'un sommet de *pickup* initial et d'un sommet de *delivery* initial. Le but de cette première étape est de choisir pour chaque requête, deux sommets initiaux dans le graphe correspondant au *pickup* et au *delivery* du client. Pour cela, la première étape consiste en la création de deux zones dans le graphe concerné : une zone résidentielle et une zone industrielle (*Figure 2.17*).

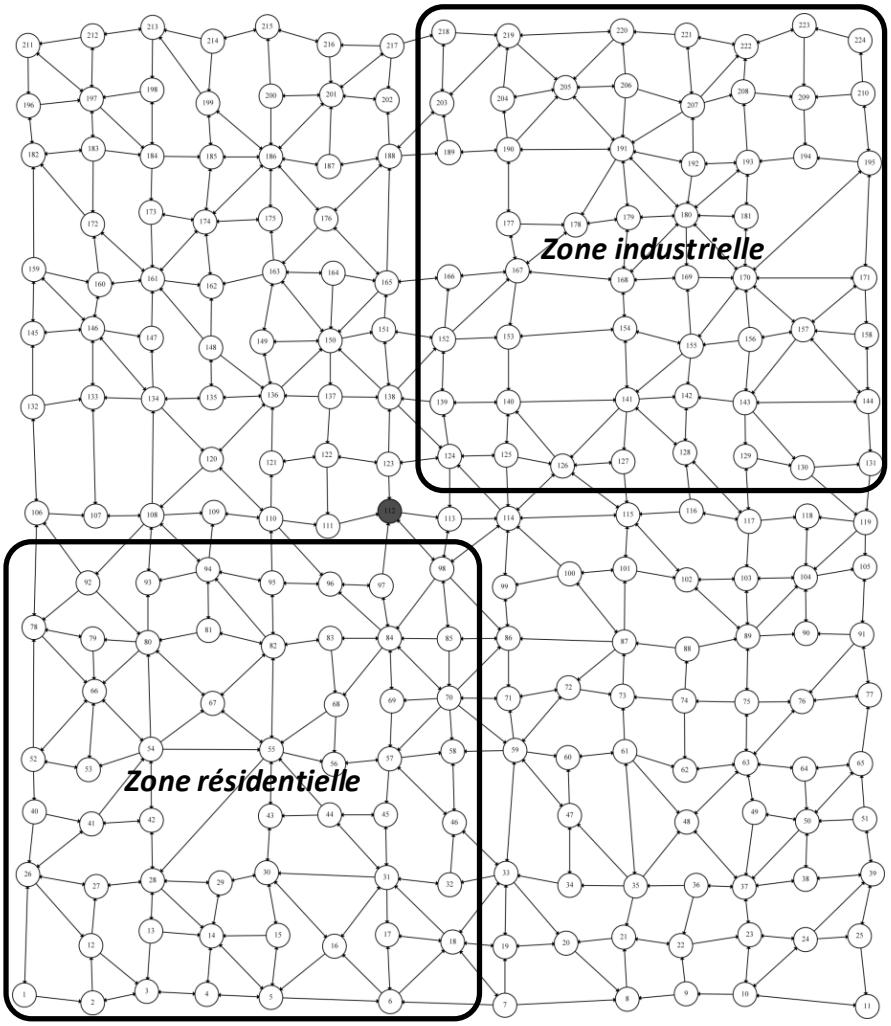


Figure 2.17 : Détermination d'une zone résidentielle et d'une zone de travail dans le graphe

Chaque requête a 70% de chance de modéliser un transport classique partant de la zone résidentielle et finissant dans la zone industrielle (ou zone de travail), et 30% de chance de représenter un trajet ne correspondant pas à un transport classique de va et vient tel qu'on le vit dans les grandes villes aux horaires de bureau. Si le trajet part de la zone résidentielle et finit dans la zone industrielle, alors un sommet initial est choisi aléatoirement dans la zone résidentielle comme sommet de *pickup* et un autre sommet initial est choisi aléatoirement dans la zone industrielle comme sommet de *delivery*. En revanche, si le trajet ne correspond pas à un déplacement classique, alors un sommet initial de *pickup* et un sommet initial de *delivery* sont choisis aléatoirement dans tout le graphe parmi les sommets non sélectionnés. Dans les deux cas précédents, lorsqu'une requête de transport aller, d'un sommet initial *A* à un sommet initial *B*, est créé, la requête retour, du sommet initial *B* vers le sommet initial *A*, est également ajoutée. Pour chaque sommet initial sélectionné, le chargement du véhicule est de 1 ou -1 suivant son type (*pickup* ou *delivery*).

2.4.2 Position des sommets alternatifs

L'étape suivante consiste au positionnement des sommets alternatifs liés aux sommets initiaux sélectionnés dans la section 2.4.1. À chaque sommet initial est lié un nombre de sommets alternatifs choisi aléatoirement entre 1 et 4. Chacun de ces sommets alternatifs est situé aléatoirement à une distance de 1 ou 2 arcs du sommet initial. Ainsi, lorsque les clients sont desservis par un véhicule privé, le lieu de rendez-vous sélectionné n'est pas à une distance trop importante du sommet initial.

Plusieurs sommets initiaux peuvent avoir certains de leurs sommets alternatifs en commun. De même, un sommet peut être à la fois le sommet initial d'un client et l'un des sommets alternatifs d'un autre client.

Un exemple est proposé sur la *Figure 2.18* avec un sommet initial et ses trois sommets alternatifs.

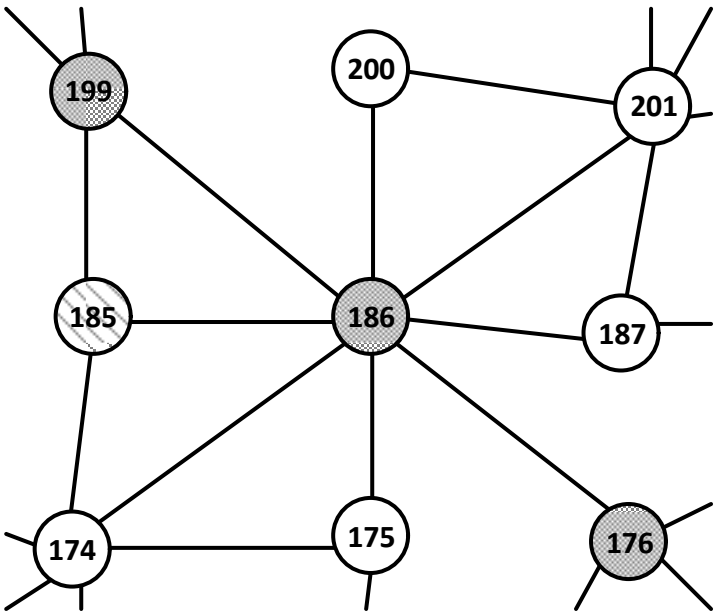


Figure 2.18 : Sommet initial (185) et ses sommets alternatifs (176,186 et 199)

2.4.3 Position des véhicules privés

Le nombre et la position des véhicules privés dépendent des clients : chacun peut mettre à disposition son véhicule personnel. Si un client décide de potentiellement utiliser son véhicule privé, alors ce dernier commencera sa tournée sur le sommet de *pickup* du client et finira sur son sommet de *delivery*. Le nombre de véhicules privés varie de 0% des clients (DARP classique) à 100% des clients. Ainsi, sur une instance de 30 clients, si 50% de ces derniers mettent à disposition leur véhicule privé, 15 clients sont choisis aléatoirement comme possédant leur propre véhicule et le mettant à disposition pour une éventuelle utilisation.

2.4.4 Fenêtres de temps

Pour fixer les fenêtres de temps, une normalisation des valeurs de chaque arc du graphe doit être effectuée. Pour cela, le plus court chemin de chaque couple de nœuds est calculé. La plus longue des distances ainsi trouvées représente une distance parcourue fixée à deux heures. Grâce à cette valeur fixée, chaque arc peut être normalisé et un temps de parcourt dépendant de leur distance leur est affecté. Ainsi, la matrice des distances entre chaque sommet d'une instance est représentée par des temps de parcours en minutes, proportionnels à la distance.

Une fenêtre de temps $[e_i; l_i]$ est associée à chaque sommet i initial (les sommets alternatifs liés au sommet initial ont la même fenêtre de temps), avec $0 \leq e_i \leq l_i \leq 1440$ ($[0; 1440]$ représente une durée d'une journée en minute). Afin de simuler des demandes réalistes, à chaque sommet initial de type *delivery* est affecté l'une des trois situations possibles suivantes :

- Le début d'une journée de travail, $l_{i^-} \in [360; 600]$ (entre 6h et 10h) ;
- Le retour au domicile, $l_{i^-} \in [960; 1200]$ (entre 16h et 20h) ;
- Un déplacement aléatoire, $l_{i^-} \in [0; 1440]$ (entre 0h et 24h).

Le plus court chemin SP_i est calculé dans le graphe entre le sommet initial de *pickup* et le sommet initial de *delivery*, afin de ne pas fixer des fenêtres de temps impossibles à respecter. Enfin, les fenêtres du *pickup* $[e_{i^+}; l_{i^+}]$ et du *delivery* $[e_{i^-}; l_{i^-}]$ sont calculées comme suit :

- $e_{i^+} = l_{i^-} - \alpha \times SP_i$;
- $l_{i^+} = l_{i^-} - SP_i$;
- $e_{i^-} = e_{i^+} + SP_i$.

avec $\alpha \in \mathbb{N}$ représentant le coefficient de tolérance de temps supplémentaire par rapport au plus court chemin.

2.4.5 Données supplémentaires

Dans toutes les instances, la durée maximale d'une tournée est fixée à 480, la capacité maximale d'un véhicule public est égale à 10 (5 pour un véhicule privé) et le temps maximal d'un client dans un véhicule est de 240 soit un transport de durée maximale de 4h. Pour chaque sommet, le temps de service est égal à 2. Un nombre de véhicules publics important est fourni afin d'aider l'algorithme à trouver des solutions initiales. Peu d'entre eux sont utilisés dans les solutions comme montré dans la section 2.5.

2.5 Résultats

Les résultats présentés dans cette section ont été réalisés sur des instances classiques de la littérature dans le cas de la résolution du DARP. Lors de la résolution du DARP-PV-AN, les instances étudiées sont des versions modifiées des instances classiques ou bien de nouvelles instances proposées (section 2.4).

Tous les tests ont été réalisés sur une machine sous Windows 7 équipée de 16Gio de RAM et d'un processeur Intel Core i7-4790@3.60GHz avec un seul thread. Le solveur CPLEX en version 12.6 a été utilisé pour calculer les solutions optimales pour le modèle linéaire proposé avec un temps de calcul maximal d'une heure. Considérant (Dongarra et al., 2011), la vitesse de la machine est d'environ 4.130 Gflops et toutes les méthodes ont été codées en C++. Les instances et les résultats sont disponibles sur la page « http://fc.isima.fr/~brevet/DARP-PV-AN.html ».

2.5.1 Évaluation du modèle linéaire

Les résultats du modèle linéaire résolu avec CPLEX sont présentés dans le *Tableau 2.1*. Ce dernier est divisé en 4 parties principales, chacune étant composée d'instances correspondant à un même ensemble de clients mais à une composition de véhicules publics/privés différente. La colonne n indique le nombre de requêtes (égale au nombre de clients), N^T représente le nombre total de sommets (initiaux + alternatifs), K représente le nombre de véhicules publics disponibles, K' le nombre de véhicules privés disponibles et Obj la valeur de la fonction objectif, c'est-à-dire la somme des distances parcourues par les véhicules définie dans le modèle linéaire (e). LB est la meilleure borne inférieure fournie par CPLEX dans le temps imparti. T^* est le temps nécessaire pour trouver la meilleure solution (dans la limite des 3600 secondes) et TT représente le temps total pour terminer l'exploration de l'arbre de recherche en entier. Les colonnes $\#K$ et $\#K'$ indiquent respectivement le nombre de véhicules publics et le nombre de véhicules privés utilisés dans la meilleure solution trouvée. Le symbole / indique qu'aucune solution n'a été trouvée dans le délai imparti.

Une simple observation concerne le nombre de véhicules privés et publics utilisés en fonction du nombre de véhicules privés disponibles. Comme on peut le constater, de l'instance PCD_2v_10n_0p à l'instance PCD_5v_14n_7p, plus il y a de véhicules privés disponibles, plus ces derniers sont utilisés et moins on utilise de véhicules publics. Cela conduit par exemple, à une solution de coût 35,36 sur l'instance PCD_2v_12n_0p avec 2 véhicules publics et 0 véhicule privé, minimisée à un coût de 19,20 lorsque l'on ajoute 6 véhicules privés (instance PCD_5v_12n_6p). On observe également la difficulté du modèle à résoudre des instances avec plus de 7 demandes : le délai maximal de 3600 secondes est atteint et les solutions optimales ne sont pas obtenues pour 6 instances. Pour 3 de ces 6 instances (PCD_0v_14n_2p, PCD_1v_16n_1p et PCD_0v_16n_2p), aucune solution réalisable n'a été trouvée. C'est la raison pour laquelle une métaheuristique a été proposée afin de traiter les instances de taille réaliste dans les sections suivantes. On peut également noter que l'ajout d'un seul véhicule privé potentiel, passant K' de 0 à 1, peut améliorer sensiblement la valeur de la solution. Cependant, le temps de calcul requis augmente également fortement. Pour rappel, les configurations avec 0 véhicule privé correspondent au DARP classique, tandis que celles avec 0 véhicule public forment un problème de covoiturage (*Ridesharing Problem*). La dernière instance de chaque partie correspond à un contexte sans restriction dans lequel tous les clients autorisent l'utilisation de leur véhicule privé mais peuvent également prendre un véhicule public.

Tableau 2.1 : Résultat du modèle linéaire sur les instances modifiées de (Ropke et al., 2007)

Name	n	N^T	K	K'	Obj	LB	T*(s)	TT (s)	#K	#K'
PCD_2v_10n_0p	5	31	2	0	32.90	32.90	0.84	0.95	2	0
PCD_1v_10n_1p	5	31	1	1	28.92	28.92	1.67	1.89	1	1
PCD_2v_10n_1p	5	31	2	1	28.92	28.92	2.34	2.51	1	1
PCD_0v_10n_2p	5	31	0	2	29.62	29.62	14.45	14.91	0	2
PCD_5v_10n_5p	5	31	5	5	20.10	20.10	5.04	7.11	0	3
PCD_2v_12n_0p	6	37	2	0	32.36	32.36	16.65	20.94	2	0
PCD_1v_12n_1p	6	37	1	1	27.64	27.64	110.25	114.89	1	1
PCD_2v_12n_1p	6	37	2	1	27.64	27.64	273.78	367.58	1	1
PCD_0v_12n_2p	6	37	0	2	25.05	25.05	21.98	38.58	0	2
PCD_5v_12n_6p	6	37	5	6	19.20	19.20	288.34	316.64	0	5
PCD_2v_14n_0p	7	42	2	0	35.98	35.98	195.06	203.47	2	0
PCD_1v_14n_1p	7	42	1	1	32.10	32.10	1682.64	2438.23	1	1
PCD_2v_14n_1p	7	42	2	1	32.10	32.10	984.80	1975.21	1	1
PCD_0v_14n_2p	7	42	0	2	/	26.50	/	3600.00	/	/
PCD_5v_14n_7p	7	42	5	7	24.37	21.90	3521.21	3600.00	0	6
PCD_2v_16n_0p	8	47	2	0	48.19	48.19	380.57	1485.53	2	0
PCD_1v_16n_1p	8	47	1	1	/	34.46	/	3600.00	/	/
PCD_2v_16n_1p	8	47	2	1	48.78	34.58	3320.82	3600.00	2	1
PCD_0v_16n_2p	8	47	0	2	/	33.50	/	3600.00	/	/
PCD_5v_16n_8p	8	47	5	8	31.60	27.65	3316.85	3600.00	0	6

2.5.2 Paramétrage de l'opérateur de mutation sur l'algorithme ELS

La mutation repose sur deux opérateurs (voir section 2.3.2) afin de définir la diversité. L'efficacité de l'équilibre entre le premier et le deuxième opérateur est évaluée sur 5 instances proposées par (Cordeau et Laporte, 2003). Les résultats sont présentés dans le *Tableau 2.2*. Dans le tableau, BKS fait référence au coût de la meilleure solution publiée, et le couple $(\alpha, 1-\alpha)$ définit l'équilibre entre le premier et le second opérateur de mutation, α étant la probabilité d'appeler le premier opérateur (suppression d'une requête et réinsertion) et par conséquent, $1-\alpha$ la probabilité d'appeler le second (*cut and paste*). Ainsi, $(1;0), (0;1)$ et $(0{,}7;0{,}3)$ consistent respectivement à n'utiliser que le premier opérateur, que le deuxième opérateur, ou l'équilibre choisi. Pour chacun des trois couples, Obj est la meilleure solution trouvée par l'ELS et $Gap\%$ est l'écart relatif par rapport à la meilleure solution (BKS). Ces résultats illustrent l'importance de la collaboration entre les deux opérateurs de mutation.

Tableau 2.2 : Comparaison des opérateurs de mutation sur les instances de (Cordeau et Laporte, 2003)

Name	BKS	(1 ; 0)		(0 ; 1)		(0.7 ; 0.3)	
		Obj	Gap%	Obj	Gap%	Obj	Gap%
R1a	190.02	190.79	0.41	190.79	0.41	190.02	0.00
R2a	301.34	308.42	2.35	308.36	2.33	301.34	0.00
R3a	532.00	539.71	1.45	558.52	4.98	534.28	0.43
R4a	570.25	604.51	6.01	624.09	9.44	572.95	0.47
R5a	626.93	658.71	5.07	689.47	9.98	640.27	2.13

2.5.3 Évaluation de la métaheuristique ELS sur les instances de (Cordeau et Laporte, 2003)

La première évaluation de la métaheuristique ELS a été effectuée sur 5 instances proposées à l'origine par (Cordeau et Laporte, 2003) et les résultats sont présentés dans le *Tableau 2.3*. Ces résultats concernent le DARP classique : aucun véhicule privé n'est disponible et aucun sommet alternatif ne peut donc être utilisé. Dans le tableau, BKS fait référence au coût de la meilleure solution publiée obtenue par (Chassaing et al., 2016) et la colonne Obj fait référence au coût de la solution fournie par l'algorithme ELS. La colonne T^* donne le temps nécessaire en secondes pour atteindre la meilleure solution et TT le temps total de la métaheuristique. Le pourcentage d'écart entre la solution de l'ELS et la BKS a été calculée comme suit :

$$Gap\% = \frac{Obj \times 100}{BKS} - 100$$

Des solutions équivalentes à la BKS ont été trouvées sur les instances $R1a$ et $R2a$. Les instances $R3a$ et $R4a$ ont été résolues avec des résultats supérieurs à 0,5% de la BKS. Ces résultats illustrent le fait que la métaheuristique ELS proposée fournit également de bons résultats sur le DARP classique, qui est une version spécifique du DARP-PV-AN.

Tableau 2.3 : Résultats de l'ELS sur 5 instances de (Cordeau et Laporte, 2003)

Type	Name	n	N^T	K	BKS	Obj	Gap%	T*(s)	TT(s)	#K
	R1a	24	48	3	190.02	190.02	0.00	0.00	15.00	3
	R2a	48	96	5	301.34	301.34	0.00	24.00	60.00	5
DARP	R3a	72	144	7	532.00	534.28	0.43	53.80	150.00	7
	R4a	96	192	9	570.25	572.95	0.47	324.00	456.00	8
	R5a	120	240	11	626.93	640.27	2.13	360.00	498.00	11

Dans le *Tableau 2.4*, les instances du *Tableau 2.3* sont modifiées pour inclure les véhicules privés et les sommets alternatifs. Pour chaque instance DARP d'origine, le nombre de véhicules privés autorisés varie de 1 à 3. La colonne UB fait référence à la meilleure solution trouvée des instances DARP classiques d'où sont issues les instances modifiées, ce qui constitue une borne supérieure valide pour les instances DARP-PV-AN : la fonction objectif devrait diminuer par rapport au nombre de véhicules privés disponibles. Ceci peut être observé sur les instances issues de $R1a$ et dans une moindre mesure, sur celles issues de $R2a$ et $R3a$. Toutefois, pour les instances $R4a$ et $R5a$, la taille de ces dernières (650 et 819 sommets) empêche la méthode d'atteindre des solutions proches de BKS.

Tableau 2.4 : Résultats de l'ELS sur les instances modifiées de (Cordeau et Laporte, 2003)

Type	Name	n	N^T	K	K'	UB	Obj	Gap%	T*(s)	TT(s)	#K	#K'
DARP-PV-AN	R1a-1p-pn	24	168	3	1	190.02	190.02	0.00	0.00	15.00	3	0
	R1a-2p-pn	24	168	3	2	190.02	190.02	0.00	0.00	15.00	3	0
	R1a-3p-pn	24	168	3	3	190.02	186.90	-1.64	7.20	15.00	2	1
	R2a-1p-pn	48	318	5	1	301.34	299.84	-0.50	38.40	60.00	5	1
	R2a-2p-pn	48	318	5	2	301.34	294.39	-2.31	44.40	60.00	5	2
	R2a-3p-pn	48	318	5	3	301.34	294.92	-2.13	32.40	60.00	5	2
	R3a-1p-pn	72	479	7	1	532.00	529.90	-0.39	85.20	150.00	7	1
	R3a-2p-pn	72	479	7	2	532.00	530.39	-0.30	72.00	150.00	7	2
	R3a-3p-pn	72	479	7	3	532.00	522.57	-1.77	76.20	150.00	7	3
	R4a-1p-pn	96	650	9	1	570.25	574.94	0.82	298.80	438.00	8	0
	R4a-2p-pn	96	650	9	2	570.25	572.99	0.48	283.80	438.00	8	0
	R4a-3p-pn	96	650	9	3	570.25	573.10	0.50	256.80	438.00	8	1
	R5a-1p-pn	120	819	11	1	626.93	629.18	0.36	367.20	498.00	11	1
	R5a-2p-pn	120	819	11	2	626.93	633.76	1.09	231.00	498.00	11	2
	R5a-3p-pn	120	819	11	3	626.93	638.50	1.85	190.20	498.00	11	3

2.5.4 Évaluation de la métaheuristique ELS sur les nouvelles instances

Les tests précédents sur la métaheuristique ELS ont été effectués sur des instances classiques adaptées au DARP-PV-AN, mais proposées à l'origine pour d'autres problèmes. L'algorithme a également été testé sur un nouvel ensemble d'instances dédiées au DARP-PV-AN, avec des sommets alternatifs communs à plusieurs requêtes. D'après la littérature, les instances de test présentant ces caractéristiques n'étaient pas disponibles pour la version du DARP étudiée dans cette thèse. Les résultats de l'algorithme sur ces nouvelles instances sont proposés dans le *Tableau 2.5*.

L'utilisation de véhicules privés tout en respectant la vie privée des utilisateurs améliore considérablement les solutions du DARP classique, comme le souligne la colonne $Gap\%$ dans le *Tableau 2.5*. Par exemple, sur les instances PCD_40, les distances parcourues sont jusqu'à 50,17% inférieures lorsque tous les clients ont la possibilité d'utiliser leur propre véhicule (même si seulement 12 véhicules sur 20 sont vraiment utilisés). Ce comportement est valable pour chaque groupe d'instances. Les économies dépendent notamment de la capacité des véhicules privés qui a été définie à 5 sur ces instances. Pour une capacité plus restreinte, l'amélioration de la solution sans véhicules privés aurait été plus limitée. Pour toutes les instances, l'algorithme ELS utilise la totalité du temps qui lui est octroyé. Cependant, le temps nécessaire pour trouver la meilleure solution augmente lorsque des véhicules privés peuvent être utilisés, par rapport au temps nécessaire lorsqu'aucun véhicule privé n'est disponible (c'est-à-dire $K' = 0$).

Tableau 2.5 : Résultat de l'ELS sur les nouvelles instances

Name	n	N^T	***K***	***K'***	***Obj***	***Gap%***	***T*(s)***	***TT(s)***	***#K***	***#K'***
PCD_20_0VP	10	61	4	0	811.35	/	0.00	60.00	3	0
PCD_20_2VP	10	61	4	2	730.65	-9.94	30.00	60.00	3	1
PCD_20_4VP	10	61	4	4	730.65	-9.94	30.00	60.00	3	1
PCD_20_6VP	10	61	4	6	658.24	-18.87	30.00	60.00	2	4
PCD_20_10VP	10	61	4	10	435.52	-46.32	34.80	60.00	0	7
PCD_40_0VP	20	125	8	0	1135.31	/	6.60	120.00	4	0
PCD_40_2VP	20	125	8	2	1028.41	-9.42	60.60	120.00	4	2
PCD_40_6VP	20	125	8	6	747.73	-34.14	118.80	120.00	1	6
PCD_40_8VP	20	125	8	8	667.43	-41.21	78.00	120.00	1	6
PCD_40_10VP	20	125	8	10	647.31	-42.98	60.60	120.00	1	8
PCD_40_20VP	20	125	8	20	565.69	-50.17	88.80	120.00	0	12
PCD_60_0VP	30	191	20	0	1530.00	/	46.20	180.00	6	0
PCD_60_2VP	30	191	20	2	1493.06	-2.41	115.80	180.00	4	2
PCD_60_8VP	30	191	20	8	1240.68	-18.91	165.00	180.00	4	5
PCD_60_10VP	30	191	20	10	1044.67	-31.72	178.80	180.00	2	8
PCD_60_16VP	30	191	20	16	916.88	-40.07	146.40	180.00	1	9
PCD_60_30VP	30	191	20	30	961.22	-37.18	97.20	180.00	0	17
PCD_80_0VP	40	243	20	0	2060.88	/	2.40	240.00	7	0
PCD_80_2VP	40	243	20	2	2060.88	0.00	2.40	240.00	7	0
PCD_80_10VP	40	243	20	10	1991.44	-3.37	129.60	240.00	5	2
PCD_80_14VP	40	243	20	14	1840.34	-10.70	189.00	240.00	4	5
PCD_80_20VP	40	243	20	20	1509.49	-26.76	240.00	240.00	2	11
PCD_80_40VP	40	243	20	40	1447.02	-29.79	162.00	240.00	1	18
PCD_100_0VP	50	307	50	0	2415.43	/	81.60	300.00	7	0
PCD_100_4VP	50	307	50	4	2386.15	-1.21	163.20	300.00	7	3
PCD_100_14VP	50	307	50	14	2029.22	-15.99	268.80	300.00	4	8
PCD_100_18VP	50	307	50	18	2078.92	-13.93	294.60	300.00	4	9
PCD_100_26VP	50	307	50	26	2079.45	-13.91	292.20	300.00	4	10
PCD_100_50VP	50	307	50	50	1888.48	-21.82	242.40	300.00	1	21
PCD_120_0VP	60	371	50	0	2928.59	/	257.40	360.00	10	0
PCD_120_4VP	60	371	50	4	2804.60	-4.23	268.20	360.00	8	1
PCD_120_16VP	60	371	50	16	2465.74	-15.80	335.40	360.00	5	5
PCD_120_20VP	60	371	50	20	2407.83	-17.78	320.40	360.00	6	4
PCD_120_30VP	60	371	50	30	2581.43	-11.85	269.40	360.00	7	3
PCD_120_60VP	60	371	50	60	2106.29	-28.08	360.00	360.00	2	20

Une évaluation de la qualité de la solution a également été effectuée selon les critères définis par (Cordeau et Laporte, 2003) : TRT (*Total Riding Time*), TWT (*Total Waiting Time*) et TD (*Total Duration*). Les figures 2.19, 2.20 et 2.21 reportent les valeurs moyennes obtenues à partir du *Tableau 2.5*. Comme le montre la *Figure 2.19*, tous les critères ont été améliorés lorsque les véhicules privés sont disponibles et le TWT est proche de 0% si tous les clients peuvent utiliser leur propre véhicule (ce qui ne signifie pas nécessairement que tous seront utilisés).

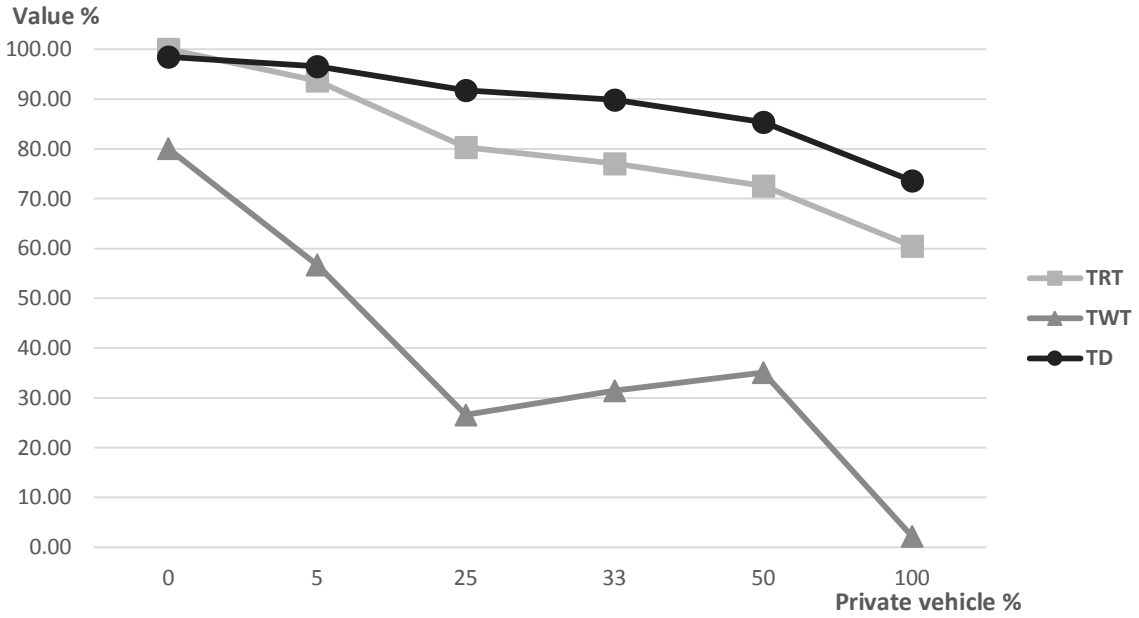


Figure 2.19 : Qualité de service moyenne selon la définition de (Cordeau et Laporte, 2003)

Il est également possible d'évaluer l'effet des véhicules privés sur l'efficacité des trajets en mesurant le nombre d'arcs utilisés en fonction du nombre de véhicules privés disponibles. Si le nombre d'arcs est plus faible, les trajets sont plus directs et les détours sont moins importants. Comme le montre la *Figure 2.20*, le nombre d'arcs utilisés passe de 324 en moyenne avec 0% de véhicules privés à 219 avec 100% de véhicules privés disponibles. Les contraintes imposées aux véhicules privés n'incluent pas une durée maximale des tournées, mais uniquement des temps maximaux passés dans le véhicule pour chaque client (*maximal riding time*) Les déplacement en véhicules privés sont très limités et utilisent moins d'arcs que les déplacements attribués aux véhicules publics. Les causes de ces différences sont :

- Le temps de parcours (*riding time*) est défini entre chaque sommet de *pickup* et de *delivery* ;
- Le sommet de départ d'un véhicule privé est situé sur un sommet de *pickup* et non sur un dépôt centralisé (de même pour le sommet de fin d'un véhicule privé situé sur un sommet de *delivery*) ;
- Le temps maximal de parcours des clients (*maximal riding time*) est plus faible que le temps maximal de durée d'une tournée (*maximal Total Duration*).

Les analyses suivantes évaluent, dans chaque solution, la moyenne des écarts entre la distance de transport optimal d'un client (c'est-à-dire sans détour entre son *pickup* et son *delivery*) et la distance de transport réel du client. Pour un client donné, cet écart est représenté comme le rapport entre sa distance parcourue dans la solution actuelle et la plus courte distance entre

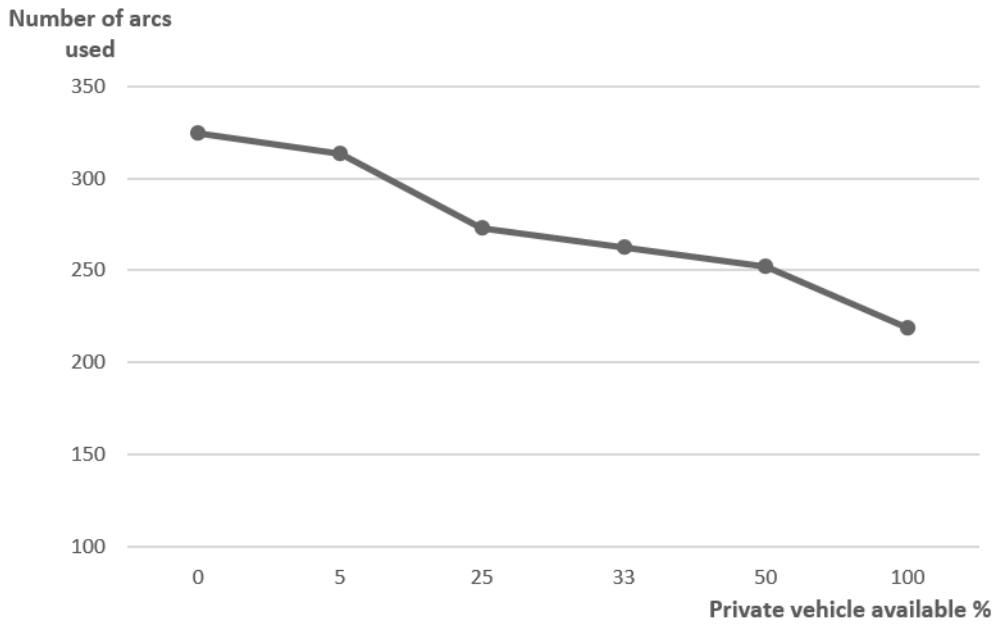


Figure 2.20 : Moyenne du nombre d'arcs utilisés

son *pickup* et son *delivery*. Par exemple, un client avec un plus court trajet de 10, mais effectuant dans la solution un trajet de 15, parcourt une distance égale à 150% du plus court chemin. Puisqu'un véhicule peut gérer plusieurs requêtes en même temps, le plus court chemin de chaque requête peut très bien ne pas être utilisé car des détours sont effectués pour satisfaire d'autres requêtes. Cet écart moyen en fonction du nombre de véhicules privés disponibles a été étudié. Comme le montre la *Figure 2.21*, ou 100% signifie que le chemin le plus court possible pour une requête est utilisé, la longueur moyenne du transport d'un client correspond à 161,44% du chemin le plus court lorsqu'aucun véhicule privé n'est utilisé. Avec les véhicules privés, cet écart diminue progressivement pour atteindre 129,29%. Cela signifie également un gain de temps moyen de 19,91% de temps pour une requête de transport.

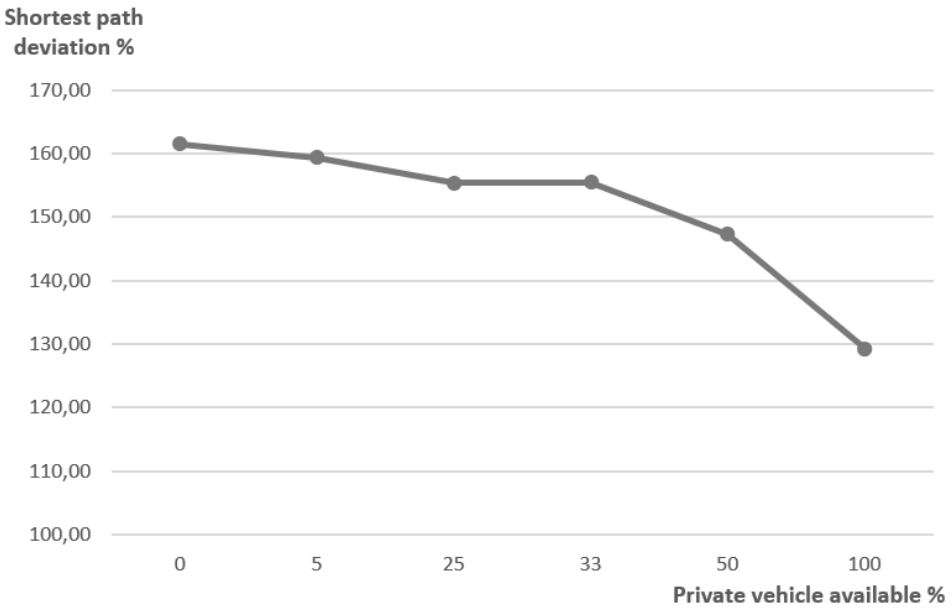


Figure 2.21 : Déviation moyenne du plus court chemin pour chaque requête

L'observation suivante porte sur l'utilisation des sommets alternatifs communs à plusieurs requêtes par les véhicules privés. Sur la *Figure 2.22*, certains clients partagent des sommets alternatifs. Ainsi, ces sommets peuvent être utilisés en tant que lieu de covoiturage pour traiter plusieurs requêtes en même temps afin d'optimiser la valeur de la fonction objectif. Dans cet exemple, une partie d'un voyage privé est représentée. Le trajet en véhicule privé est indiqué par les arcs 15, 16, 17 et 18. Les sommets alternatifs sont en gris (190 et 205) et les sommets initiaux rayés (191, 204 et 220). Comme on peut le constater, le sommet alternatif 205 est partagé par trois requêtes ayant pour sommets initiaux 191, 204 et 220. Étant donné que le sommet alternatif 205 est visité, les trois requêtes associées peuvent être effectuées en même temps. Dans ce cas, il s'agit d'un triple *delivery*, mais cela peut correspondre à une combinaison quelconque de *pickups* et de *deliveries*.

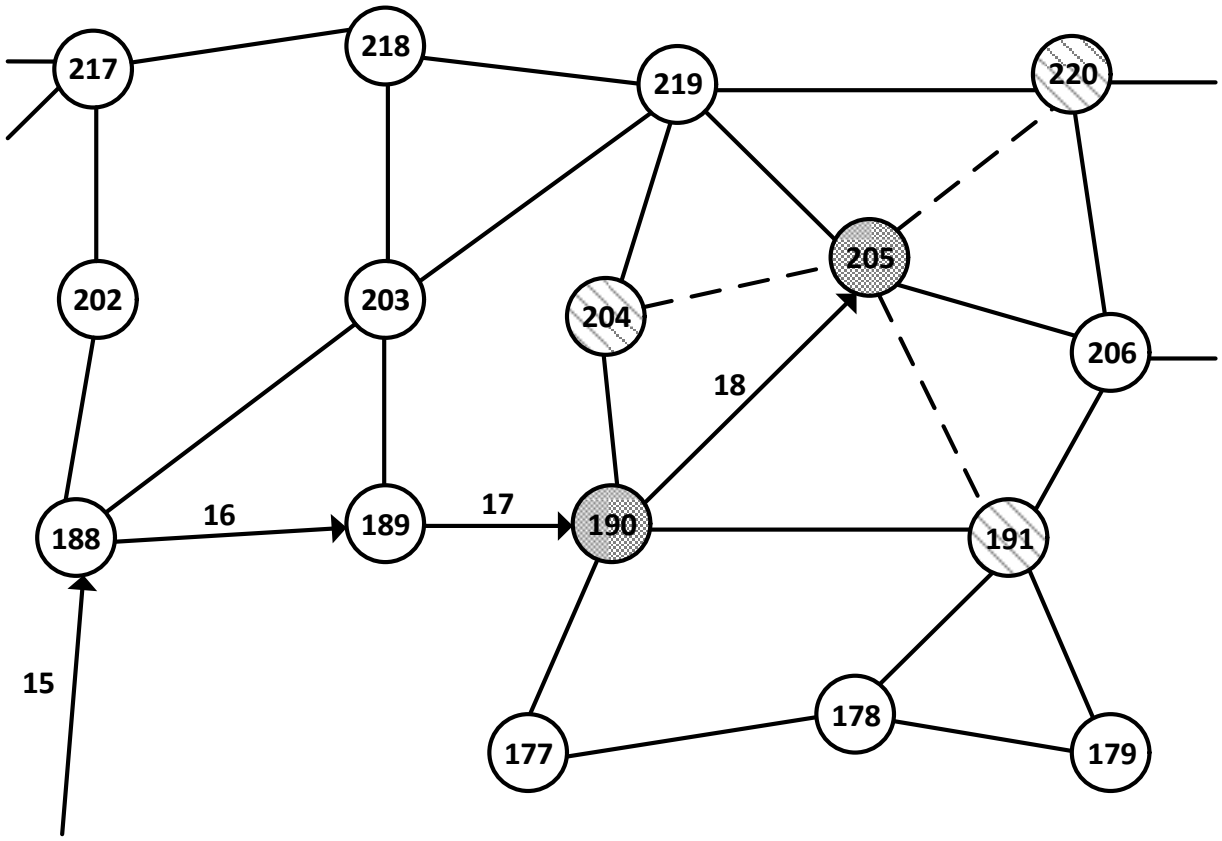


Figure 2.22 : Sommet 205 utilisé comme sommet de « réunion » dans la tournée 7 de l'instance PCD_100_14VP

2.5.5 Évaluation de l'ELS avec durée dépendante du véhicule

L'algorithme ELS a également été évalué lorsque le temps de trajet entre deux sommets dépend du véhicule le parcourant. Le but de cette évaluation est de tester l'utilité des véhicules privés même si ces derniers ne disposent pas des mêmes avantages que les véhicules publics, comme par exemple des voies dédiés aux bus/taxis dans les villes ou encore la priorité lors de feux tricolores. Pour ce faire, lorsque le coût d'une tournée est calculé dans la procédure d'évaluation, sa valeur est multipliée par un coefficient β :

- La valeur de β est égale à 1 si c'est un véhicule public ;

- La valeur de β est est choisie aléatoirement entre une valeur Min et une valeur Max si c'est un véhicule privé. Ce coefficient peut être inférieur à 1 ce qui signifie que les véhicules privés sont plus rapides que les véhicules publics, ou supérieur à 1, ce qui signifie le contraire.

Les tests ont été effectués sur les instances PCD_20_2, PCD_20_4 et PCD_20_6 proposées dans la section 2.4 avec des valeurs Min allant de 0,80 à 2,20 et Max allant de 1,00 à 2,40. La colonne $Obj*$ représente la valeur de la fonction objectif sans la multiplication du coût des tournées par le coefficient β. La colonne Obj représente cette valeur avec la multiplication du coût de tournées par β. Comme le montre le *Tableau 2.6*, l'utilisation de coefficients inférieurs à 1 impose l'utilisation de plus de véhicules privés (colonne $\#K'$) mais ne supprime pas pour autant l'utilisation des véhicules publics. Cela diminue également le coût de la fonction objectif (colonne Obj). Au contraire, des coefficients supérieurs à 1 réduisent l'utilisation de véhicules privés dans les solutions trouvées. Cependant, ces derniers sont toujours utilisés jusqu'à ce que β atteigne une valeur suffisamment grande, autour de 2 dans le *Tableau 2.6* et la *Figure 2.23*.

Tableau 2.6 : Résultat de l'ELS sur les instances avec coût dépendant du véhicule

Name	Min	Max	Obj*	Obj	Gap%	#K	#K'
PCD_20_2VP	0.80	1.00	811.35	696.92	-14.10	2	2
	1.00	1.20	811.35	736.46	-9.23	3	1
	1.20	1.40	811.35	749.38	-7.64	3	1
	1.40	1.60	811.35	762.31	-6.04	3	1
	1.60	1.80	811.35	775.23	-4.45	3	1
	1.80	2.00	811.35	788.16	-2.86	3	1
	2.00	2.20	811.35	801.08	-1.27	3	1
	2.20	2.40	811.35	811.35	0.00	3	0
PCD_20_4VP	0.80	1.00	811.35	690.43	-14.90	2	4
	1.00	1.20	811.35	736.46	-9.23	3	1
	1.20	1.40	811.35	749.38	-7.64	3	1
	1.40	1.60	811.35	762.31	-6.04	3	1
	1.60	1.80	811.35	775.23	-4.45	3	1
	1.80	2.00	811.35	788.16	-2.86	3	1
	2.00	2.20	811.35	801.08	-1.27	3	1
	2.20	2.40	811.35	811.35	0.00	3	0
PCD_20_6VP	0.80	1.00	811.35	630.86	-22.25	2	4
	1.00	1.20	811.35	672.27	-17.14	2	4
	1.20	1.40	811.35	702.17	-13.46	2	2
	1.40	1.60	811.35	720.53	-11.19	2	2
	1.60	1.80	811.35	738.89	-8.93	2	2
	1.80	2.00	811.35	757.25	-6.67	2	2
	2.00	2.20	811.35	775.61	-4.41	2	2
	2.20	2.40	811.35	793.97	-2.14	2	2

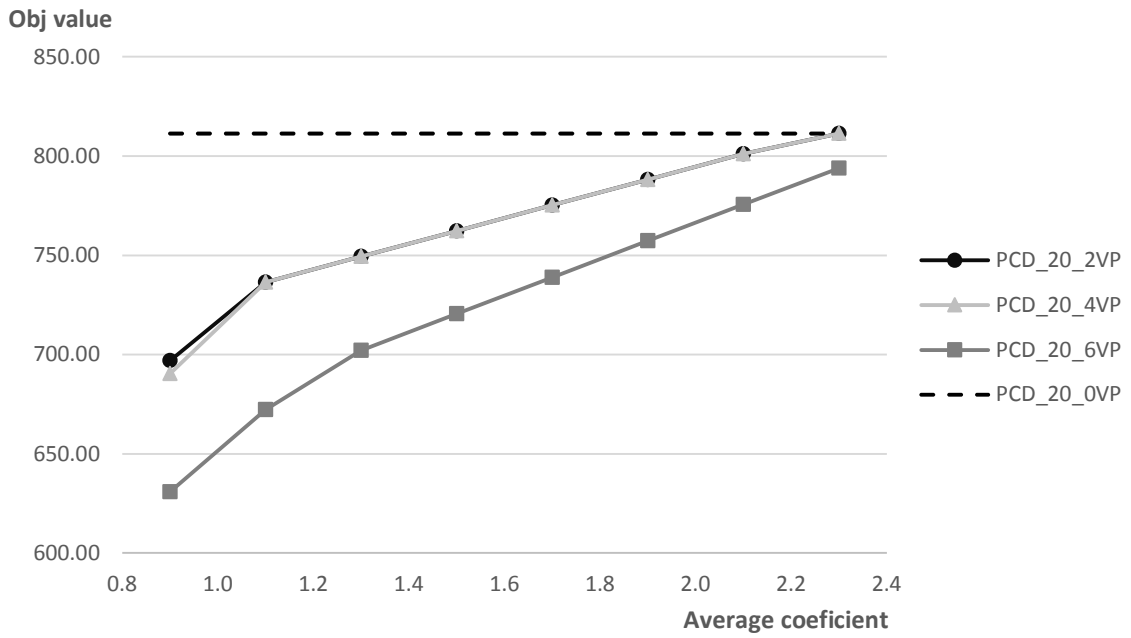


Figure 2.23 : Effet du coefficient β sur le coût d'une solution

2.6 Conclusion

Dans ce premier chapitre, le nouveau problème du transport à la demande avec véhicules privés et sommets alternatifs (DARP-PV-AN) a été présenté. Ce problème permet de combiner deux flottes de véhicules : une flotte de véhicules publics situés sur un dépôt, et une flotte de véhicules privés, chacun appartenant à un client. La première flotte conduit à un système centralisé (société de transport, publique ou privée) tandis que la seconde est décentralisée. Les emplacements de *pickups* et de *deliveries* pour chaque requête de transport sont associés à un sommet initial ainsi qu'à plusieurs sommets alternatifs. Ces derniers sont utilisés pour améliorer la confidentialité du client.

Les sommets initiaux sont utilisés par les véhicules publics, tandis que les sommets alternatifs sont utilisés par les véhicules privés. Les sommets alternatifs empêchent de connaître les sommets initiaux de départ et d'arrivé des clients par les tierces personnes présentes dans le véhicules, c'est-à-dire, que les sommets initiaux et finaux restent privés. De plus, un même sommet alternatif peut être partagé par plusieurs clients : un véhicule privé passant par un tel sommet a la possibilité de satisfaire plusieurs demandes en une seule fois.

Un modèle linéaire en nombres entiers mixtes a été proposé ainsi qu'une métaheuristique de type « recherche locale évolutive » (*Evolutionary Local Search* – ELS). Ces deux propositions ont été testées sur un ensemble d'instances de la littérature et leurs versions modifiées. De plus, un nouvel ensemble d'instances, dédié au DARP-PV-AN, a également été créé et évalué par l'algorithme ELS.

Les résultats montrent que l'utilisation de véhicules privés peut entraîner certaines améliorations sur plusieurs critères. Premièrement, la somme des distances parcourues est considérablement réduite car les véhicules privés n'ont pas besoin de partir et revenir au dépôt, et ont un choix plus large de sommets de *pickup* et *delivery* pour un même client. La qualité du service (*Total Duration* – TD, *Total Riding Time* – TRT, *Total Waiting Time* – TWT) est améliorée et la distance parcourue par chaque client est plus courte, ce qui engendre moins de détours. De plus, le nombre d'arcs utilisés est réduit et mieux réparti sur l'ensemble

du graphe. Ainsi, l'effet sur le trafic global est diminué de manière non négligeable. Couplé à une distance totale inférieure, cela pourrait également entraîner une réduction des émissions globales de NOx/CO_2.

Un point d'étude concerne un problème soulevé par les opérateurs de mutation et de recherche locale. Lorsqu'une modification est effectuée sur une tournée par un de ces deux opérateurs, son gain potentiel est calculé à partir des sommets initiaux, pour fournir rapidement l'estimation d'un tel mouvement. Cependant, si le mouvement est considéré comme intéressant et est effectué avec une tournée privée, la fonction d'évaluation va calculer le coût de la tournée en prenant en compte les sommets alternatifs. Cette estimation du coût de mouvement, utilisant les sommets initiaux pour les tournées privées, est permise de par la proximité des sommets alternatifs à leur sommet initial. La prise en compte des sommets alternatifs lors des estimations de gains des méthodes de recherche locale ou mutation est coûteuse en termes de temps de calcul sur l'algorithme ELS. Un autre algorithme est donc proposé dans le chapitre suivant traitant de façon plus précise le problème des sommets alternatifs dans les tournées privés.

Références

Agatz, N., Erera, A., Savelsbergh, M., Wang, X., 2012. Optimization for dynamic ride-sharing: A review. Eur. J. Oper. Res. 223, 295‑303. https://doi.org/10.1016/j.ejor.2012.05.028

Aïvodji, U.M., Gambs, S., Huguet, M.-J., Killijian, M.-O., 2016. Meeting points in ridesharing: A privacy-preserving approach. Transp. Res. Part C Emerg. Technol. 72, 239‑253. https://doi.org/10.1016/j.trc.2016.09.017

Artigues, C., Anceaume, E., Gambs, S., Guette, G., Hurfin, M., Schettini, F., Deswarte, Y., Guiochet, J., Huguet, M.-J., Killijian, M.-O., Powell, D., Roy, M., Bidan, C., Prigent, N., 2012. AMORES: an architecture for mobiquitous resilient systems, in: Proceedings of the 1st European Workshop on AppRoaches to MObiquiTous Resilience - ARMOR '12. Présenté à the 1st European Workshop, ACM Press, Sibiu, Romania, p. 1‑6. https://doi.org/10.1145/2222436.2222443

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. https://doi.org/10.1016/j.trb.2014.05.007

Bruck, B.P., Incerti, V., Iori, M., Vignoli, M., 2017. Minimizing CO2 emissions in a practical daily carpooling problem. Comput. Oper. Res. 81, 40 ‑ 50. https://doi.org/10.1016/j.cor.2016.12.003

Chassaing, M., Duhamel, C., Lacomme, P., 2016. An ELS-based approach with dynamic probabilities management in local search for the Dial-A-Ride Problem. Eng. Appl. Artif. Intell. 48, 119‑133. https://doi.org/10.1016/j.engappai.2015.10.002

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. https://doi.org/10.1016/S0191-2615(02)00045-0

Cottrill, C.D., Thakuriah, P., 2015. Location privacy preferences: A survey-based analysis of consumer awareness, trade-off and decision-making. Transp. Res. Part C Emerg. Technol. 56, 132‑148. https://doi.org/10.1016/j.trc.2015.04.005

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numer. Math. 1, 269‑271. https://doi.org/10.1007/BF01386390

Dongarra, J., Luszczek, P., Feautrier, P., Zee, F.G., Chan, E., Geijn, R.A., Bjornson, R., Dongarra, J., Luszczek, P., Philippe, B., Sameh, A., Philippe, B., Sameh, A., Dongarra, J., Luszczek, P., Steele, G.L., Gustafson, J.L., Dongarra, J., Luszczek, P., Becker, A., Zheng, G., Kalé, L.V., Pingali, K., Carro, M., Hermenegildo, M., Banerjee, U., Wismüller, R., 2011. LINPACK Benchmark, in: Padua, D. (Éd.), Encyclopedia of Parallel Computing. Springer US, Boston, MA, p. 1033‑1036. https://doi.org/10.1007/978-0-387-09766-4_155

Duhamel, C., Lacomme, P., Prodhon, C., Prins, C., 2009. A GRASP-ELS approach for real-life Location Routing Problems, in: 2009 International Conference on Computers & Industrial Engineering. Présenté à Industrial Engineering (CIE39), IEEE, Troyes, France, p. 1082‑1087. https://doi.org/10.1109/ICCIE.2009.5223564

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

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M.-E., Wang, X., Koenig, S., 2013. Ridesharing: The state-of-the-art and future directions. Transp. Res. Part B Methodol. 57, 28‑46. https://doi.org/10.1016/j.trb.2013.08.012

Gambs, S., Killijian, M.-O., del Prado Cortez, M.N., 2010. Show me how you move and I will tell you who you are, in: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS - SPRINGL ’10. Présenté à the 3rd ACM SIGSPATIAL International Workshop, ACM Press, San Jose, California, p. 34. https://doi.org/10.1145/1868470.1868479

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

Lourenço, H.R., Martin, O.C., Stützle, T., 2003. Iterated Local Search, in: Glover, F., Kochenberger, G.A. (Éd.), Handbook of Metaheuristics. Kluwer Academic Publishers, Boston, p. 320‑353. https://doi.org/10.1007/0-306-48056-5_11

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

Potvin, J.-Y., Rousseau, J.-M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. Eur. J. Oper. Res. 66, 331‑340. https://doi.org/10.1016/0377-2217(93)90221-8

Prins, C., 2009. A GRASP × Evolutionary Local Search Hybrid for the Vehicle Routing Problem, in: Pereira, F.B., Tavares, J. (Éd.), Bio-Inspired Algorithms for the Vehicle Routing Problem. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 35‑53. https://doi.org/10.1007/978-3-540-85152-3_2

Quercia, D., Lathia, N., Calabrese, F., Di Lorenzo, G., Crowcroft, J., 2010. Recommending Social Events from Mobile Phone Location Data, in: 2010 IEEE International Conference on Data Mining. Présenté à 2010 IEEE 10th International Conference on Data Mining (ICDM), IEEE, Sydney, Australia, p. 971‑976. https://doi.org/10.1109/ICDM.2010.152

Ropke, S., Cordeau, J.-F., Laporte, G., 2007. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks 49, 258‑272. https://doi.org/10.1002/net.20177

Wilson, N.H., Sussman, J., Wong, H.-K., Higonnet, T., 1971. Scheduling Algorithms for a dial-a-ride system. Massachusetts Institute of Technology. Urban Systems Laboratory.

Wolf, S., Merz, P., 2007. Evolutionary Local Search for the Super-Peer Selection Problem and the p-Hub Median Problem, in: Bartz-Beielstein, T., Blesa Aguilera, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (Éd.), Hybrid Metaheuristics. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 1‑15. https://doi.org/10.1007/978-3-540-75514-2_1

CHAPITRE 3

Une métaheuristique hybride pour résoudre le problème du transport à la demande avec véhicules privés et sommets alternatifs

Objectifs du chapitre :

Dans ce chapitre, une seconde méthode de résolution du problème de transport à la demande avec véhicules privés et sommets alternatifs (DARP-PV-AN) est proposée, basée sur une métaheuristique hybride.
Une justification du développement de cette seconde méthode est d'abord exposée, avant d'expliquer le fonctionnement global de l'algorithme hybride. Ensuite, chacune des trois sous-parties composant la méthode sont développées en détail : affectation des clients dans les véhicules par un algorithme génétique, fonction d'évaluation optimale des tournées par une approche de type *Branch and Bound* et enrichissement de la population en résolvant le problème de couverture par ensembles avec poids (*Weighted Set Cover Problem – WSCP*) par PLNE. La méthode est ensuite comparée à celle proposée dans le chapitre précédent.

3.1 Le problème des sommets alternatifs

Dans le problème du transport à la demande avec sommets alternatifs et véhicules privés (DARP-PV-AN), la gestion des sommets alternatifs pose problème. En effet, comme expliqué dans la conclusion du chapitre précédent, les sommets initiaux sont utilisés dans les méthodes de recherche locale et de mutation des tournées privées afin d'estimer le gain potentiel d'un mouvement sur la fonction objectif. Puisque les sommets alternatifs sont proches des sommets initiaux, le gain d'un mouvement calculé à partir des sommets initiaux est vérifié et validé lors de l'évaluation d'une tournée privée avec les sommets alternatifs.

Cependant, ces estimations peuvent empêcher la découverte de certaines solutions, comme cela est illustré par l'exemple de la *Figure 3.1*. En effet, en supposant que le temps de service à chaque sommet est de 1 unité de temps, l'estimation par les sommets initiaux refuserait un mouvement engendrant l'ordre de visite $(1^+, 2^+, 3^+)$ puisque le véhicule ne pourrait arriver avant la date 11 sur le sommet 2^+ (10 de temps de trajet et 1 de temps de service sur le sommet 1^+). Pourtant, la tournée est réalisable car elle passe par les sommets alternatifs : elle partirait du sommet 1.2^+ au temps 1, arriverait au sommet 2.2^+ au temps 9 et au sommet 3.1^+ au temps 14. Ainsi, l'estimation de la faisabilité d'une tournée en utilisant les sommets initiaux au lieu des sommets alternatifs peut éventuellement supprimer des solutions valides lors de l'exploration de l'espace des solutions par les méthodes de recherche locale/mutation.

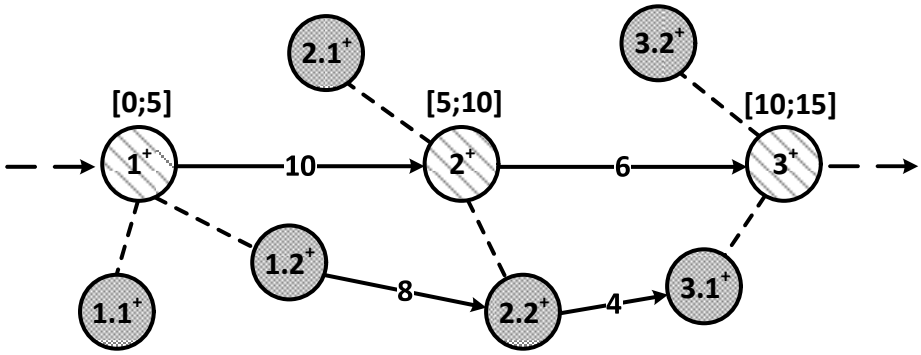


Figure 3.1 : Fausse estimation si les sommets alternatifs sont trop éloignés des sommets initiaux

En revanche, si les sommets alternatifs étaient utilisés pour prévoir l'intérêt d'un mouvement de recherche locale/mutation, le nombre de calculs augmenterait significativement : en effet, pour un ordre de visite donné, l'ensemble des tournées alternatives devraient être calculées afin de trouver la plus courte, tout en prenant en compte l'ensemble des contraintes du DARP-PV-AN. Or les méthodes de recherche locale/mutation font parties des fonctions les plus utilisées par l'algorithme ELS : ne pas se baser sur les sommets initiaux pour faire des estimations serait trop coûteux en terme de temps.

Une autre méthode a donc été imaginée afin de pallier à ce problème. Elle est formée par une hybridation entre un algorithme génétique, une méthode d'évaluation optimale des tournées de type « séparation et évaluation » (*Branch and Bound*) et une méthode d'enrichissement de la population par PLNE. La section suivante propose une présentation générale de la méthode de résolution.

3.2 Métaheuristique hybride proposée

L'algorithme proposé est une métaheuristique hybride de type *cluster-first route-second* (*Figure 3.2*). Ces méthodes de résolution ont été parmi les premières utilisées pour résoudre les problèmes de tournées de véhicules (Clarke et Wright, 1964) :

- La partie *cluster-first* consiste à affecter à chaque véhicule l'ensemble de ses visites ;
- La partie *route-second* détermine pour chaque véhicule l'ordre dans lequel il va effectuer ses visites.

Les métaheuristiques de type *route-first cluster-second* forment une autre approche des problèmes de tournées de véhicules : elles consistent à former en premier un tour géant en relaxant les contraintes de capacité des véhicules, puis à le découper ensuite en différentes tournées. D'abord proposé par (Newton et Thomas, 1974) dans le cadre du découpage d'un tour géant par un algorithme glouton pour le *bus routing problem*, la méthode a été utilisée à nouveau par (Beasley, 1983) sur le VRP en proposant un découpage optimal du tour géant basé sur le calcul d'un plus court chemin dans un graphe auxiliaire. Depuis, cette approche est utilisée dans de nombreuses variantes en offrant d'excellents résultats sur un grand nombre de problèmes de tournées de véhicules comme le CARP (Lacomme et al., 2004; Ulusoy, 1985), le VRP (Prins, 2004) ou encore le VRP avec fenêtres de temps (Vidal et al., 2013). Cependant, cette méthode de résolution atteint certaines limites lorsque le découpage du tour géant ne peut pas se faire optimalement de manière efficace. Aucune méthode de type *route-first cluster-second* n'a de ce fait été proposée actuellement pour le DARP. Ainsi une méthode de type *cluster-first route-second* est favorisée.

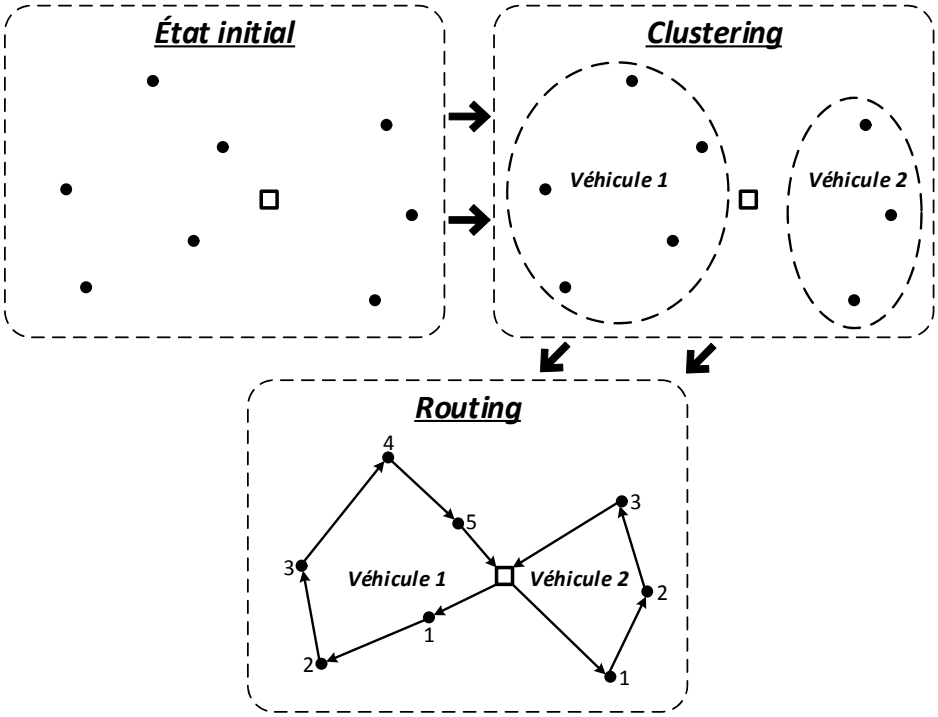


Figure 3.2 : Méthode de type cluster-first route-second (Bodin et Golden, 1981)

La méthode proposée (*Figure 3.3*) est basée sur l'hybridation d'un algorithme génétique avec une méthode d'évaluation des tournées de type « séparation et évaluation » (*Branch and Bound*) ainsi qu'un enrichissement de la population par programmation linéaire en nombres entiers (PLNE). L'algorithme se divise en 3 étapes principales :

- La première étape est effectuée par un algorithme génétique et est responsable de la partie *cluster-first* : il permet de déterminer les véhicules utilisés ainsi que les clients qui leurs sont affectés. La présentation des algorithmes évolutionnistes ainsi que leur implémentation pour le DARP-PV-AN est proposée dans la section 3.3 ;
- La deuxième étape correspond à la méthode d'évaluation des individus créés par l'algorithme génétique. Elle est effectuée par un algorithme de *Branch and Bound* et est responsable de la partie *route-second* : une fois les clients affectés aux véhicules par l'algorithme génétique, la méthode évalue l'ordre de visite optimal pour chaque véhicule en respectant l'ensemble des contraintes du DARP-PV-AN. Pour éviter de calculer plusieurs fois certaines tournées identiques, ces dernières sont sauvegardées. La méthode est décrite dans la section 3.4 ;
- La dernière étape consiste en l'enrichissement de la population de l'algorithme génétique en résolvant le problème de couverture par ensembles avec poids (*Weighted Set Cover Problem* – WSCP) par programmation linéaire en nombres entiers (PLNE). Cette résolution permet de créer une nouvelle solution en utilisant les tournées sauvegardées lors de l'évaluation par le *Branch and Bound*. L'individu représentant la nouvelle solution est alors inséré dans la population pour la génération suivante afin d'améliorer la vitesse de convergence et les solutions obtenues par l'algorithme. L'utilisation de la méthode d'enrichissement est un paramètre booléen de l'algorithme génétique : avec la valeur *true*, l'algorithme génétique utilise la méthode d'enrichissement, sinon il se déroule comme un algorithme génétique classique. Le WSCP ainsi que la méthode de résolution sont décrits dans la section 3.5.

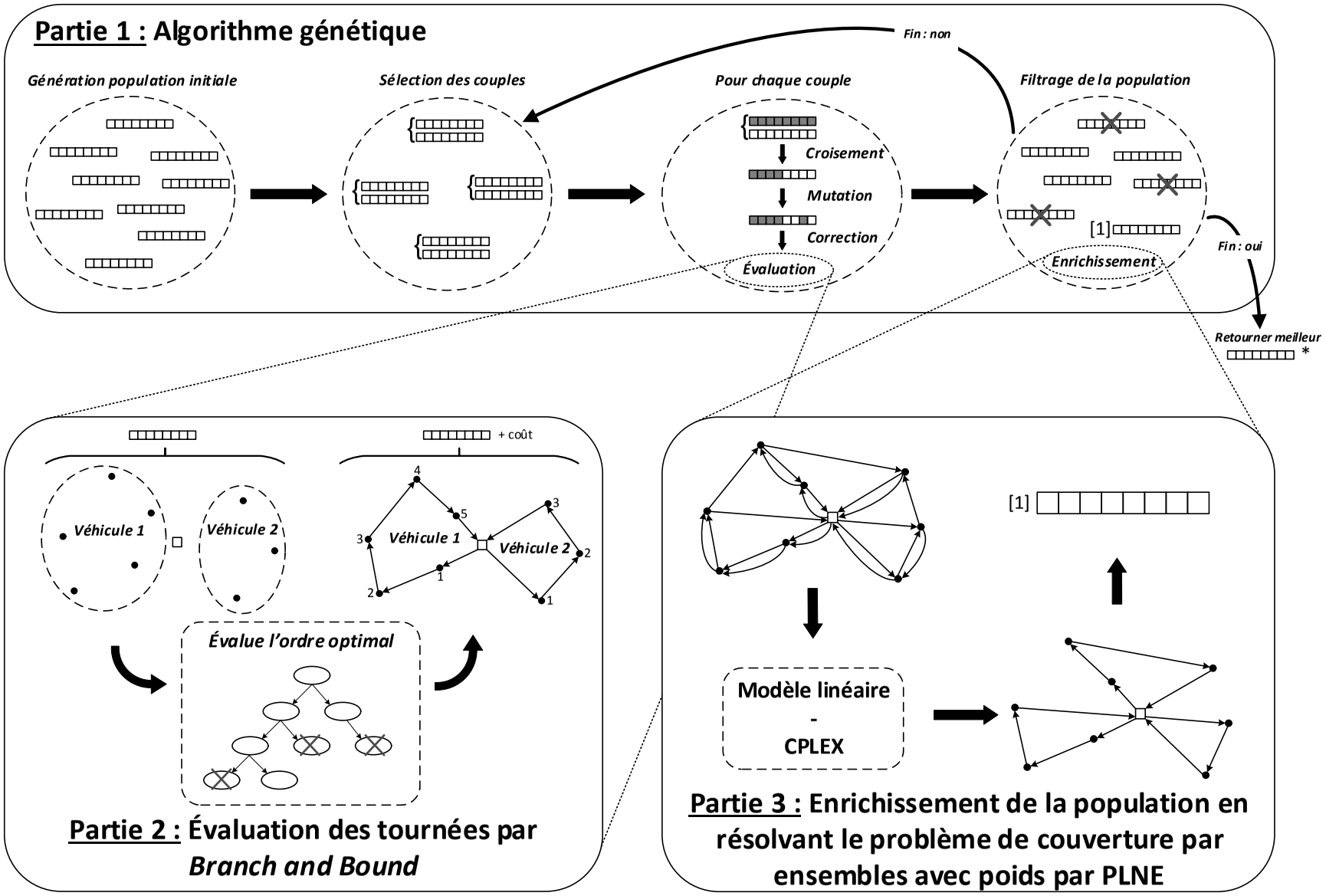


Figure 3.3 : Schéma global de la métaheuristique hybride proposée pour résoudre le DARP-PV-AN

3.3 Algorithme génétique

La première étape de l'algorithme hybride proposé est effectuée par un algorithme génétique qui appartient à la famille des algorithmes évolutionnistes. Ces derniers s'inspirent de la théorie de l'évolution pour résoudre des problèmes d'optimisation : le but étant d'améliorer itérativement des solutions existantes en les combinant avec d'autres solutions. Ces algorithmes sont basés sur des processus d'évolutions stochastiques et sont en ce sens non déterministes. Puisqu'ils sont utilisés pour résoudre des problèmes d'optimisation, mais ne donnent pas nécessairement la solution optimale, ils sont classés dans la catégorie des métaheuristiques.

Les premiers algorithmes évolutionnistes sont les stratégies d'évolution, proposées par (Rechenberg, 1965) pour résoudre les problèmes d'optimisation continue. Ensuite, les principes de la programmation génétique ont été proposés par (Fogel, 2009) dans les années 60 pour la conception d'automates à états finis. Enfin, les algorithmes génétiques, très utilisés aujourd'hui dans l'optimisation combinatoire, ont été proposés par (Holland, 1975) puis popularisés par (Goldberg, 1989) : ils sont utilisés dans ce chapitre dans la méthode hybride résolvant le DARP-PV-AN.

Comme expliqué précédemment, les algorithmes génétiques utilisent la notion de sélection naturelle appliquée à une population composée d'individus. Chaque individu est une solution potentielle du problème étudié s'il respecte l'ensemble des contraintes. Au cours de l'algorithme, les meilleurs individus « survivent » et transmettent leurs bonnes caractéristiques aux générations suivantes, améliorant ainsi la population génération après génération.

Un algorithme génétique classique se déroule comme illustré par les rectangles blancs sur la *Figure 3.4* : une première phase génère l'ensemble de la population initiale avant de répéter successivement les phases de sélection, croisement, mutation, évaluation et filtration jusqu'à une condition d'arrêt. Le fonctionnement de ces différentes étapes ainsi que leur implémentation dans le cadre de la méthode hybride proposée est détaillé dans cette section. Le rectangle gris représentant l'enrichissement de la population en résolvant le problème de couverture par ensembles (WSCP) par PLNE, est responsable de la partie hybride de la méthode de résolution proposée et est décrit section 3.5.

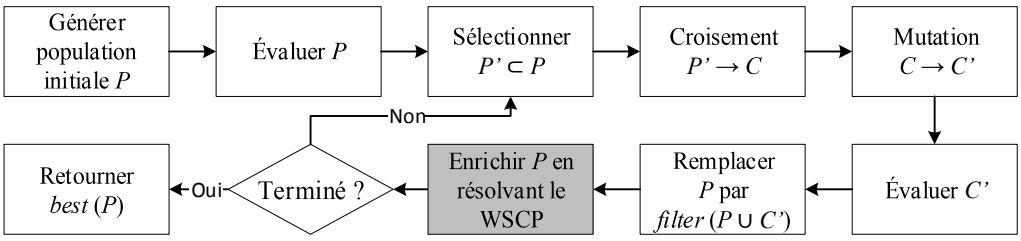


Figure 3.4 : Schéma global d'un algorithme génétique (rectangles blancs) avec P représentant les individus parents et C les individus enfants. Le rectangle gris représente la partie hybride de la méthode proposée

Avant d'expliquer le déroulement et l'implémentation de l'algorithme génétique, il est important de déterminer l'encodage des individus formant la population manipulée par ce dernier.

3.3.1 Encodage d'un individu

Un des facteurs les plus importants dans les algorithmes génétiques est la façon dont sont représentés les individus, ou encore, quelle structure de données représente leurs particularités. Pour cela, différentes méthodes existent et peuvent s'appliquer à différents problèmes.

La représentation en codage binaire permet d'encoder un individu sous la forme d'un tableau contenant uniquement des valeurs booléennes. Ainsi chaque case du tableau représente une caractéristique de l'individu qui peut prendre deux valeurs. Par exemple, le problème du sac à dos (*knapsack problem*) illustré sur la *Figure 3.5* est un problème d'optimisation consistant à sélectionner des objets en maximisant le gain tout en respectant la charge maximale. Pour ce problème étudié depuis le début des années 1900 (Dantzig, 1930; Karp, 1972) un encodage binaire est parfaitement adapté (Chu et Beasley, 1998).

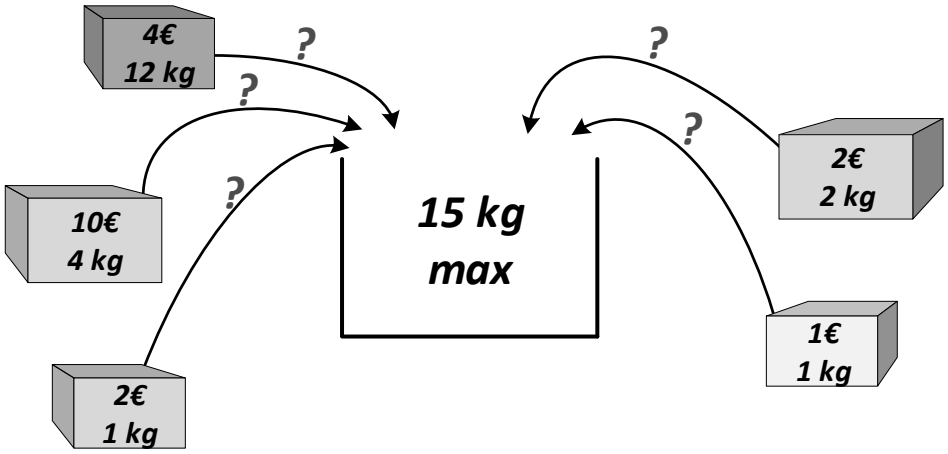


Figure 3.5 : Illustration du problème de sac à dos

Comme représenté sur la *Figure 3.6*, l'encodage d'un individu pour le problème de sac à dos utilise un tableau de valeurs booléennes : si un objet est sélectionné, la case correspondant à cet objet aura la valeur 1, sinon 0. L'individu représente une solution valide au problème si l'ensemble des objets sélectionnés ne dépassent pas la charge maximale.

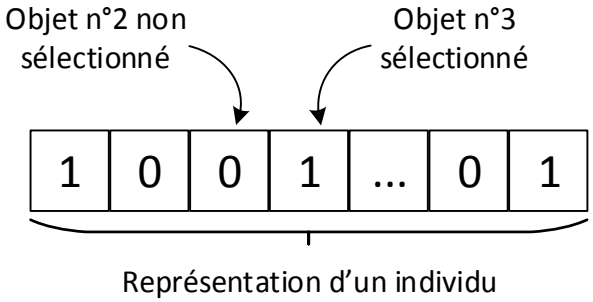


Figure 3.6 : Représentation d'un individu en codage binaire pour le problème du sac à dos

Une autre représentation concerne l'encodage à caractères multiples : le but est de stocker dans une structure de données des caractères différents des valeurs booléennes. Ce type d'encodage est souvent utilisé et est plus naturel que l'encodage précédent. Par exemple, pour le problème du voyageur de commerce (*Travelling Salesman Problem* – TSP) illustré sur la

Figure 3.7, un individu est encodé par un tableau d'entiers (Goldberg et Lingle, 1985) : chaque case contient un numéro de ville (sans duplicata) et l'ensemble des cases représente donc un ordre de visite. Avec cette méthode d'encodage pour le TSP, chaque individu est une solution valide car il représente un ordre de visite sur l'ensemble des villes.

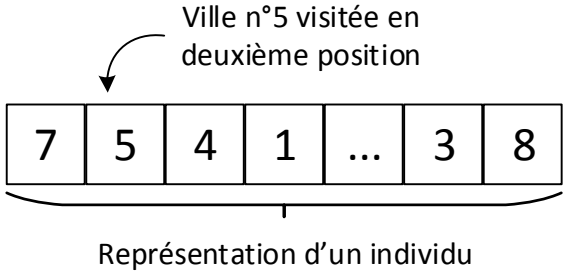


Figure 3.7 : Représentation d'un individu en codage à caractères multiples pour le TSP

Dans la méthode hybride de résolution du DARP-PV-AN proposée, l'algorithme génétique est responsable de la partie *clustering* : un individu représente l'affectation de chaque client $c \in C$ à un véhicule k par lequel il est transporté de son lieu de *pickup* à son lieu de *delivery*.

La structure de données représentant l'encodage d'un individu du DARP-PV-AN doit également permettre :

- L'accès au véhicule transportant le client i en $O(1)$;
- La modification du véhicule transportant le client i en $O(1)$;
- Le calcul d'une clé unique pour chaque individu en $O(n)$: deux individus identiques auront la même clé, alors que deux individus différents auront des clés différentes.

Pour représenter un individu encodant une solution potentielle du DARP-PV-AN, un vecteur de véhicules α est utilisé où $i \in \{0, \dots, n-1\}$ est l'indice du client concerné. Ainsi, l'accès au véhicule transportant un client et sa modification en utilisant un indice i peut être effectué en $O(1)$. Une représentation d'un individu est proposée sur la *Figure 3.8* : elle montre une instance contenant 7 clients indicés de 0 à 6, où les clients 0, 2 et 3 sont dans le véhicule V1 et les clients 1, 4, 5 et 6 sont dans le véhicule V2.

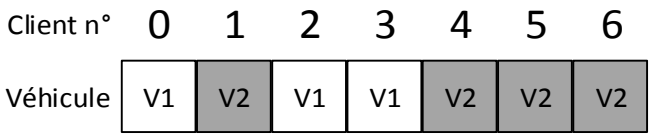


Figure 3.8 : Vecteur α représentant l'encodage d'un individu de 7 clients et 2 véhicules

Un défaut des métaheuristiques de type algorithme génétique est la convergence de la méthode vers des minima locaux. Cette convergence est due à l'exploration d'une même région de l'espace des solutions, notamment lorsque des individus identiques apparaissent dans la population au fur et à mesure des générations. Dans un algorithme génétique, les nombreuses caractéristiques communes entre les solutions valides augmentent ce risque, même si la génération de nouveaux individus est effectuée avec des méthodes utilisant des processus aléatoires.

De plus, l'ordre de visite des sommets dans une tournée est évalué optimalement (section 3.4), mais représente une quantité de calcul important. Comme toutes les tournées calculées sont sauvegardées, il est nécessaire de détecter les tournées déjà évaluées pour éviter de refaire certains calculs.

Ainsi, un système de détection des individus et tournées identiques a été mis en place, basé sur les techniques de hachage. Chaque nouvelle génération de l'algorithme génétique ne devant pas contenir d'individus identiques, des clés sous forme d'une chaine de caractères sont associées à chaque individu (*Algorithme 5*). De manière identique, une clé est associée à chaque tournée rencontrée pour éviter de calculer à nouveau son ordre de visite.

Algorithme 5 key_tour

Input parameter
t : Tour that needs a key

Output variable
key : Key computed

Local variable
$clientsList$: Array that contains clients' id

Begin
1 key := "|"
2 s
3 **for all** $client$ **in** t.get_clients() **do**
4 | $clientList$.add($client$.get_ID())
5 **end for**
6 sort($clientList$)
7 **for all** $clientID$ **in** $clientList$ **do**
8 | key := key + $clientID$ + "|"
9 **end for**
10 **if** t.is_public() **then**
11 | key := key + "pub"
12 **else**
13 | key := key + t.get_Vehicle().get_ID()
14 **end if**
15
16 **return** { key }
End

Pour chaque tournée, la clé est une chaine de caractères composée des identifiants des clients servis par le véhicule, triés par ordre croissant et séparés par le caractère « | ». Le type de véhicule est ajouté à la fin. Par exemple, si le véhicule privé n°5 visite les clients n° 2, 10 et 14, la clé correspondante serait « |2|10|14|pri5 ». Les véhicules privés étant différents, il est important de les faire apparaître dans la clé : des groupes identiques de clients desservis par des véhicules privés différents ne donnent pas des tournées identiques. En revanche, les véhicules publics ne sont pas différents : la présence de leur identifiant est donc inutile dans le calcul de la clé.

Pour un individu, l'algorithme key_ind calculant la clé correspond simplement à la concaténation des clés des tournées par ordre croissant. Les clés calculées sont ensuite insérées dans des structures de données de type *HashSet* ou *HashMap* pour savoir en $O(1)$ si une tournée ou un individu a déjà été rencontré et évalué.

3.3.2 Implémentation de l'algorithme génétique pour le DARP-PV-AN

Un schéma global représentant l'implémentation de l'algorithme génétique est proposé sur l'*Algorithme 6*. Ce dernier est composé de 8 étapes principales. La première est la génération de la population initiale et est décrit dans la section 3.3.3. Le fonctionnement et l'implémentation des méthodes de sélection, croisement, mutation, correction et filtrage sont expliqués respectivement dans les sections 3.3.4, 3.3.5, 3.3.6, 3.3.7, 3.3.10. L'étape 6 représentant l'évaluation d'un individu est expliquée dans la section 3.4. Enfin l'enrichissement de la population en résolvant le problème de couverture par ensembles avec poids (WSCP) par PLNE est représenté par l'étape 7 et est décrit dans la section 3.5. Dans l'algorithme génétique, il est important de distinguer les individus « parents » représentés par les individus de la génération précédente, des individus « enfants » représentant les individus créés lors de la génération courante.

Algorithme 6 $genetic\_algorithm$

Input parameters

t	:	Time limit of the algorithm
m	:	Mutation's probability
$sizePop$	:	Number of individuals in the population

Output variable

$S*$	:	The best solution found

Local variables

$parents$	:	Array of individuals
$firstParent$	:	First parent for crossover
$secondParent$	:	Second parent for crossover
$child$	:	Child created by crossover
$children$	:	Array of current generation's individuals
$elapsedTime$	:	Time since algorithm's launch

Begin

1 **// Step 1. Generate initial population**
2 $parents$:= generate_initial_population($sizePop$)
3 **while** $elapsedTime < t$ **do**
4 | **for** i := 0 **to** $sizePop - 1$ **do**
5 | | **// Step 2. Parents selection before crossover**
6 | | $\{firstParent, secondParent\}$:= bin_tournament_selection($parent$)
7 | | **// Step 3. Child creation by crossover**
8 | | $child$:= one_cut_crossover($firstParent, secondParent$)
9 | | **// Step 4. Mutation on child**
10 | | $child$:= mutation($child, m$)
11 | | **// Step 5. Keep private vehicles or not (correction)**
12 | | private_vehicle_keeping($child$)
13 | | **// Step 6. Individual's evaluation (described section 3.4)**
14 | | evaluation($child$)
15 | | $children$.add($child$)
16 | **end for**
17 | **// Step 7. CPLEX tour selection (described section 3.5)**
18 | launch_cplex_tour_selection($children$)
19 | **// Step 8. Selection for next iteration**
20 | next_iteration_selection($parents, children$)
21 | $S*$:= best_solution($S*$,$parents$)
22 **end while**
23 **return** { $S*$ }

End

3.3.3 Génération de la population initiale

Dans cette première étape, une population initiale doit être créée. Elle représente un ensemble d'individus pour le problème étudié. Ces individus peuvent représenter des solutions valides (respecter l'ensemble des contraintes du problème) ou non. La génération des individus formant la population initiale peut être aléatoire, utiliser un algorithme glouton, ou bien des méthodes de recherche locale. Le but principal étant de former un ensemble d'individus hétérogène afin de fournir une population initiale avec une bonne diversité. Cette diversité génétique devra être maintenue durant l'algorithme afin d'éviter une convergence prématurée.

Dans ce chapitre, les individus sont générés aléatoirement et ne représentent pas nécessairement des solutions valides. En revanche, la population initiale ne contient pas d'individus identiques. La fonction générant la population initiale est décrite dans l'*Algorithme 7* : elle consiste en l'appel de la fonction $generate\_individual$ jusqu'à avoir généré n individus différents. La comparaison des individus générés peut être effectuée rapidement grâce à la fonction de hachage.

Algorithme 7 $generate\_initial\_population$

Input parameter
n : Number of individuals to generate

Output variable
$population$: Array of generated individuals of size n

Local variables
$individualsCreated$: HashSet containing all individuals' created keys
$individualKey$: Individual's key
i : Number of different individuals generated
$newIndividual$: Individual generated

```
Begin
1   i := 0
2
3   while i < n do
4   |   newIndividual := generate_individual()
5   |   individualKey := key_ind(newIndividual)
6   |
7   |   if not individualsCreated.contains(individualKey) then
8   |   |   individualsCreated.add(individualKey)
9   |   |   evaluation(newIndividual)
10  |   |   population.add(newIndividual)
11  |   |   i := i + 1
12  |   end if
13  end while
14
15  return { population }
End
```

La génération d'un nouvel individu par la fonction $generate\_individual$ est illustrée par l'*Algorithme 8*. Ce dernier se divise en trois parties principales :

- La première consiste en la sélection d'un ensemble de véhicules pouvant être utilisés dans l'individu généré. Cette pré-sélection des véhicules utilisables est nécessaire avant chaque génération d'individu afin d'augmenter la diversité : sans elle, la plupart des véhicules apparaitraient dans les individus, les rendant trop similaires entre eux ;

- La deuxième partie affecte à chaque client, un véhicule parmi ceux sélectionnés dans la partie 1 ;
- La dernière partie consiste en l'affectation dans les véhicules privés, des clients les conduisant, ce qui n'était pas garanti par l'étape 2. Chaque véhicule privé utilisé transporte donc le client le possédant.

Algorithme 8 $generate\_individual$

Output variable
$individual$: Individual generated, with $individual.clientAffectation$ an array of $vehicles$

Local variables
$vehiclesAvailable$: Array containing all vehicles that can be used in the generated individual
$random$: Generator of random values

Global variables
$allVehicles$: Array containing all vehicles
n : Number of clients

```
Begin
1   // Step 1. Creation of a set of vehicles that can be used in the generated individual
2   while vehiclesAvailable.size() = 0 do
3   |   for all vehicle in allVehicles do
4   |   |   if random.next_boolean() then
5   |   |   |   vehiclesAvailable.add(vehicle)
6   |   |   end if
7   |   end for
8   end while
9
10  // Step 2. Associates clients with authorized vehicles
11  for i := 0 to n − 1 do
12  |   individual.clientAffectation[i] :=
13  |                   vehiclesAvailable.get(random.next_int(vehiclesAvailable.size()))
14  end for
15
16  // Step 3. Move a client in its private vehicle if used
17  for all vehicle in vehiclesAvailable do
18  |   if vehicle.is_private() and vehicle.is_used_in(individual) then
19  |   |   individual.clientAffectation[vehicle.get_owner()] := vehicle
20  |   end if
21  end for
22
23  return { individual }
End
```

3.3.4 Sélection des couples

L'opérateur de sélection consiste à sélectionner les individus qui vont se reproduire, parmi la population. Issue du processus de sélection naturelle, cette méthode applique le principe suivant : les individus les plus adaptés à leur environnement sont plus enclins à être choisis tandis que les autres meurent avant d'avoir pu se reproduire. Ainsi, l'adaptation des individus, et donc la qualité des solutions, augmente de génération en génération.

Plusieurs techniques de sélection existent :

- La sélection aléatoire permet à chaque individu d'être sélectionné avec une probabilité identique ;

- La sélection par rang consiste à classer les individus selon leur score, du meilleur au plus faible. La méthode va ensuite sélectionner les k meilleurs individus pour les phases suivantes, selon une probabilité dépendant de leur rang. Ainsi, plus un individu a un rang élevé, plus il a de chances d'être sélectionné ;
- La sélection par tournoi consiste à tirer aléatoirement k paires d'individus et pour chacune, de sélectionner celui ayant le score le plus élevé ;
- La sélection par roulette donne à chaque individu une probabilité d'être sélectionnée proportionnelle à sa valeur sur l'objectif optimisé. On utilise ensuite un principe de roue sur laquelle chaque individu est représenté : plus la probabilité d'un individu d'être sélectionné est grande, plus sa représentation sur la roue est importante. Un tirage aléatoire est ensuite répété k fois sur la roue.

Dans l'algorithme proposé, la méthode de sélection par tournoi est utilisée (*Figure 3.9*) : deux individus différents sont choisis aléatoirement dans la population de parents et celui ayant la meilleure valeur de la fonction objectif est sélectionné comme premier parent reproducteur. La fonction est ensuite relancée pour sélectionner un deuxième parent différent du premier. Ainsi, deux individus différents et de bonne qualité sont disponibles pour l'étape suivante.

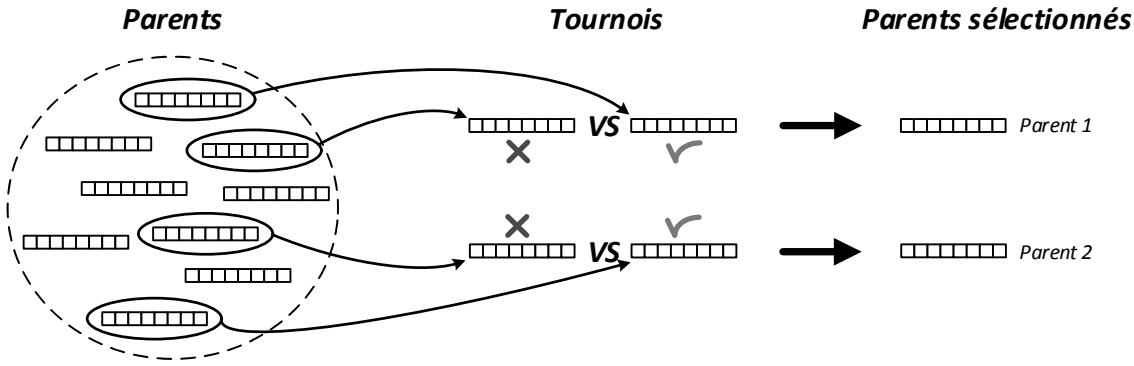


Figure 3.9 : Méthode de sélection des parents par tournoi

3.3.5 Croisement

L'opérateur de croisement permet à chaque couple d'individus (parents) sélectionnés par l'opérateur de sélection de former un ou plusieurs nouveaux individus (enfants) en mélangeant leurs caractéristiques. Lors de cette opération, les deux individus parents dupliquent leur matériel génétique et les combinent pour créer un ou plusieurs individus enfants. Les chromosomes des parents se coupent et se rassemble en un ou plusieurs points appelés points de croisement. L'exemple de la *Figure 3.10* montre une opération de croisement avec un seul point : deux enfants sont créés et ajoutés dans la population C.

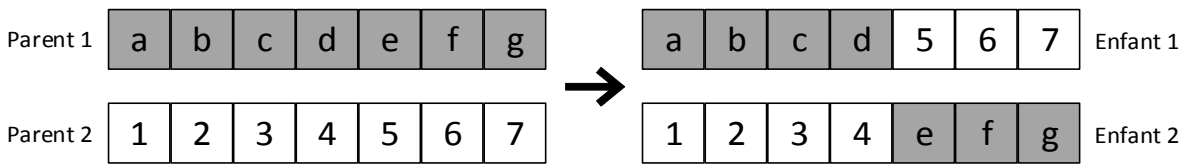


Figure 3.10 : Opération de croisement avec un point de coupe

La méthode proposée (*Figure 3.11*) utilise un point de croisement ϕ choisi aléatoirement dans l'intervalle $[0; n-1[$. Les parents étant définis par les vecteurs α^{P_1} pour le parent 1 et α^{P_2} pour le parent 2, le vecteur enfant α^E représentant le nouvel individu est modélisé par :

$$\alpha^E = (\alpha^{P_1}_{0 \le i \le \phi} , \alpha^{P_2}_{\phi < j < n})$$

L'individu α^E généré peut engendrer une violation des contraintes dans la partie *clustering*. Si un véhicule privé est utilisé, le client le possédant doit être affecté dans ce dernier. Or l'individu généré par point de croisement peut contenir un véhicule privé sans que ce dernier ne transporte le client le possédant. Une correction de l'individu est donc nécessaire pour soit supprimer le véhicule privé et réaffecter sa liste de clients, soit affecter le client possédant le véhicule privé dans ce dernier. Le fonctionnement de cette méthode de correction est expliqué dans la section 3.3.7.

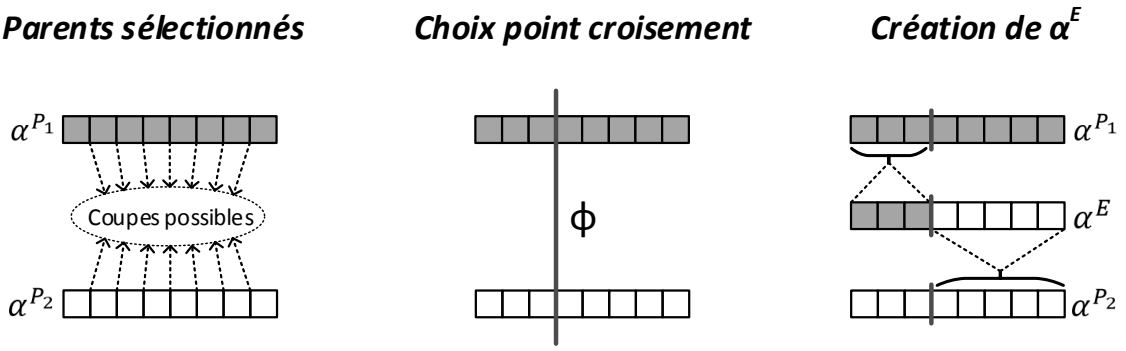


Figure 3.11 : Méthode de croisement avec un point de coupe

3.3.6 Mutation

L'opérateur de mutation agit sur les enfants générés lors de la phase de croisement. Elle consiste en une probabilité pour chaque gène de chaque enfant de changer de valeur aléatoirement donnant ainsi un individu modifié (*Figure 3.12*). Il est donc possible qu'un enfant ne subisse aucune modification ou bien plusieurs. Cet opérateur est très important pour éviter une convergence prématurée de l'algorithme vers des optima locaux.

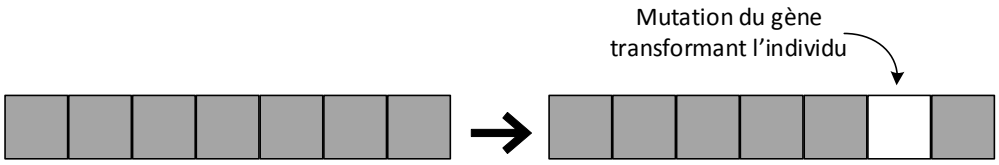


Figure 3.12 : Mutation d'un individu

Dans l'algorithme proposé, plusieurs mutations ont été étudiées :

- *random_move* : faible probabilité pour chaque client d'être déplacé dans un véhicule aléatoire ;
- *relocate* : suppression d'une tournée et réallocation des clients la composant dans les tournées restantes ;
- *exchange* : échange de deux clients desservis par des tournées différentes ;

a) Mouvement aléatoire

La première mutation consiste à appliquer pour chaque client, avec une faible probabilité, un changement de véhicule. Si un client change de véhicule, alors le nouveau est choisi aléatoirement parmi K^T. Il est donc possible de faire apparaître de nouveaux véhicules dans les individus ayant subi une mutation.

Algorithme 9 $random\_move$

Input parameters
- $individual$: Individual generated by the crossover
- $mutProb$: Mutation probability

Local variable
- $random$: Generator of random values

Global variables
- $allVehicles$: Array containing all vehicles
- n : Number of clients

Begin
1 **for** i := 0 **to** $n - 1$ **do**
2 | **if** $random$.next_int($mutProb$) $= 0$ **then**
3 | | $individual.clientAffectation[i]$:= $allVehicles$[$random$.next_int($allVehicles$.size())]
4 | **end if**
5 **end for**
End

b) Réallocation

La seconde mutation permet de supprimer une tournée dans l'individu et de réaffecter l'ensemble des clients dans les autres tournées restantes.

Algorithme 10 $relocate$

Input parameter
- $individual$: Individual generated by the crossover, with $individual.nbTour$ the number of tour in the individual

Local variables
- $random$: Generator of random values
- $removedVehicle$: Vehicle removed from the individual
- $vehiclesUsed$: Array containing all vehicles used in the individual

Global variable
- n : Number of clients

Begin
1 **if** $individual.nbTour > 1$ **then**
2 | $removedVehicle$:= $vehiclesUsed$[$random$.next_int($vehiclesUsed$.size())]
3 | $vehiclesUsed$.remove($removedVehicle$)
4 | **for** i := 0 **to** $n - 1$ **do**
5 | | **if** $individual.clientAffectation[i] = removedVehicle$ **then**
6 | | | $individual.clientAffectation[i]$
7 | | | := $vehiclesUsed$[$random$.next_int($vehiclesUsed$.size())]
8 | | **end if**
9 | **end for**
10 **end if**
End

c) Échange

La dernière fonction de mutation choisi aléatoirement deux clients différents et échange le véhicule dans lequel ils sont affectés. Il est possible que les clients sélectionnés soient dans le même véhicule, auquel cas la fonction n'applique pas de modifications à l'individu.

Algorithme 11 $exchange$

Input parameter
$individual$: Individual generated by the crossover

Local variables
$random$: Generator of random values
$pos1$: Position of first client
$pos2$: Position of second client
$temp$: Temporary variable for vehicle

Global variable
n : Number of clients

Begin
1 $pos1$:= $random$.next_int(n)
2 $pos2$:= $random$.next_int(n)
3 **while** $pos1 = pos2$ **do**
4 | $pos2$:= $random$.next_int(n)
5 **end while**
6 $temp$:= $individual.clientAffectation[pos1]$
7 $individual.clientAffectation[pos1]$:= $individual.clientAffectation[pos2]$
8 $individual.clientAffectation[pos2]$:= $temp$
End

d) Sélection de la mutation

Lors de la mutation d'un individu, l'une des trois méthodes est choisie aléatoirement puis appliquée. Certaines mutations engendrent une modification (réallocation, échange, déplacement du pire client) alors que la mutation « mouvement aléatoire » peut laisser l'individu sans changements. Puisqu'une mutation ne doit pas faire systématiquement de modifications (afin de ne pas tomber dans une recherche aléatoire), la mutation « mouvement aléatoire » à une forte probabilité d'être appelée (98%). Les mutations « réallocation » et « échange » ont quant à elles une probabilité plus faible (1%). La méthode de sélection est décrite dans l'*Algorithme 12*.

Algorithme 12 $mutation\_selection$

Input parameter
$individual$: Individual generated by the crossover

Local variables
$random$: Generator of random values
$value$: Random value generated by $random$

Begin
1 $value$:= $random$.next_int(100)
2 **switch** $value$
3 | **case** 98 : $relocate(individual)$
4 | **case** 99 : $exchange(individual)$
5 | **default** : $random\_move(individual)$
6 **end switch**
End

3.3.7 Correction

Comme souligné dans la section 3.3.5, les individus générés par croisement (mais également par mutation) peuvent violer certaines contraintes propres à la partie *clustering* : un véhicule privé peut être utilisé mais le client le possédant n'est pas affecté dans ce même véhicule. Une correction effectuée par la fonction $private\_vehicle\_keeping$ est donc nécessaire pour éviter de générer des individus représentant des solutions non valides. Un choix aléatoire est effectué entre deux possibilités :

- Déplacer le client dans son propre véhicule s'il est utilisé ;
- Supprimer le véhicule privé et réaffecter aléatoirement l'ensemble des clients y étant affecté dans d'autres véhicules.

Comme illustré par l'exemple de la *Figure 3.13*, le client 4 n'est pas affecté dans son véhicule qui est pourtant utilisé pour transporter les clients 5 et 6. Deux possibilités sont disponibles : déplacer le client 4 dans son véhicule ou bien supprimer le véhicule du client 4 et réaffecter les clients 5 et 6 dans d'autres véhicules.

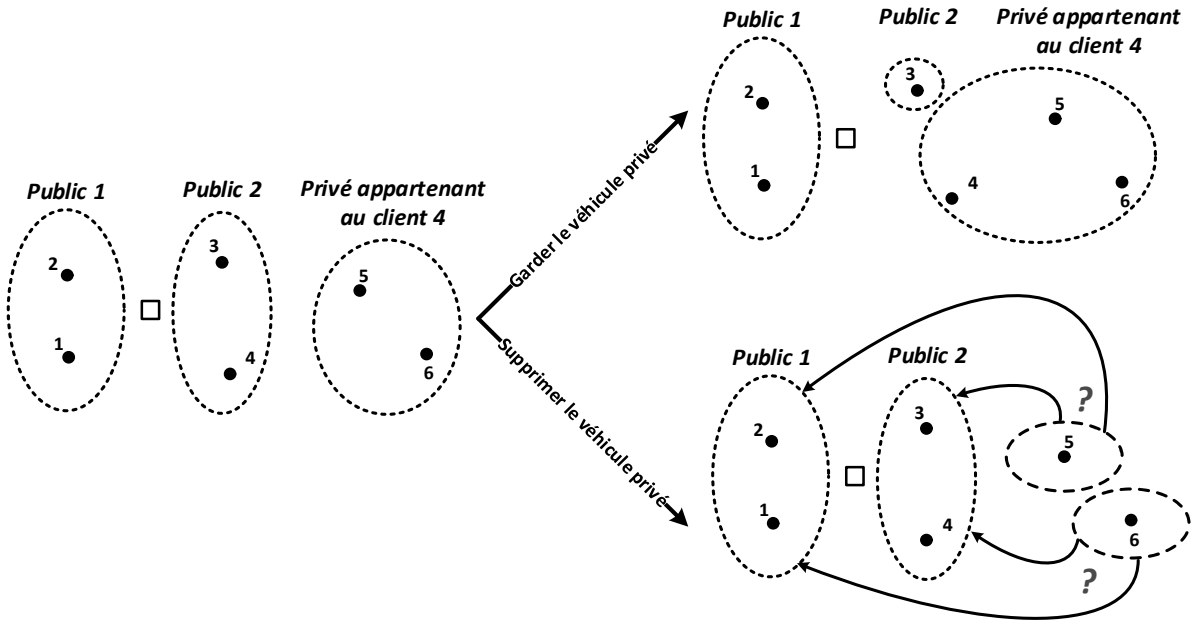


Figure 3.13 : Les deux méthodes de correction d'un clustering *lors d'une erreur sur les véhicules privés*

3.3.8 Évaluation

Cet opérateur consiste en la vérification du respect des contraintes et en l'affectation d'une valeur pour chaque individu. Ceux respectant les contraintes représentent des solutions valides et parmi ces derniers, les individus ayant une meilleure valeur que les autres sont considérés comme des meilleures solutions. Dans la plupart des algorithmes d'optimisation, la valeur attribuée à un individu correspond à celle de la fonction objectif : dans le TSP, la valeur correspond à la distance parcourue lors du trajet, alors que dans le problème du sac à dos, elle correspond au coût.

L'évaluation d'un individu représentant une solution potentielle du DARP-PV-PN consiste à évaluer chaque tournée le composant. Elle est effectuée par une méthode de *Branch and*

Bound appliquée sur chaque tournée : cette méthode vérifie si un ordre de visite respectant l'ensemble des contraintes est possible et, si tel est le cas, détermine l'ordre de visite optimal des clients. Un coût peut ensuite être affecté à chaque tournée et donc à l'individu. Le fonctionnement de la méthode est exposé dans la section 3.4.

3.3.9 Enrichissement de la population

Comme expliqué précédemment, les tournées optimales calculées par la procédure de *Branch and Bound* sont sauvegardées afin d'éviter de recalculer des ordres de visites déjà rencontrés. Il est donc possible d'utiliser cet ensemble de tournées pour sélectionner les meilleures et les rassembler au sein d'une solution qui n'a pas encore été trouvée par l'algorithme génétique.

Cette sélection des meilleures tournées revient à résoudre le problème de couverture par ensembles avec poids (WSCP). Pour ce faire, un modèle linéaire résolu par le solveur CPLEX est utilisé. La présentation du WSCP et du modèle est décrit section 3.5.

Ainsi, à chaque fin de génération, une nouvelle solution est créée et l'individu la représentant est inséré dans la population courante. L'ajout de cet individu, correspond à la partie hybride de la méthode de résolution du DARP-PV-AN et améliore la vitesse de convergence de l'algorithme ainsi que la qualité des solutions optimales tournées.

3.3.10 Filtrage de la population

La dernière étape lors de la création d'une génération dans l'algorithme génétique, consiste à fusionner l'ensemble des parents avec celui des enfants. L'ensemble créé est ensuite filtré en supprimant des individus pour avoir un ensemble final de taille identique à l'ensemble des parents initial. Ce nouvel ensemble filtré devient ensuite la population de parents pour la génération suivante.

La sélection des individus formant ce nouvel ensemble ne doit pas s'effectuer uniquement sur leur coût, cela entraînant une convergence trop rapide des algorithmes génétiques vers des optima locaux. Il est donc nécessaire de former la nouvelle population en sélectionnant les meilleurs individus mais également quelques individus aléatoires.

Le fonctionnement de la méthode de filtrage est décrit dans l'*Algorithme 13*. La première étape consiste à vider l'ensemble des individus du tableau $parents$ dans le tableau $children$, puis à trier les individus du tableau $children$ par ordre croissant de la valeur de leur fonction objectif. Ainsi les meilleurs individus seront en premier et les plus mauvais en dernier.

La deuxième étape consiste à supprimer les individus en double afin d'éviter un appauvrissement général de la population. Deux individus identiques ne se suivent pas nécessairement dans le tableau trié, puisque deux individus avec la même fonction objectif peuvent être différents : les individus identiques ne sont donc pas obligatoirement situés à des positions adjacentes dans le tableau $children$. Il est donc nécessaire de parcourir pour un individu situé à la position i, les $n - i$ cases restantes du tableau dans le pire des cas, soit une complexité en $O(n^2)$.

La troisième étape sélectionne les z premiers individus stockés dans $children$, z correspondant à $sizePop$ divisée par 3, et les insère dans le tableau $parents$ qui contiendra les individus pour la génération suivante.

Enfin, la dernière étape complète le tableau $parents$, afin que ce dernier ai une taille identique à $sizePop$, avec des individus choisis aléatoirement dans le tableau $children$.

Algorithme 13 $next\_iteration\_selection$

Input parameters

$parents$: Array of individuals that were used to create children
$children$: Array of individuals created by individuals in $parents$

Local variables

$random$: Generator of random values
$randPos$: Randomly selected position
$it1$: First iterator on $children$
$it2$: Second iterator on $children$
$indKey$: Temporary variable saving individual key

Global variable

$sizePop$: Size of the population

Begin

```
1   // Step 1. Put all individuals in children and sort them by ascending order of their cost
2   children.add_all(parents)
3   parents.clear()
4   sort(children)
5
6   // Step 2. Remove identical individuals
7   it1 := 0
8   while it1 < children.size() −1 do
9   |
10  |   indKey := key_ind(children[it1])
11  |   it2 := it1 + 1
12  |
13  |   while it2 < children.size() do
14  |   |   |   if indKey = key_ind(children[it2]) then
15  |   |   |   |   children.remove(it2)
16  |   |   |   else
17  |   |   |   |   it2 := it2 + 1
18  |   |   |   end if
19  |   end while
20  |
21  |   it := it + 1
22  |
23  end while
24
25  // Step 3. Keep the sizePop/3 first individuals of children
26  for i := 0 to sizePop/3 − 1 do
27  |   parents.add(children[0])
28  |   children.remove(0)
29  end for
30
31  // Step 4. Randomly selects remaining individuals
32  for i := 0 to sizePop ∗ 2/3 − 1 do
33  |   randPos := random.next_int(children.size())
34  |   parents.add(children[randPos])
35  |   children.remove(randPos)
36  end for
37
38  children.clear()
```

End

3.4 Évaluation des tournées par *Branch and Bound*

La méthode d'évaluation des tournées appelée dans l'algorithme génétique concerne la partie *route-second* des méthodes de type *cluster-first route-second* et est résolue par une méthode de type *Branch and Bound.* Cette évaluation s'applique à chaque individu créé par l'algorithme génétique de la façon suivante : pour chaque groupe de clients affectés à un véhicule, l'ordre de visite optimal respectant l'ensemble des contraintes du DARP-PV-AN est calculé. Les dates d'arrivée et de départ sont également affectées à chaque sommet visité. L'objectif à minimiser étant la distance parcourue par l'ensemble des tournées.

L'évaluation d'un individu S est illustrée par l'*Algorithme 14*. Elle consiste tout d'abord, pour chaque affectation de clients à un véhicule dans S, à vérifier dans la *HashMap* si la tournée correspondant a précédemment été calculée (ligne 5). Dans ce cas, il est inutile de lancer la méthode d'évaluation car la distance de la tournée précédemment calculée est utilisée, une distance arbitrairement grande représentant une tournée invalide (ligne 6). En revanche, si l'affectation est nouvelle, alors la méthode d'évaluation est lancée (ligne 8) calculant l'ordre de visite optimal respectant l'ensemble des contraintes. Le coût de l'individu S est ensuite affecté comme la somme de l'ensemble des tournées calculées, une valeur arbitrairement grande représentant une solution invalide.

Algorithme 14 $evaluation$

Input parameter

S : Individual that will be evaluated, with $S.tours$ an array of tour

Local variables

$sumDistance$: Objective function value
$allToursValid$: Boolean value saying if all tour are valid
$tourKey$: Key value of a given tour

Global variables

MAX_VALUE : Big M value
$toursCreated$: Map assigning to each computed tour key its created tour
$precedenceGraph$: Computed precedence graph necessary for evaluation of each tour

Begin

```
1   sumDistance := 0
2   allToursValid := true
3   for all tour in S.tours do
4   |   tourKey := key_tour(tour)
5   |   if toursCreated.contains(tourKey) then
6   |   |   tour := toursCreated.get(tourKey)
7   |   else
8   |   |   evaluation_tour(tour, precedenceGraph)
9   |   |   if not tour.is_valid() then
10  |   |   |   tour.cost := MAX_VALUE
11  |   |   end if
12  |   |   toursCreated.put(tourKey, tour)
13  |   end if
14  |   sumDistance := sumDistance + tour.cost
15  |   allToursValid := allToursValid and tour.is_valid()
16  end for
17  if not allToursValid then
18  |   sumDistance := MAX_VALUE
19  end if
20  S.cost := sumDistance
End
```

Les sections suivantes abordent en détail les différentes étapes de l'algorithme de *Branch and Bound*. Tout d'abord, l'encodage d'une tournée est décrit ainsi que le calcul d'un graphe de précédence utilisé pour des coupes lors du parcours de l'arbre. L'arbre de recherche utilisé par l'algorithme de *Branch and Bound* est ensuite définit avec notamment la description des nœuds le constituant. Enfin, un algorithme de parcours en profondeur de l'arbre est proposé ainsi que les différentes coupes permettant un élagage efficace, accélérant ainsi la vitesse de résolution.

3.4.1 Encodage d'une tournée

Lorsque l'affectation des clients dans les véhicules a été effectuée par l'algorithme génétique, une méthode de *Branch and Bound* est lancée pour chaque affectation non rencontrée précédemment, calculant son ordre de visite optimal en respectant l'ensemble des contraintes du DARP-PV-AN. Dans l'algorithme de *Branch and Bound*, une tournée est un ordre de visite représenté par un vecteur de sommets λ de taille l où $\forall i \in \{0, \dots, l-1\}$, λ_i représente un sommet de *pickup* ou de *delivery* associé à un client (ou bien le sommet du dépôt).

Dans le DARP-PV-AN, un client peut être servi par un véhicule public sur son sommet initial ou par un véhicule privé sur l'un de ses sommets alternatifs. Un sommet est décrit par un couple $(x.y)$, x représentant le client et y le sommet utilisé, avec $y = 0$ le sommet initial et $y > 0$ un sommet alternatif. Le couple « 0.0 » représentant le dépôt.

La *Figure 3.14* représente deux vecteurs, le premier étant une tournée publique et le second une tournée privée.

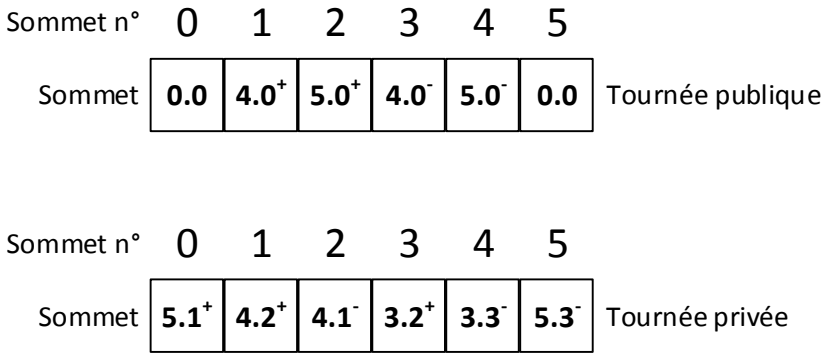


Figure 3.14 : Représentation de deux tournées, la première étant publique et la seconde privée

Une autre information importante est la date de début de service du véhicule affecté à chaque sommet de la tournée. Cette information est calculée lors de l'algorithme de *Branch and Bound* pour vérifier le respect de l'ensemble des contraintes, mais son stockage n'est pas nécessaire une fois l'ordre optimal calculé : lorsque la méthode hybride proposée fournit la meilleure solution trouvée pour une instance, les dates de début de service peuvent être recalculées en évaluant chaque tournée avec l'algorithme de (Cordeau et Laporte, 2003) en $O(n^2)$. En revanche, si les dates de début de service sont stockées pour chaque tournée, la consommation mémoire augmente mais également le temps de calcul nécessitant des copies potentielles lors de l'algorithme génétique.

L'encodage d'une tournée optimale fournie par l'algorithme de *Branch and Bound* comporte donc les sommets sélectionnés ainsi que leur ordre de visite, mais également le coût associé à la tournée, un coût arbitrairement grand représentant une tournée non valide.

3.4.2 Graphe de précédence

Le graphe de précédence est une structure de donnée calculée avant le début de résolution de chaque instance (avant le lancement de l'algorithme génétique) et utilisée lors de l'algorithme de *Branch and Bound* permettant des coupes lors de l'énumération des différentes positions d'insertion dans l'arbre de recherche.

Dans une tournée λ, les fenêtres de temps associées aux sommets de *pickup* et *delivery* peuvent imposer un ordre pour le passage du véhicule. Plus particulièrement, la présence de relation de précédence pour chaque couple de sommet et chaque couple de client est étudiée. Ainsi, si pour un ensemble donné un ordre est imposé, alors une contrainte de précédence est présente entre les deux éléments.

Pour représenter cet ensemble de précédences entre les clients, un graphe orienté $G = (V, A)$ est utilisé où V est l'ensemble des sommets représentant les *pickups* et *deliveries* des clients et A l'ensemble des arcs. Chaque arc est un couple de sommets (x, y) représentant la précédence du sommet x sur le sommet y.

Un exemple de ce graphe est proposé sur la *Figure 3.15* dans le cadre d'une tournée publique. On constate la précédence des sommets de *pickup* (+) sur les sommets de *delivery* (-) pour un même client, mais d'autres précédences sont également présentes : par exemple, les sommets 1.0^- et 4.0^+ doivent être insérés avant le sommet 2.0^+.

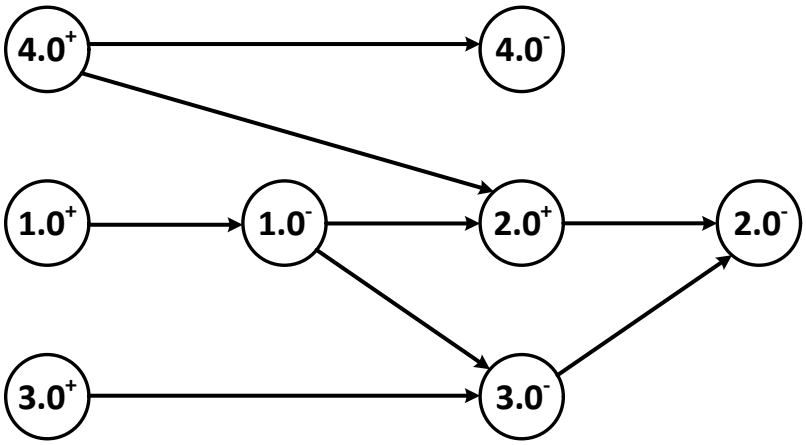


Figure 3.15 : Graphe de précédence entre les différents sommets de pickup et delivery des clients

La méthode générant le graphe de précédence est décrite dans l'*Algorithme 15* et se compose de trois étapes principales :

1. Ajout des arcs de précédence pour chaque couple de sommets (*pickup*, *delivery*) d'un même client ;
2. Calcul de la précédence pour chaque couple de sommets ;
3. Calcul de la précédence pour chaque couple de clients.

Les fonctions vérifiant la présence de contraintes précédence entre un couple de sommets ou un couple de clients sont détaillées dans les sections suivantes. Puisque les contraintes de précédence dépendent des fenêtres de temps, il est possible d'effectuer un prétraitement réduisant ces dernières en se basant sur les contraintes du DARP. Grâce à ce prétraitement, des contraintes de précédences supplémentaires peuvent être trouvées, fournissant un graphe plus complet et permettant ainsi une résolution plus rapide de l'algorithme de *Branch and Bound*. Le prétraitement est décrit dans la partie suivante.

Algorithme 15 $compute\_precedence\_graph$

Input parameters

$allClients, allNodes$: Array that contains all clients / all nodes

Output variable

$arcs$: Hashset that contains all arcs in the precedence graph

Local variables

$c1, c2$: Temporary client variables

$pubRes, priRes$: Temporary result variables

```
Begin
1   // Step 1. Compute precedence arcs (pickup, delivery) for each client
2   for all client in allClients do
3   |  arcs.add(client.get_initial_pickup(), client.get_initial_delivery())
4   |  for all alternativePickup in client.get_alternative_pickups() do
5   |  |  for all alternativeDelivery in client.get_alternative_deliveries() do
6   |  |  |  arcs.add(alternativePickup, alternativeDelivery)
7   |  |  end for
8   |  end for
9   end for
10  // Step 2. Compute precedence arcs between each node couple
11  for all start in allNodes do
12  |  for all end in allNodes do
13  |  |  if not (start.is_depot() or end.is_depot()) and precedence_constraint(start, end)) then
14  |  |  |  arcs.add(start, end)
15  |  |  end if
16  |  end for
17  end for
18  // Step 3. Compute precedence arcs between each client couple
19  for i := 0 to allClients.size() − 1 do
20  |  c1 := allClients[i]
21  |  for j := i + 1 to allClients.size() − 61 do
22  |  |  c2 := allClients[j]
23  |  |  pubRes := public_precedence_constraint(c1, c2)
24  |  |  priRes := private_precedence_constraint(c1, c2)
25  |  |  if pubRes = 1 then
26  |  |  |  arcs.add(c1.get_initial_pickup(), c2.get_initial_pickup())
27  |  |  end if
28  |  |  if pubRes = −1 then
29  |  |  |  arcs.add(c2.get_initial_pickup(), c1.get_initial_pickup())
30  |  |  end if
31  |  |  if priRes = 1 then
32  |  |  |  for all n1 in c1.get_alternative_pickups() do
33  |  |  |  |  for all n2 in c2.get_alternative_pickups() do
34  |  |  |  |  |  arcs.add(n1, n2)
35  |  |  |  |  end for
36  |  |  |  end for
37  |  |  end if
38  |  |  if priRes = −1 then
39  |  |  |  for all n1 in c1.get_alternative_pickups() do
40  |  |  |  |  for all n2 in c2.get_alternative_pickups() do
41  |  |  |  |  |  arcs.add(n2, n1)
42  |  |  |  |  end for
43  |  |  |  end for
44  |  |  end if
45  |  end for
46  end for
End
```

a) Prétraitement des fenêtres de temps

Le but du prétraitement sur les fenêtres de temps est de restreindre leurs intervalles. Ce calcul n'est effectué qu'une seule fois, lors de la lecture des données de l'instance. La réduction des intervalles de temps permet d'augmenter le nombre de coupes dans la phase de *Branch and Bound*, et ainsi accélérer la vitesse d'évaluation d'une tournée.

Certaines instances de la littérature imposent une fenêtre de temps uniquement sur le sommet de *pickup* ou de *delivery* d'un client. Grâce à l'utilisation des contraintes de précédence du problème, il est possible de réduire ces intervalles. Chaque ensemble de fenêtre de temps $[e_i, l_i]_h$ pour le pickup i et $[e_j, l_j]_h$ pour le delivery j d'un client h est donc modifiée grâce aux équations présentées ci-dessous en respectant l'ordre de calcul imposé, d_i étant le temps de service au sommet i, c_{ij} la distance entre le sommet i et j et RT_h le riding time maximum du client h :

1. $e'_j = \max\{e_i + d_i + c_{ij}\ , e_j\}$;
2. $l'_j = \min\{l_i + d_i + RT_h\ , l_j\}$;
3. $e'_i = \max\{e'_j - d_i - RT_h\ , e_i\}$;
4. $l'_i = \min\{l'_j - d_i - c_{ij}\ , l_i\}$.

Ce prétraitement peut être appliqué uniquement si l'inégalité triangulaire est respectée, ce qui est le cas dans les instances de la littérature mais également les instances générées et utilisées dans le cadre de cette thèse. Une illustration de la réduction des fenêtres de temps est proposée sur la *Figure 3.16*.

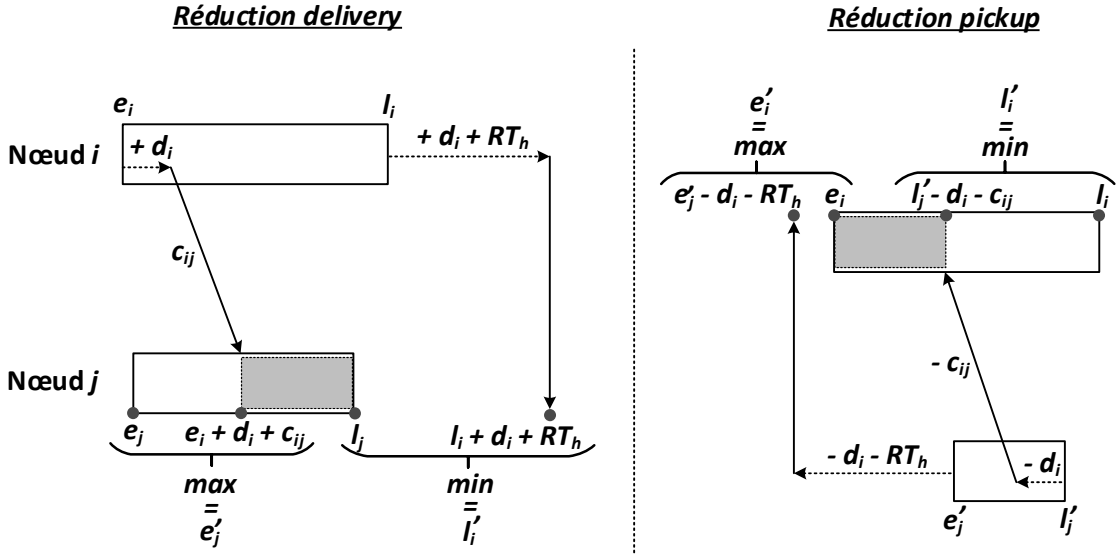


Figure 3.16 : Prétraitement sur les fenêtres de temps (Chassaing, 2015)

b) Précédence entre les sommets

Une première méthode pour ajouter des arcs dans le graphe de précédence est l'utilisation de la fonction $precedence\_constraint(i, j)$ déterminant si un sommet i doit être visité avant un sommet j en utilisant les fenêtres de temps. Un sommet i précède un sommet j si les deux équations suivantes sont valides :

- $e_i + d_i + c_{ij} \leq l_j$;
- $e_j + d_j + c_{ji} > l_i$.

L'utilisation de ces deux équations est illustrée par l'*Algorithme 16* prenant en entrée deux sommets et retournant $true$ si le premier sommet doit précéder le deuxième, et $false$ sinon.

Algorithme 16 $precedence\_constraint$

Input parameters
$start$: First visited node
end : Last visited node

Output variable
res : Boolean result

Global variable
$dist$: Distance matrix

Begin
1 $res := start.inf + start.serviceTime + dist[start][end] \leq end.sup$
2
3 $res := res$ **and** $end.inf + end.serviceTime + dist[end][start] > start.sup$
4
5 **return** { res }
End

c) Précédence entre les clients

Le calcul de précédence pour chaque couple de sommets peut être généralisé aux clients (Chassaing et al., 2016) : en effet, il est possible de déterminer les ordres d'insertion dans les tournées de véhicules pour chaque couple de clients. Si un ordre est imposé, alors il existe une contrainte de précédence entre les deux clients.

Les contraintes du DARP imposent l'insertion du sommet de *pickup* avant celle du sommet de *delivery* pour un même client. Le nombre total de tournées potentiellement réalisables composées de deux clients est donc de 6. L'ensemble des ordres possibles est représenté par un ensemble de tournées $\lambda_k, k \in \{1, \dots, 6\}$ avec i^+/j^+ et i^-/j^- les sommets initiaux de *pickup* et *delivery* du client i/j :

- $\lambda_1 = (i^+, j^+, j^-, i^-)$;
- $\lambda_2 = (i^+, j^+, i^-, j^-)$;
- $\lambda_3 = (i^+, i^-, j^+, j^-)$;
- $\lambda_4 = (j^+, i^+, i^-, j^-)$;
- $\lambda_5 = (j^+, i^+, j^-, i^-)$;
- $\lambda_6 = (j^+, j^-, i^+, i^-)$.

L'ensemble λ_k est divisé ensuite en deux sous-ensembles :

- Le sous-ensemble $\lambda_{i \to j} = \lambda_k, k \in \{1,2,3\}$ représente les tournées ou le sommet initial de *pickup* du client i est visité avant celui du client j ;
- Au contraire, le sous-ensemble $\lambda_{j \to i} = \lambda_k, k \in \{4,5,6\}$ représente les tournées ou le sommet initial de *pickup* du client j est visité avant celui du client i.

Une contrainte de précédence est imposée entre les sommets initiaux de type *pickup* des deux clients si les ordres valides, c'est-à-dire les ordres respectant l'ensemble des contraintes du

DARP-PV-AN, appartiennent à un même sous-ensemble $\lambda_{i\to j}$ ou $\lambda_{j\to i}$. Si aucun ordre n'est valide ou s'il existe des ordres valides dans les deux sous-ensembles, aucune contrainte de précédence n'est imposée.

L'exemple de la *Figure 3.17* montre le test de précédence entre les clients 1 et 2. Dans cet exemple la distance entre chaque sommet est de 5 et la durée de service à chaque sommet est de 3. On remarque que seul l'ensemble $\lambda_{1\to 2}$ contient un ordre ne violant pas les contraintes du DARP-PV-AN (l'ordre $1^+, 1^-, 2^+, 2^-$). Il existe donc une contrainte de précédence forçant le sommet 1^+ à être servi avant le sommet 2^+.

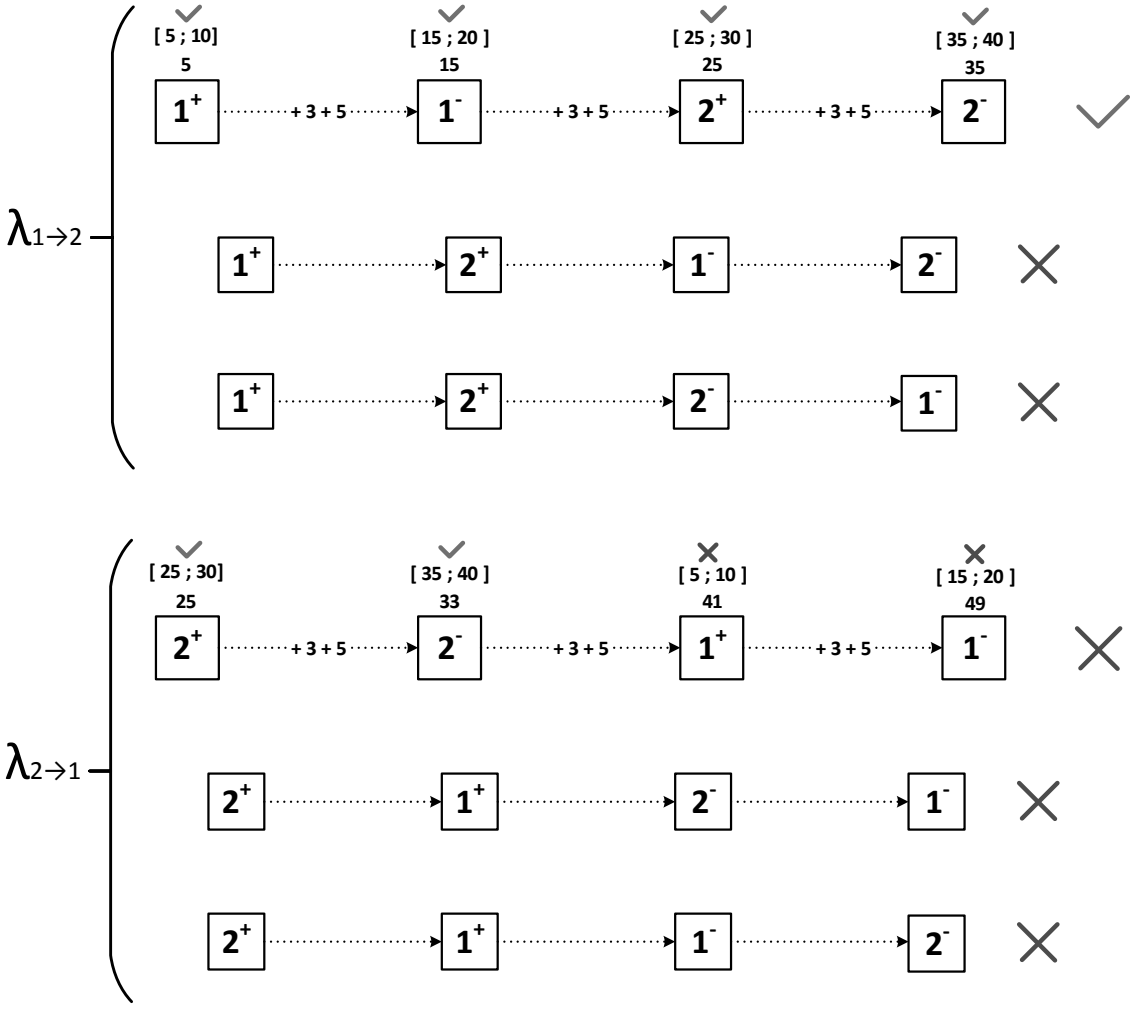


Figure 3.17 : Test de précédence entre les clients 1 et 2 : une contrainte existe forçant le sommet de pickup *du client 1 à être servi avant le sommet de* pickup *du client 2*

L'*Algorithme 17* illustre le calcul vérifiant si une contrainte de précédence existe entre deux clients étant transporté par un véhicule public (donc sur leur sommet initial) La fonction $is\_valid(a, b, c, d)$ renvoie la valeur $true$ si la tournée composée des sommets a, b, c et d dans cet ordre est valide, $false$ sinon. Pour cela la fonction renvoie la valeur $true$ si les 3 inégalités définies ci-dessous sont valides :

Soit $bst_k \in \mathbb{R}, k \in \{1,2,3\}$ tel que

$$bst_1 = e_a + d_a + c_{ab}$$

$$bst_2 = \max\{bst_1, e_b\} + d_b + c_{bc}$$

$$bst_3 = \max\{bst_2, e_c\} + d_c + c_{cd}$$

Si les inégalités suivantes sont valides, alors $is\_valid(a, b, c, d)$ renvoie $true$:

$$bst_1 \leq l_b$$

$$bst_2 \leq l_c$$

$$bst_3 \leq l_d$$

Algorithme 17 $public\_precedence\_constraint$

Input parameters

$c1, c2$: First client and second client

Output variable

res : Integer result with $0 \rightarrow$ no precedence constraint,
$1 \rightarrow c1$ initial pickup must be before $c2$ initial pickup,
$-1 \rightarrow c2$ initial pickup must be before $c1$ initial pickup.

Local variables

$p1, d1, p2, d2$: Temporary node variables
$c1c2, c2c1$: Temporary boolean variables

Begin

1 $p1$:= $c1$.get_initial_pickup()
2 $d1$:= $c1$.get_initial_delivery()
3 $p2$:= $c2$.get_initial_pickup()
4 $d2$:= $c2$.get_initial_delivery()
5
6 $c1c2$:= $false$
7 $c2c1$:= $false$
8
9 **// Step 1. Evaluate order of type $\lambda_{c1 \rightarrow c2}$**
10 $c1c2$:= $c1c2$ **or** is_valid($p1, p2, d2, d1$)
11 **if not** $c1c2$ **then** $c1c2$:= $c1c2$ **or** is_valid($p1, p2, d1, d2$) **end if**
12 **if not** $c1c2$ **then** $c1c2$:= $c1c2$ **or** is_valid($p1, d1, p2, d2$) **end if**
13
14 **// Step 2. Evaluate order of type $\lambda_{c2 \rightarrow c1}$**
15 $c2c1$:= $c2c1$ **or** is_valid($p2, p1, d1, d2$)
16 **if not** $c2c1$ **then** $c2c1$:= $c2c1$ **or** is_valid($p2, p1, d2, d1$) **end if**
17 **if not** $c2c1$ **then** $c2c1$:= $c2c1$ **or** is_valid($p2, d2, p1, d1$) **end if**
18
19 **// Step 3. Compute res variable**
20 **if** $c1c2$ **and** $c2c1$ **then**
21 | res := 0
22 **else if** $c1c2$ **and not** $c2c1$ **then**
23 | res := 1
24 **else**
25 | res := -1
26 **end if**
27
28 **return** { res }

End

Comme illustré dans l'*Algorithme 17*, les contraintes de précédence entre les clients ne prennent en compte que les sommets initiaux. Il est possible d'étendre ces contraintes de

précédence en utilisant les sommets alternatifs apparaissant dans les tournées avec des véhicules privés. La méthode nommée $private\_precedence\_constraint$ est illustrée par l'*Algorithme 18*, et fonctionne comme la méthode $public\_precedence\_constraint$ expliquée précédemment : pour un ensemble de 4 sommets, les 6 ordres sont toujours à tester, ces derniers étant partitionnés en deux sous-ensembles $\lambda_{i\to j}$ et $\lambda_{j\to i}$.

Cependant, l'ensemble de 4 sommets initiaux est unique pour un couple de clients donné transporté par un véhicule public. En revanche pour ce même couple de client transporté par un véhicule privé, il existe plusieurs ensembles de 4 sommets alternatifs représentés par l'ensemble X. Les ordres possibles sont donc maintenant représentés par un ensemble de tournées $\lambda_{x,k}, x \in X, k \in \{1, \dots, 6\}$ divisé en deux sous-ensembles :

- Le sous-ensemble $\lambda_{i\to j} = \lambda_{x,k}, k \in \{1,2,3\}, x \in X$, représente les tournées ou le sommet alternatif de *pickup* du client i est visité avant celui du client j ;
- Au contraire, le sous-ensemble $\lambda_{j\to i} = \lambda_{x,k}, k \in \{4,5,6\}, x \in X$, représente les tournées ou le sommet alternatif de *pickup* du client j est visité avant celui du client i.

Algorithme 18 $private\_precedence\_constraint$

Input parameters

$c1, c2$: First client and second client

Output parameter

res : Integer result with 0 → no precedence constraint,
$1 \to c1$ alternative pickups must be before $c2$ alternative pickups,
$-1 \to c2$ alternative pickups must be before $c1$ alternative pickups.

Local variables

$c1c2, c2c1$: Temporary boolean variables

Begin

```
1   c1c2 := false
2   c2c1 := false
3   for all p1 in c1.get_alternative_pickups() do
4   |   for all d1 in c1.get_alternative_deliveries() do
5   |   |   for all p2 in c2.get_alternative_pickups() do
6   |   |   |   for all d2 in c2.get_alternative_deliveries() do
7   |   |   |   |   if not c1c2 then c1c2 := c1c2 or is_valid(p1, p2, d2, d1) end if
8   |   |   |   |   if not c1c2 then c1c2 := c1c2 or is_valid(p1, p2, d1, d2) end if
9   |   |   |   |   if not c1c2 then c1c2 := c1c2 or is_valid(p1, d1, p2, d2) end if
10  |   |   |   |   if not c2c1 then c2c1 := c2c1 or is_valid(p2, p1, d1, d2) end if
11  |   |   |   |   if not c2c1 then c2c1 := c2c1 or is_valid(p2, p1, d2, d1) end if
12  |   |   |   |   if not c2c1 then c2c1 := c2c1 or is_valid(p2, d2, p1, d1) end if
13  |   |   |   end for
14  |   |   end for
15  |   end for
16  end for
17  if c1c2 and c2c1 then
18  |   res := 0
19  else if c1c2 and not c2c1 then
20  |   res := 1
21  else
22  |   res := -1
23  end if
24  return { res }
End
```

L'ajout d'une contrainte de précédence sur les sommets alternatifs de type *pickup* de deux clients a donc lieux si les ordres valides appartiennent à un même sous-ensemble $\lambda_{i\to j}$ ou $\lambda_{j\to i}$. Comme pour les sommets initiaux, si aucun ordre n'est valide ou s'il existe des ordres valides dans les deux sous-ensembles, aucune contrainte de précédence n'est imposée.

Dans le cadre d'une précédence entre deux clients utilisant leurs sommets alternatifs, l'ensemble des arcs (x, y) est inséré dans $A, \forall x$ appartenant aux sommets de *pickup* alternatifs du premier client et $\forall y$ appartenant aux sommets de *pickup* alternatifs du second client (ligne 32 à 36 et 39 à 43 de l'*Algorithme 15*).

d) Filtrage du graphe de précédence

Puisque le graphe de précédence est calculé avant le lancement de l'algorithme génétique sur l'ensemble des clients, il contient des arcs de précédence sur des clients non présents dans une tournée. Le filtrage du graphe de précédence est donc nécessaire avant chaque application de l'algorithme de *Branch and Bound* sur cette tournée. Il permet de filtrer l'ensemble des arcs présents dans le graphe par la fonction $filter\_graph$ et retourne une liste ne contenant que les arcs des clients présents dans la tournée λ qui va être évaluée : pour tout arc (x, y) présent dans le graphe de précédence, si x ou y ne sont pas des nœuds appartenant à des clients présents dans λ, alors l'arc n'est pas inséré dans la liste de retour.

Un exemple est proposé sur la *Figure 3.18* : une instance comportant 4 clients numérotés de 1 à 4 possède un graphe de précédence. Une tournée en attente d'évaluation comporte les clients 1 et 3. Le graphe de précédence est filtré et retourne une liste ne comportant que les arrêtes avec les clients 1 et 3 car la tournée ne visite pas les clients 2 et 4.

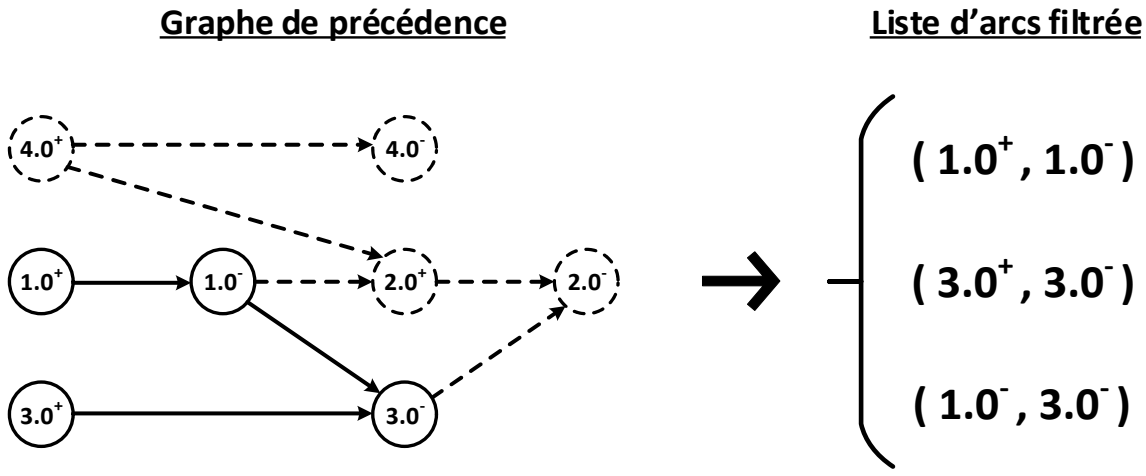


Figure 3.18 : Filtrage du graphe de précédence d'une instance de 4 clients avant l'évaluation d'une tournée comportant les clients 1 et 3

3.4.3 Nœuds de l'arbre de recherche

Le principe de séparation des problèmes en sous-problèmes dans l'algorithme de *Branch and Bound* forme une hiérarchie naturelle de recherche de solution en arbre, appelé arbre de recherche ou encore arbre de décision. Le principe d'évaluation est effectué sur un nœud de l'arbre de recherche grâce aux informations stockées dans ce dernier et aux choix effectués sur ses nœuds parents.

Dans l'algorithme proposé, l'évaluation correspond à l'insertion d'un sommet à l'endroit courant de la tournée et de vérifier que l'ensemble des contraintes du DARP-PV-AN sont respectées. L'exemple illustré sur la *Figure 3.19* montre un parcours de l'algorithme de *Branch and Bound* pour une tournée publique composée des clients 1 et 2 : la tournée commence et se termine par le dépôt (0.0), les clients utilisent leurs sommets initiaux de *pickup* (1.0^+ et 2.0^+) et de *delivery* (1.0^- et 2.0^-) et l'ordre minimisant la distance de la tournée tout en respectant l'ensemble des contraintes est $(0.0, 1.0^+, 2.0^+, 2.0^-, 1.0^-, 0.0)$.

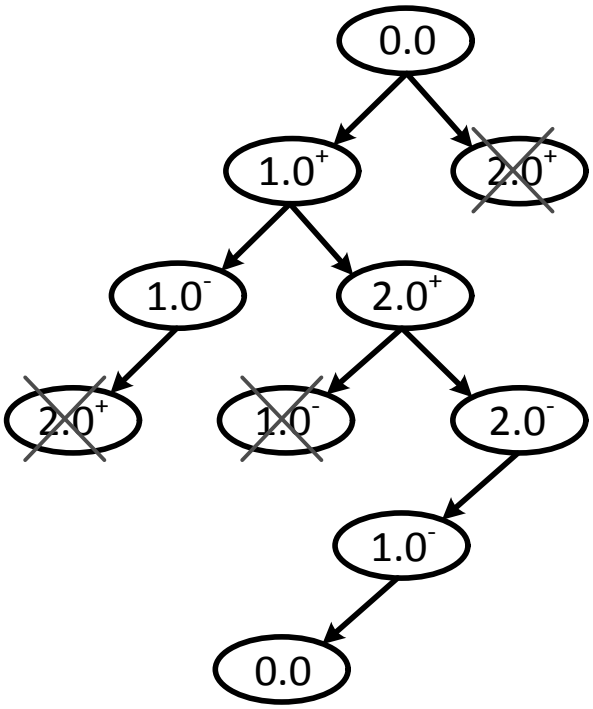


Figure 3.19 : Exemple d'un parcours de l'algorithme de Branch and Bound *sur une tournée publique*

Un nœud est donc une structure contenant un ensemble de données sur la situation courante avec les valeurs suivantes :

- $previous$: nœud parent ;
- $customerNode$: sommet inséré à ce nœud ;
- $capacity$: capacité du véhicule après insertion du sommet ;
- $sumDistance$: somme de la distance parcourue jusqu'au sommet ;
- ES : estimation du début du service au plus tôt sur ce sommet ;
- $waiting$: temps d'attente estimé sur ce sommet ;
- $RTmargin$: marge du *riding time* du client à qui appartient le sommet ;
- $clientsInV$: ensemble des clients dans le véhicule avant l'arrivée sur le sommet ;
- $remainingNodes$: liste de sommets restants à insérer avant l'arrivée sur le sommet ;
- $precedenceArcs$: liste des arcs filtrés du graphe de précédence décrit dans la section 3.4.2.

L'ensemble de ces informations permettent d'affecter le sommet courant dans la tournée mais également de calculer quand ce dernier est servi par le véhicule. Elles permettent aussi de couper (supprimer) la branche commençant par ce nœud si une contrainte du DARP-PV-AN est violée ou si l'insertion du sommet à l'endroit courant ne permet pas un ordre de visite avec

un coût inférieur au meilleur ordre connu. L'ensemble des coupes effectuées dans le *Branch and Bound* est décrit section 3.4.5.

Grâce à la structure d'arbre de recherche, il est possible de déterminer l'ordre optimal de parcours de la tournée par le véhicule. Pour cela, le fonctionnement de l'arbre et son algorithme de parcours sont expliqué dans la section suivante.

3.4.4 Arbre de recherche et parcours

L'arbre de recherche est une structure composée des nœuds décrits dans la section précédente et permettant, à travers une fonction de parcours, de trouver l'ordre optimal de visite des clients par le véhicule dans une tournée.

Tout comme la structure de nœud, l'arbre de recherche contient un ensemble de données qui sont modifiées lors de son parcours :

- $tour$: structure contenant les informations sur les clients présents dans la tournée et également l'ordre de visite optimal une fois celui-ci calculé par l'arbre de recherche ;
- $vehicle$: véhicule effectuant la tournée ;
- $best$: meilleur nœud courant lors du parcours de l'arbre ;
- $root$: nœud racine de l'arbre ;
- $precedenceGraph$: graphe de précédence décrit dans la section 3.4.2 ;
- $precedenceArcs$: liste des arcs filtrés du graphe de précédence décrit dans la section 3.4.2.

La recherche de l'ordre optimal est divisée en 3 parties décrites ci-après : le prétraitement par la fonction $start\_tree\_search$, le parcours en profondeur par la fonction $depth\_first\_search$ et l'affectation du meilleur ordre de visite trouvé par la fonction $affect\_visit\_order$.

a) Prétraitement

La première partie concerne un prétraitement sur la tournée, effectué par la fonction $start\_tree\_search$ et décrit dans l'*Algorithme 19*. Un filtrage sur le graphe de précédence est d'abord effectué afin de conserver uniquement les arcs concernant les clients présents dans la tournée (ligne 2). Ensuite, les valeurs $minSup$ et $maxInf$ sont calculées (ligne 3), représentant respectivement la plus petite borne supérieure et la plus grande borne inférieure des fenêtres de temps des clients présents dans la tournée. Elles sont nécessaires à l'appel des fonctions $cut\_min\_max\_VRT$ (ligne 6) et $cut\_boruvka$ (ligne 9), ces dernières étant des fonctions de coupe vérifiant s'il est nécessaire de parcourir l'arbre de recherche. Le fonctionnement de ces deux méthodes est décrit dans la section 3.4.5.

Si l'une des deux fonctions de coupe renvoie la valeur $false$, alors il est inutile de parcourir l'arbre de recherche car une contrainte du DARP-PV-AN sera violée. Si tel est le cas, l'évaluation de la tournée est terminée, sinon, l'algorithme de parcours de l'arbre peut être lancé.

Algorithme 19 $start\_tree\_search$

Output variable

res : Boolean result that return $true$ if no constraint is violated, $false$ otherwise

Local variables

$maxRTV$: Maximum vehicle riding time
$minSup$: Minimum upper bound of clients' time windows in the evaluated tour
$maxInf$: Maximum lower bound of clients' time windows in the evaluated tour

Begin

1 $maxRTV$:= $vehicle$.get_max_riding_time()
2 $precedenceArcs$:= filter_graph($precedenceGraph$)
3 $\{minsup, maxInf\}$:= compute_minSup_maxInf()
4 $res := true$
5
6 **if not** cut_min_max_VRT($minSup, maxInf, maxRTV$) **then**
7 | $res := false$
8 **end if**
9 **if** res **and not** cut_boruvka($tour, precedenceArcs, maxRTV$) **then**
10 | $res := false$
11 **end if**
12
13 **return** { res }
End

b) Parcours en profondeur

Le parcours de l'arbre de recherche est effectué par une méthode en profondeur sur un graphe orienté acyclique. Naturellement décrit par un algorithme récursif, il est implémenté dans la fonction $depth\_first\_search$ de manière itérative par l'utilisation d'une pile de type « dernier arrivé, premier sorti » (*Last In First Out* – LIFO).

Le choix d'un algorithme de parcours en profondeur se justifie par la nécessité de trouver le plus rapidement possible des ordres de visite valides formant ainsi une borne supérieure ou égale à l'ordre optimal. Cela permet des coupes dans l'arbre plus rapides lors du parcours accélérant ainsi la résolution optimale de l'ordonnancement de la tournée.

Comme le montre l'exemple de la *Figure 3.20*, dans une tournée comportant 2 clients, le parcours en profondeur trouve un ordre en 6 nœuds alors que celui en largeur nécessite 20 nœuds. Le parcours en profondeur est donc plus à même à trouver rapidement un ordre de visite valide lors de l'évaluation d'une tournée et est ainsi celui retenu pour cet algorithme.

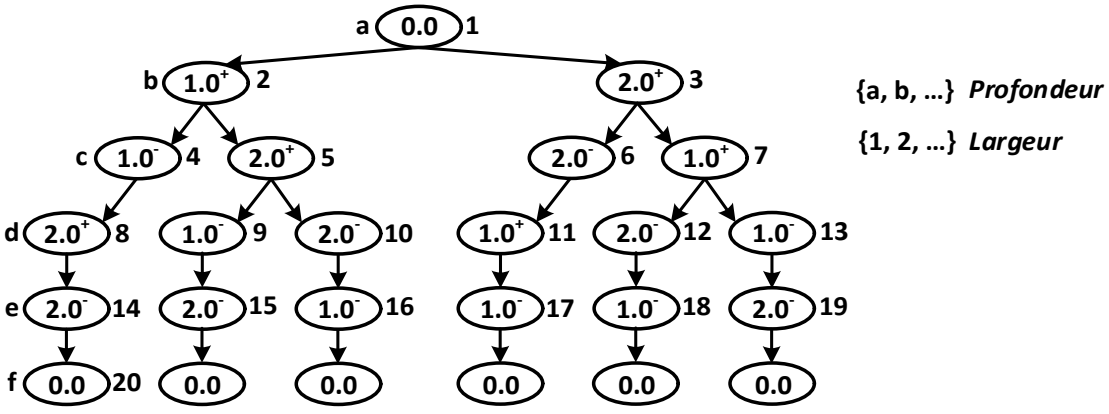


Figure 3.20 : Comparaison du parcours en profondeur et en largeur

Le déroulement de la méthode est décrit par l'*Algorithme 20*. Ce dernier commence par initialiser le nœud racine $root$ en lui affectant le dépôt (ligne 1), ainsi que la pile $stack$ grâce à la fonction $initialize\_stack$ (ligne 2). Cette dernière insère dans la pile l'ensemble des sommets n'ayant pas de contraintes de précédence. Ensuite l'algorithme traite les nœuds tant que la pile ne sera pas vide (ligne 6 à 27).

La fonction $continue\_tree\_search$ (ligne 10) teste si le sommet inséré par le nœud courant viole les contraintes du problème : si tel est le cas, le nœud courant ne produit pas de nœuds fils et est donc coupé (l'ensemble des tests effectué est décrit section 3.4.5). Sinon, les ensembles $remainingNodes$ et $precedenceArcs$ du nœud courant sont filtrés en supprimant à l'intérieur de ces derniers le sommet venant d'être inséré (ligne 11 et 12).

Algorithme 20 $depth\_first\_search$

Local variables

$stack$	:	Stack used to store untreated nodes of the tree
$bestCost$	:	Temporary variable saving the current order best cost value
$currentNode$	:	The node that is computed in the current iteration
$distance$	:	Distance between $currentNode$ and its parent

Global variables

$depot$	:	Depot of the instance
MAX_VALUE	:	Big M value
$dist$	:	Distance matrix
$best$	:	Last node of optimal order

```
Begin
1   root := instantiate_root(depot, tour, precedenceArcs)
2   stack := initialize_stack()
3   bestCost := MAX_VALUE
4   best := NULL
5
6   while not stack.is_empty do
7   |   currentNode := stack.pop()
8   |   distance := dist[currentNode.previous.customerNode][currentNode.customerNode]
9   |
10  |   if continue_tree_search(currentNode, distance , vehicle, bestCost) then
11  |   |   currentNode.set_remaining_nodes()
12  |   |   currentNode.set_precedence_arcs()
13  |   |
14  |   |   if currentNode.remainingNodes.size() = 0 then
15  |   |   |   if best = NULL or bestCost > currentNode.sumDistance then
16  |   |   |   |   best := currentNode
17  |   |   |   |   bestCost := best.sumDistance
18  |   |   |   end if
19  |   |   else
20  |   |   |   for all customerNode in currentNode.remainingNodes do
21  |   |   |   |   if currentNode.next_node_possible_to_insert(customerNode, vehicle) then
22  |   |   |   |   |   stack.push(create_node(currentNode, customerNode))
23  |   |   |   |   end if
24  |   |   |   end for
25  |   |   end if
26  |   end if
27  end while
End
```

Une fois le filtre effectué sur l'ensemble $remainingNodes$ du nœud courant, un test est nécessaire pour vérifier si le sommet inséré est le dernier, formant ainsi un nouvel ordre valide pour la tournée en cours d'évaluation (ligne 14). Si le nouvel ordre trouvé améliore le meilleur ordre courant alors le nœud courant est sauvegardé dans le nœud $best$ (ligne 15 à 17).

En revanche, si le sommet inséré par le nœud courant n'était pas le dernier de la tournée, l'algorithme parcours l'ensemble des sommets non insérés et vérifie s'ils peuvent être ajoutés à la prochaine position courante grâce à la fonction $next\_node\_possible\_to\_insert$ cette dernière utilisant les arcs du graphe de précédence (ligne 21). Pour chaque sommet dont l'ajout est possible après le sommet du nœud courant, un nouveau nœud est créé et leur est associé avant d'être ajouté dans la pile (ligne 22).

Une fois la pile vide, le nœud $best$ contient le dernier sommet de l'ordre optimal de la tournée, ou la valeur NULL si aucun ordre respectant l'ensemble des contraintes n'existe. Puisque chaque nœud contient un lien vers son nœud parent ($previous$) il est possible de parcourir le chemin jusqu'à la racine, ce dernier fournissant l'ordre de visite optimal : cette méthode est décrite dans la section suivante.

c) Affectation du meilleur ordre de visite

L'affectation du meilleur ordre de visite dans la variable $tour$ trouvée par l'arbre de recherche est effectuée par la fonction $affect\_visit\_order$ décrite dans l'*Algorithme 21*. Il consiste à remonter du nœud $best$ jusqu'à la racine en stockant les sommets parcourus (ligne 4 à 7). L'ordre est ensuite inversé puis affecté dans la variable $tour$ avec son coût (ligne 13 à 15). L'ajout du nœud racine (ligne 9 à 11) n'est effectué que si la tournée est publique, ce dernier représentant le dépôt.

Algorithme 21 $affect\_visit\_order$

```
Local variables
    it          :  Iterator on nodes
    visitOrder  :  Array of customer nodes representing the optimal order

Global variable
    best        :  Last node of optimal order

Begin
1   if not best = NULL then
2   |   it := best
3   |
4   |   while not it.previous = NULL do
5   |   |   visitOrder.add(it.customerNode)
6   |   |   it := it.previous
7   |   end while
8   |
9   |   if tour.is_public() then
10  |   |   visitOrder.add(it.customerNode)
11  |   end if
12  |
13  |   reverse(visitOrder)
14  |   tour.set_visit_order(visitOrder)
15  |   tour.set_distance(best.sumDistance)
16  end if
End
```

3.4.5 Coupes

Dans un algorithme de type *Branch and Bound*, une méthode naïve consiste à énumérer l'ensemble des solutions du problème, afin de donner celle minimisant la fonction objectif. Il est également possible d'utiliser les particularités du problème étudié afin d'éviter de parcourir des solutions dont on sait qu'elles ne pourront pas être la solution optimale. Les coupes représentent des tests effectués sur les nœuds de l'arbre de recherche pour savoir si l'insertion d'un sommet est valide dans l'ordre de visite en cours de construction et si l'estimation de l'objectif de cette dernière est inférieure à la valeur du meilleur ordre actuel. Si les tests sont négatifs, il est inutile de poursuivre la génération des fils de ce nœud et une coupe peut être appliquée, limitant ainsi le parcours de l'espace des solutions.

Dans la méthode proposée, les coupes se divisent en deux ensembles. Le premier comporte les fonctions $cut\_min\_max\_VRT$ et $cut\_boruvka$ déterminant avant le parcours de l'arbre de recherche si aucun ordre valide n'existe pour un ensemble de clients donnés. Le second ensemble est composé des fonctions $cut\_EBST, cut\_vehicle\_capacity, cut\_EVRT$, $cut\_distance$ et cut_ECRT appelées dans la fonction $continue\_tree\_search$: chacune de ces fonctions permet donc une coupe au sein de l'arbre de recherche si une violation de contrainte apparait ou bien si l'ordre de visite utilisant le nœud courant ne peut être optimal, suite à l'insertion d'un sommet.

La description du fonctionnement de ces différentes méthodes de coupe est proposée dans les sections suivantes.

a) Min/max *vehicle riding time*

La fonction $cut\_min\_max\_VRT$ vérifie par l'étude des fenêtres de temps d'un ensemble de clients, si le temps de trajet maximal du véhicule n'est pas violé. Cette vérification est effectuée avant le début du parcours de l'arbre de recherche.

Comme illustré sur la *Figure 3.21*, pour un ensemble de nœuds à visiter N avec $[e_k, l_k], k \in N$ leurs fenêtres de temps, le temps de trajet minimal T_{min} d'une tournée visitant l'ensemble N est :

$$T_{min} = \max e_k - \min l_k \,, k \in N$$

Or le DARP-PV-AN limite le temps de trajet maximal d'une tournée à T. Donc si $T_{min} > T$, il est impossible de visiter la totalité de l'ensemble N sans violer la contrainte de durée maximale d'une tournée. Par conséquent, il est inutile de parcourir l'arbre de recherche et la tournée est marquée comme invalide.

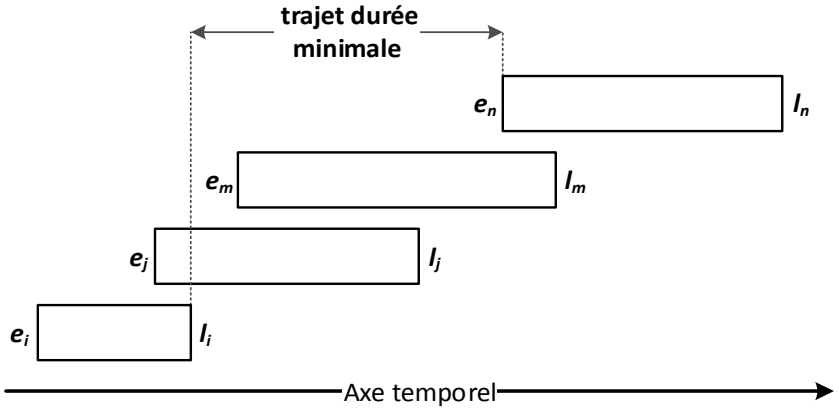


Figure 3.21 : Durée minimale du trajet d'un véhicule pour un ensemble de nœuds

b) Arbre couvrant de poids minimal

La fonction $cut\_boruvka$ vérifie comme la fonction $cut\_min\_max\_VRT$ la violation de la contrainte de trajet maximal du véhicule, mais cette fois-ci par l'étude des distances séparant les nœuds. Pour cela, la notion d'arbre couvrant de poids minimal (ACM) est utilisée : dans un graphe, un ACM est un arbre couvrant (arbre qui connecte tous les sommets du graphe) dont la somme des poids des arrêtes est minimal.

Dans le problème étudié, l'ensemble des nœuds à visiter par un véhicule dans une tournée forme un graphe complet dont les arrêtes sont pondérées, et un ACM représente une borne inférieure T_{min} de la durée d'une tournée, comme illustré sur la *Figure 3.22*. Comme pour la coupe de la fonction $cut\_min\_max\_VRT$, si $T_{min} > T$ il y a violation de la contrainte de la durée maximale d'une tournée et l'arbre de recherche n'est pas parcouru.

Pour trouver l'arbre couvrant de poids minimal, l'algorithme de Borůvka est implémenté (Borůvka, 1926), pour une complexité en $O(A \times ln(V))$ où A est le nombre d'arrêtes et V le nombre de sommets du graphe. D'autres méthodes de résolution existent comme l'algorithme de Prim (Prim, 1957) ou encore l'algorithme de Kruskal (Kruskal, 1956) et des méthodes plus récentes ont également été proposées utilisant l'inverse de la fonction d'Ackermann (Chazelle, 2000).

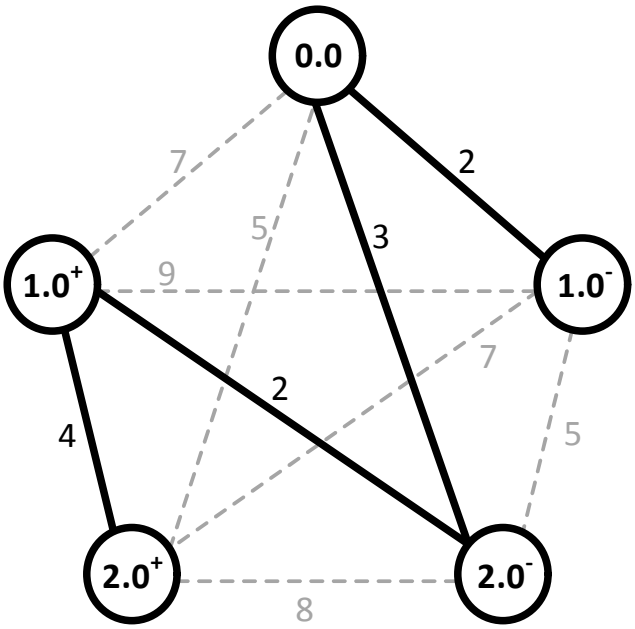


Figure 3.22 : Arbre couvrant de poids minimal sur une tournée publique avec deux clients

c) Début du temps de service

La fonction cut_EBST (*Estimated Beginning Service Time* – EBST) est appelée par $continue\_tree\_search$ sur chaque nœud de l'arbre de recherche et est représentée par l'*Algorithme 22*. Elle met à jour, pour le sommet courant, le début du temps de service au plus tôt (ligne 8), l'attente du véhicule sur le sommet (ligne 9) et la marge entre le temps de trajet actuel du client et la durée maximale (11 à 18). La méthode $compute\_RT\_margin$ est appelée si le sommet inséré est de type *delivery* : elle renvoie alors la marge entre le temps de trajet du client et la durée de transport maximale. La fonction vérifie également que le début du temps de service au plus tôt respecte la fenêtre de temps du sommet inséré (ligne 4).

Algorithme 22 cut_EBST

Input parameters

$currentNode$: Current node of the decision tree
$distance$: Distance between previous customer node and current one

Output variable

res : Boolean value returning $true$ if time window constraint is not violated, $false$ otherwise

Local variable

sum : Temporary integer containing the new estimated beginning service time

Global variable

$depot$: Depot customer node

Begin

```
1   sum := currentNode.previous.ES + distance +
2               currentNode.previous.customerNode.get_service_time()
3
4   if sum > currentNode.customerNode.get_upper_TW() then
5   |   res := false
6   else
7   |   res := true
8   |   currentNode.ES := max(sum ; currentNode.customerNode.get_lower_TW())
9   |   currentNode.waiting := max(0 ; currentNode.customerNode.get_lower_TW() − sum)
10  |
11  |   if not currentNode.customerNode = depot then
12  |   |   if currentNode.customerNode.is_pickup() then
13  |   |   |   currentNode.RTmargin := currentNode.customerNode.get_max_RT()
14  |   |   else
15  |   |   |   currentNode.RTmargin :=
16  |   |   |       compute_RT_margin(currentNode.cusomterNode.get_client(), currentNode)
17  |   |   end if
18  |   end if
19  end if
20
21  return { res }
End
```

d) Capacité du véhicule

La fonction $cut\_vehicle\_capacity$ (*Algorithme 23*) est appelée par $continue\_tree\_search$ à chaque nœud de l'arbre de recherche pour mettre à jour le nombre de clients dans le véhicule ($capacity$) et vérifier que la capacité maximale du véhicule n'est pas violée, suite à l'insertion du sommet dans l'ordre de visite.

e) Distance parcourue

La fonction $cut\_distance$ (*Algorithme 24*) est appelée par $continue\_tree\_search$ pour chaque nœud de l'arbre de recherche. Elle met à jour la distance parcourue depuis le début de la tournée ($sumDistance$) et teste si cette distance n'est pas supérieure au coût du meilleur ordre trouvé suite à l'insertion du sommet.

Algorithme 23 $cut\_vehicle\_capacity$

Input parameters

$currentNode$: Current node of the decision tree
$vehicle$: Vehicle that performed the evaluated tour

Output variable

res : Boolean value returning $true$ if load constraint is not violated, $false$ otherwise

Local variable

sum : Temporary integer containing the new vehicle load

Begin

1 sum := $currentNode.previous.capacity + currentNode.customerNode$.get_nb_moves()
2
3 **if** $sum > vehicle$.get_max_capacity() **then**
4 | res := $false$
5 **else**
6 | res := $true$
7 | $currentNode.capacity$:= sum
8 **end if**
9
10 **return** { res }
End

Algorithme 24 $cut\_distance$

Input parameters

$currentNode$: Current node of the decision tree
$distance$: Distance between previous customer node and current one
$bestCost$: Distance of the best current order

Output variable

res : Boolean value returning $true$ if new distance is smaller than best order distance, $false$ otherwise

Local variable

sum : Temporary real containing the new traveled distance

Begin

1 sum := $currentNode.previous.sumDistance + distance$
2
3 **if** $sum > bestCost$ **then**
4 | res := $false$
5 **else**
6 | res := $true$
7 | $currentNode.sumDistance$:= sum
8 **end if**
9
10 **return** { res }
End

f) Temps de trajet du véhicule

La fonction cut_EVRT (*Estimated Vehicle Riding Time* – EVRT) appelée par $continue\_tree\_search$ sur chaque nœud de l'arbre de recherche permet de tester si une violation de la durée maximale du trajet du véhicule apparaît lors de l'insertion d'un nouveau sommet dans l'ordre courant. La méthode est divisée en deux tests principaux réalisés

simultanément par l'*Algorithme 25* aux lignes 12 et 13, nécessitant la valeur du temps de service au plus tard du premier sommet de l'ordre testé calculé par la boucle *while*.

Algorithme 25 cut_EVRT

Input parameters

$currentNode$	:	Current node of the decision tree
$distance$	:	Distance between previous customer node and current one
$vehicle$	:	Vehicle that performed the evaluated tour
$maxInf$	:	Maximum time windows lower bound between all nodes in the tour

Output variable

res	:	Boolean value returning $true$ if the maximum riding time constraint is not violated, $false$ otherwise

Local variables

LS	:	Temporary variable that will contains the value of the first node service time latest date
s	:	Smallest margin of each node between their latest service time date and their time window upper bound
it	:	Iterator on node
$tmp1, tmp2$	:	Temporary variables

Global variables

MAX_VALUE	:	Big M value
$dist$	:	Distance matrix

Begin

1 LS := $currentNode.ES$
2 s := MAX_VALUE
3 it := $currentNode$
4 **while not** $it.previous$ = NULL **then**
5 | s := min(s ; $it.customerNode$.get_upper_TW() − LS)
6 | $tmp1$:= $it.previous.customerNode$.get_upper_TW()
7 | $tmp2$:= $LS - it.previous.customerNode$.get_service_time()
8 | $- dist[it.previous.customerNode][it.customerNode]$
9 | LS := min($tmp1$; $tmp2$)
10 | it := $it.previous$
11 **end while**
12 **if** $currentNode.ES - LS > vehicle$.get_max_riding_time() **or**
13 $maxInf - LS - s > vehicle$.get_max_riding_time() **then**
14 | res := $false$
15 **else**
16 | res := $true$
17 **end if**
18 **return** { res }
End

Le premier est basé sur un calcul de la date de visite au plus tard du premier sommet et est illustré par la *Figure 3.23*. Les dates de début de service de chaque sommet étant déjà calculées par la fonction cut_EBST, la méthode remonte jusqu'au nœud racine pour calculer le temps de service au plus tard du premier nœud, correspondant à 15 dans l'exemple. Ainsi, la différence entre le temps de service au plus tôt du nœud courant et le temps de service au plus tard du nœud racine donne le temps de trajet minimal T_{min} du véhicule pour visiter l'ensemble des sommets déjà insérés dans l'ordre courant. Si $T_{min} > T$ alors il est impossible pour le véhicule de respecter la durée de trajet maximale T suite à l'insertion du sommet et la branche du nœud courant est coupée.

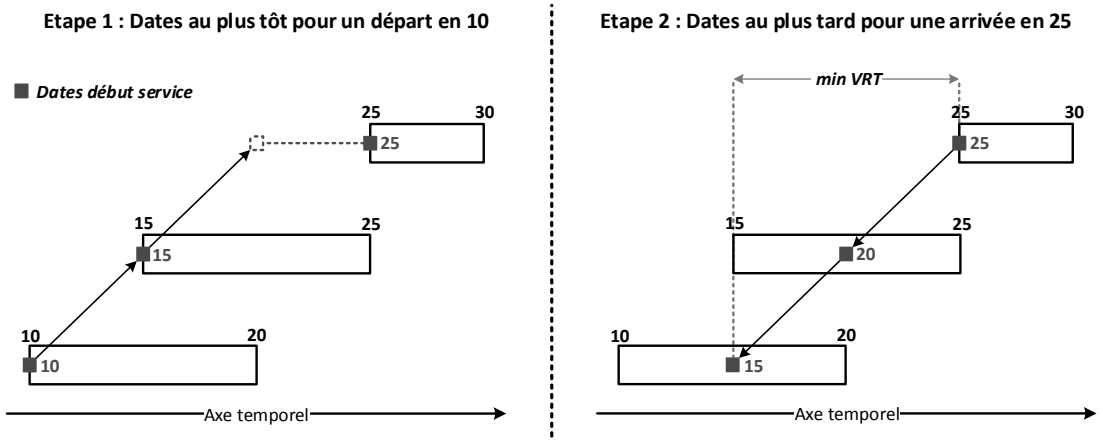


Figure 3.23 : Utilisation partielle de la méthode d'évaluation de (Cordeau et Laporte, 2003) pour déterminer lors de l'insertion d'un nouveau sommet, le temps de trajet minimal du véhicule le visitant

Le second test se base sur l'utilisation du nœud ayant la plus grande borne inférieure ayant pour valeur e_{max}. Lorsque chaque nœud est parcouru pour connaître sa date de début de service au plus tard, il est également possible de connaître la marge entre cette dernière et la borne supérieure du sommet. Ces marges sont illustrées par les valeurs m_1, m_2 et m_3 de la *Figure 3.24*. Puisque le sommet inséré par le nœud courant n'est pas nécessairement le dernier de la tournée, il est possible de décaler la tournée sans modifier le *riding time* maximal du véhicule, le décalage maximal est la valeur minimale des marges. Le sommet non inséré ayant la plus grande borne inférieure apparaissant obligatoirement dans la tournée, la validité de l'équation suivante peut être testée, indiquant si l'insertion future de ce nœud violera ou non la contrainte de *riding time* maximal du véhicule :

$$e_{max} - minVRT - \min\{m_i\} \leq T$$

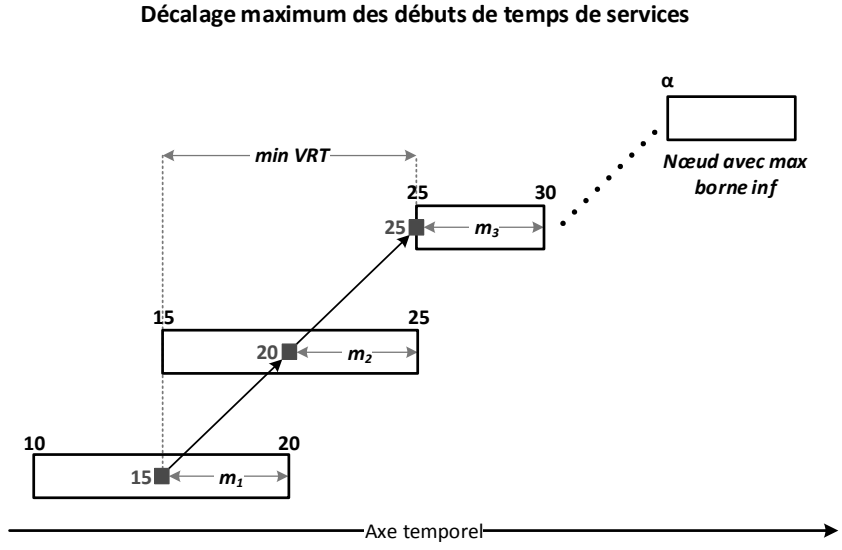


Figure 3.24 : Calcul du décalage maximal de la tournée en cours sans modifier la valeur du temps de trajet du véhicule

g) Temps de trajet des clients

La dernière coupe est effectuée par la fonction *cut_ECRT* (*Estimated Client Riding Time* – ECRT), vérifiant dans l'ordre de visite courant, qu'aucune contrainte de temps de trajet

maximum n'est violée pour chaque client. Cette fonction est appelée pour chaque nœud de l'arbre de recherche et est détaillée par l'*Algorithme 26*.

Algorithme 26 cut_ECRT

Input parameter

$currentNode$: Current node of the decision tree

Output variable

res : Boolean value returning $true$ if there is no client riding time violation, $false$ otherwise

Local variables

$clientRT$: Temporary real containing the riding time of a client
$client$: Temporary client variable
$size, i$: Temporary integer variables
$current$: Temporary node variable
$sumWT$: Temporary real variable containing the sum of waiting time
$launchCordeau$: Boolean variable that is set to $true$ if Cordeau evaluation must be launch, $false$ otherwise

Begin

```
1  for all client in currentNode.previous.clientInV do
2  |  currentNode.clientInV.add(client)
3  end for
4  res := true
5  launchCordeau := false
6  if not currentNode.clientInV.size() = 0 then
7  |  size := currentNode.clientInV.size()
8  |  i := 0
9  |  while res = true and in i < size do
10 |  |  client := currentNode.clientInV[i]
11 |  |  clientRT := compute_RT_margin(client, currentNode)
12 |  |  if clientRT < 0 then
13 |  |  |  launchCordeau := true
14 |  |  |  current := currentNode.previous
15 |  |  |  sumWT := currentNode.waiting
16 |  |  |  while not current.customerNode.get_client() = client then
17 |  |  |  |  sumWT := sumWT + current.waiting
18 |  |  |  |  current := current.previous
19 |  |  |  end while
20 |  |  |  if sumWT < clientRT then
21 |  |  |  |  res := false
22 |  |  |  end if
23 |  |  end if
24 |  |  |  i := i + 1
25 |  end while
26 |  if res := true and launchCordeau := true then
27 |  |  res := Cordeau_evaluation(currentNode)
28 |  end if
29 end if
30 if currentNode.customerNode.is_pickup() then
31 |  currentNode.clientInV.add(currentNode.customerNode.get_client())
32 else if currentNode.customerNode.is_delivery() then
33 |  currentNode.clientInV.remove(currentNode.customerNode.get_client())
34 end if
35 return { res }
```

End

La méthode est divisée en une série de tests permettant d'éviter au maximum l'appel à la fonction $Cordeau\_evaluation$. Cette dernière est de complexité $O(n^2)$ (n étant le nombre de sommets dans l'ordre de visite testé) et implémente l'algorithme de (Cordeau et Laporte, 2003) pour vérifier qu'aucune violation de la contrainte des temps de trajet des clients n'apparaisse suite à l'insertion d'un nouveau sommet.

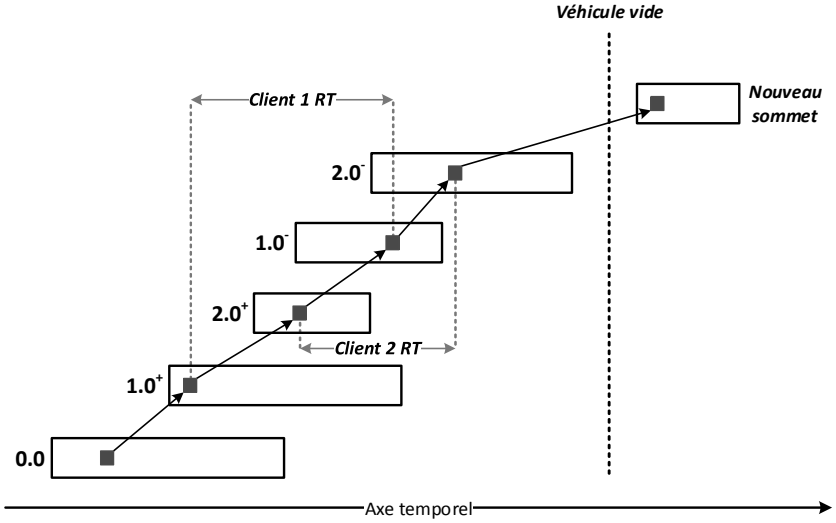


Figure 3.25 : Insertion d'un nouveau sommet lorsque le véhicule est vide : aucune contrainte de temps de trajet maximal ne peut être violée

Le premier test est effectué à la ligne 6, en vérifiant si des clients sont présents dans le véhicule lors de l'insertion du nouveau sommet : si ce test est valide, alors les clients dans le véhicule voient la durée de leur trajet augmenter et des vérifications supplémentaires sont nécessaires. Sinon, aucun temps de trajet n'augmente et aucune contrainte n'est violée comme illustré sur la *Figure 3.25* : le nouveau sommet est ajouté dans un véhicule vide donc aucun temps de trajet de client n'augmente.

La seconde vérification est effectuée si le premier test est valide. Pour chaque client présent dans le véhicule lors de l'insertion du nouveau sommet elle calcule la durée de son trajet suite à l'ajout du nouveau sommet (ligne 11) et vérifie qu'il ne dépasse pas la durée maximale de trajet (ligne 12) comme illustré sur la *Figure 3.26*.

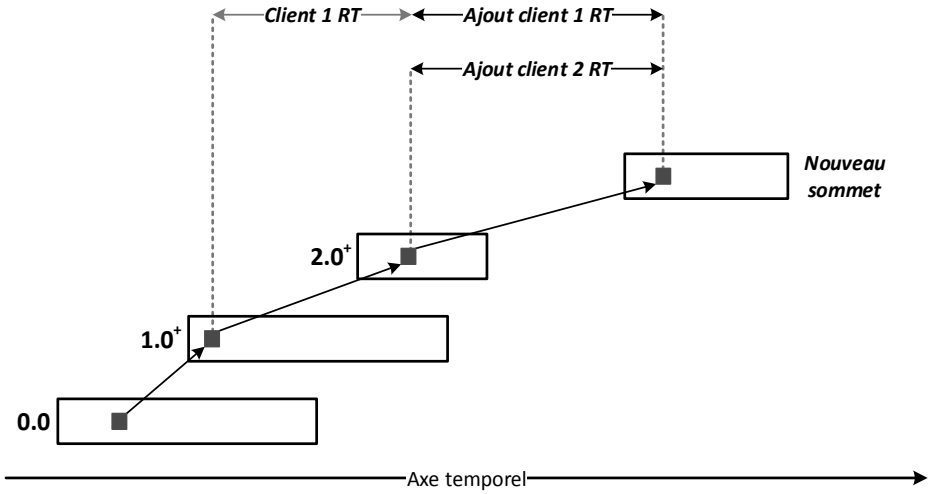


Figure 3.26 : Calcul de la nouvelle durée des trajets des clients présent dans le véhicule lors de l'insertion d'un nouveau sommet

Si la durée de trajet maximale d'un client est dépassée, l'ordre de visite n'est pas nécessairement invalide grâce à la présence éventuelle des temps d'attente entre chaque sommet insérés. Comme illustré sur la *Figure 3.27*, la suppression du temps d'attente w_1 et la réduction du temps d'attente w_2 induits par les déplacements des débuts de temps de service des nœuds $0.0, 1.0^+$ et 2.0^+ permet au client 1 de respecter sa contrainte de durée de trajet maximale. Le calcul est effectué ligne 14 à 25 de l'*Algorithme 26* en vérifiant que la somme des temps d'attente est supérieure au dépassement de la durée de trajet maximale : si elle est inférieure, alors les temps d'attentes ne sont pas suffisant et une coupe intervient sur le nœud courant.

Enfin, si les temps d'attentes peuvent compenser les durées de trajet trop importantes des clients, la méthode d'évaluation de (Cordeau et Laporte, 2003) est utilisée pour vérifier si cette compensation est réalisable. En effet, même si les temps d'attentes sont suffisants, l'ordre peut rester invalide comme sur l'exemple de la *Figure 3.28* : le client 1 effectue un trajet de durée 18 et la limite est de 16 unités maximum. Le temps d'attente situé avant le nœud 3.0^+ n'est pas utilisable pour satisfaire la contrainte de temps, bien qu'il soit supérieur au dépassement de la durée de trajet maximale ($5 > 2$). La présence de temps d'attente est donc une condition nécessaire mais non suffisante pour corriger la violation d'une contrainte de durée de trajet maximale. La méthode $Cordeau\_evaluation$ est alors appelée pour établir la validité de l'ordre de visite ou non.

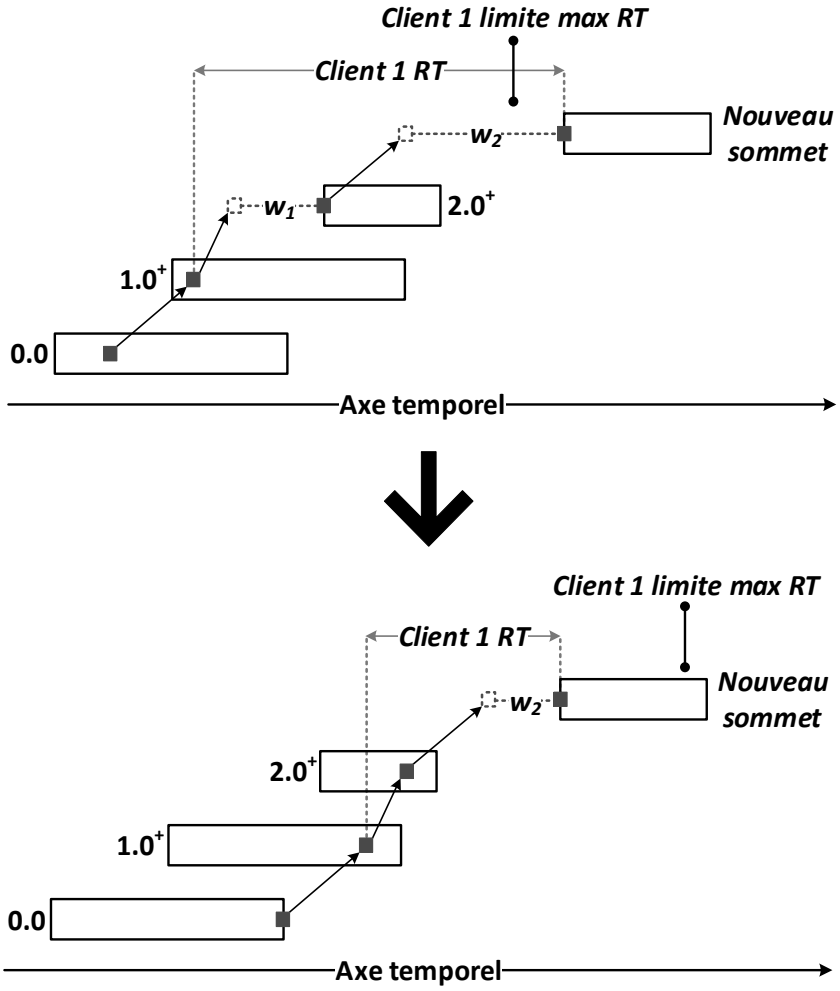


Figure 3.27 : Déplacement des temps de service au plus tôt des sommets 0.0, 1.0^+ et 2.0^+ en utilisant les temps d'attentes w_1 et w_2 permettant au client 1 de valider sa contrainte de durée de trajet maximale

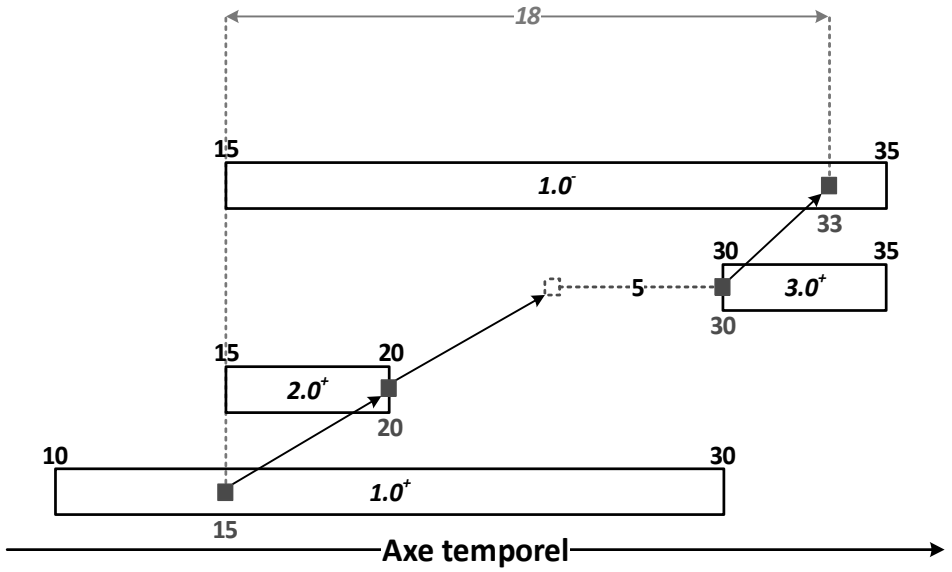


Figure 3.28 : Malgré le temps d'attente de 5 avant le nœud 3.0^{+}*, le client 1 ne peut pas baisser son trajet de durée 18 sous la limite maximale de 16.*

3.5 Enrichissement de la population en résolvant le problème de couverture par ensembles avec poids par PLNE

La dernière étape de la méthode de résolution proposée comporte l'ajout dans la population, à la fin de chaque génération de l'algorithme génétique, d'un individu représentant une solution calculée grâce à l'ensemble des tournées sauvegardées. En effet, lorsque l'algorithme de *Branch and Bound* donne l'ordre de visite optimal d'une tournée, cette dernière est sauvegardée dans une structure de type *HashMap*, évitant ainsi de recalculer l'ordre de visite optimal si la tournée réapparaît dans d'autres individus de l'algorithme génétique. Ainsi, une autre méthode pour créer des solutions valides consiste à sélectionner certaines tournées parmi celle sauvegardées et à les rassembler au sein d'une même solution.

La création de nouvelles solutions en utilisant cette méthode présente de nombreux avantages :

- La sélection des tournées parmi celles disponibles est optimale pour former une nouvelle solution ;
- La totalité des tournées créées depuis le lancement de l'algorithme sont accessibles, contrairement à la partie algorithme génétique/*Branch and Bound* qui n'a accès qu'aux tournées de la génération en cours ;
- La création de la solution est rapide en utilisant la programmation linéaire en nombres entiers ;
- La capacité à améliorer la meilleure solution courante augmente avec le temps : plus le nombre de tournées optimales sauvegardées augmente, plus le nombre de possibilités de création de nouvelles solutions augmente également.

La sélection des meilleures tournées pour former de nouvelles solutions, est en fait une résolution du problème de couverture par ensembles avec poids (WSCP) décrit dans la section suivante.

3.5.1 Problème de couverture par ensembles avec poids

Le problème de couverture par ensembles (*Set Cover Problem* – SCP) est un problème d'informatique théorique faisant parti des 21 problèmes NP-complets de Karp (Karp, 1972).

Soit l'univers $U = \{1,2,3, \dots, n\}$, et $S = \{S_1, S_2, S_3 \dots, S_m\}$ une famille de sous-ensembles dont l'union est égale à l'univers : résoudre le SCP dans sa version optimisation consiste à trouver la plus petite sous-famille de S dont l'union est égale à l'univers. Il existe une version généraliste du problème utilisée dans cette thèse, appelé *Weighted Set Cover Problem* (WSCP). Dans celle-ci, un poids P_i est affecté à chaque ensemble i de S : une solution optimale du WSCP n'est pas la plus petite sous-famille de S formant une couverture de U, mais la sous-famille de poids minimal de S couvrant U. Ce problème est NP-difficile.

Pour illustrer le SCP, l'exemple suivant est proposé :

- $U = \{1,2,3,4,5,6\}$;
- $S = \{\{1,2\},\{2,4\},\{3,5\},\{3,4,5,6\}\}$.

Une première solution C évidente est $C = S$. Cependant, un plus petit nombre d'éléments peut être sélectionné pour couvrir U : $C = \{\{1,2\},\{3,4,5,6\}\}$.

En revanche, dans le cas du WSCP, la meilleure solution n'est pas nécessairement celle avec la plus petite sous-famille comme le montre l'exemple ci-dessous :

- $U = \{1,2,3,4,5,6\}$;
- $S = \{\{1,2\},\{2,4\},\{3,5,6\},\{3,4,5,6\}\}$;
- $P_{\{1,2\}} = 2, P_{\{2,4\}} = 1, P_{\{3,5,6\}} = 3, P_{\{3,4,5,6\}} = 5$.

Dans cet exemple, la solution avec la plus petite sous-famille $C = \{\{1,2\},\{3,4,5,6\}\}$ a un poids de 7 alors que la solution optimale $C = \{\{1,2\},\{2,4\},\{3,5,6\}\}$ a un poids de 6, mais avec un élément supplémentaire. Ainsi, la sous-famille de poids minimal n'est pas obligatoirement corrélée avec la sous-famille de taille minimale.

Dans le DARP-PV-AN, l'univers U représente les clients et l'ensemble S représente toutes les tournées optimales calculées et sauvegardées par le *Branch and Bound* dans la *HashMap*. Pour fournir une solution s valide au DARP-PV-AN, $s \subseteq S$ doit être une couverture exacte de U de poids minimal, c'est-à-dire une partition de poids minimal de U (une partition de U étant un ensemble d'éléments disjoints 2 à 2 et dont l'union est U). La recherche de la couverture exacte optimale dans le cadre du WSCP est une résolution du *Set-Partitioning Problem* (SPP).

Dans le cadre de la méthode implémentée, le WSCP est résolu sans obligation de couverture exacte : en effet, si un client c_i est présent dans plusieurs tournées sélectionnées, par exemple A et B, alors la solution ou c_i n'est présent que dans A ou dans B est obligatoirement de coût inférieur, et sera donc choisie par le solveur. Cependant, s'il n'existe pas de tournées A et B sans le client c_i encore découvertes, il est possible d'avoir des clients présents dans plusieurs tournées. Dans ce cas, une méthode de correction supprimant les duplications de clients dans la solution fournie par le solveur est expliqué dans la section 3.5.4. Le choix de la résolution du WSCP, et non du SPP, est dû à la vitesse de résolution supérieure du WSCP, permettant ainsi une convergence plus efficace de la méthode de résolution hybride globale.

Pour résoudre le WSCP, une résolution par programmation linéaire en nombres entiers est utilisée. Le modèle décrit dans les sections suivantes est ensuite résolu par CPLEX à chaque

génération de l'algorithme génétique : l'individu représentant la solution créée est ensuite inséré dans la population courante avant la génération suivante.

3.5.2 Données du modèle

U ensemble des clients

S ensemble des tournées optimales valides

$a_{ij} = \begin{cases} 1 \text{ si le client } j \in U \text{ est couvert par la tournée } i \in S \\ 0 \text{ sinon} \end{cases}$

d_i distance de la tournée i, $i \in S$

3.5.3 Définition des variables

Le modèle utilise un ensemble de variables binaires x_i représentant les variables de décision responsable de la sélection des différentes tournées pour créer une couverture de poids minimal de U.

$x_i = \begin{cases} 1 \text{ si la tournée } i \in S \text{ est sélectionnée} \\ 0 \text{ sinon} \end{cases}$

3.5.4 Contraintes et fonction objectif

Un seul ensemble de contraintes est proposé, obligeant chaque client appartenant à U à apparaître au moins une fois dans les tournées sélectionnées :

$$\sum_{i \in S} x_i \times a_{ij} \geq 1, \forall j \in U \quad (1)$$

L'objectif du modèle est de minimiser la somme des distances parcourues par les tournées sélectionnées :

$$\min \sum_{i \in S} d_i \times x_i \quad (2)$$

Lorsque la solution optimale est fournie par CPLEX, une vérification sur le nombre de tournées publiques présentes est effectuée : si le nombre est inférieur ou égal au nombre de véhicules publics disponibles, alors l'individu représentant la solution est inséré dans la population de l'algorithme génétique à la fin de la génération courante. Sinon aucun ajout n'est effectué dans la population. Cette contrainte n'est pas ajoutée dans le modèle pour une résolution plus rapide de ce dernier. De plus, la minimisation du nombre de tournées publiques est une conséquence de la minimisation des distances parcourues : ainsi, la contrainte est rarement violée. Une seconde vérification est effectuée sur la fréquence d'apparition des clients dans les tournées : puisque le WSCP est résolu et non le SPP, certains clients peuvent apparaitre dans plusieurs tournées au sein d'une même solution. Dans ce cas, le client est supprimé de toutes les tournées sauf une, choisie aléatoirement : l'ensemble des tournées ou le client a été supprimé n'ayant pas encore été découvertes, ces dernières sont évaluées par la procédure de *branch and bound* puis sauvegardées dans la *HashMap*.

Dans la section suivante présentant les résultats obtenus lors de l'application de la méthode hybride proposée, une comparaison de l'algorithme avec et sans la méthode de sélection présenté dans cette section est effectuée et montre l'apport de cette dernière.

3.6 Résultats

Dans cette partie, les résultats présentés sont réalisés sur les nouvelles instances proposées dans le chapitre précédent pour la résolution du DARP-PV-AN. La méthode de résolution hybride proposée dans ce chapitre est comparée à l'algorithme ELS du chapitre précédent.

La totalité des expériences ont été réalisées via SLURM sur un serveur de calcul sous Linux équipé de 3To de ram et d'un Intel Xeon E7-8890v3@2,5GHz avec un seul thread activé. Le solveur CPLEX en version 12.6 a été utilisé pour la résolution du problème de couverture par ensembles avec poids à chaque génération de la méthode hybride. Considérant la méthode d'évaluation proposée par (Dongarra et al., 2011), la vitesse de la machine est d'environ 3,33 GFlops et l'algorithme hybride proposé a été codé en Java.

3.6.1 Évaluation de l'algorithme hybride sur les nouvelles instances

L'évaluation de la méthode hybride proposée dans ce chapitre a été effectuée sur les nouvelles instances représentant le DARP-PV-AN et les résultats sont présentés dans le *Tableau 3.1*. Le tableau est divisé en 6 parties principales, chacune correspondant à un groupe d'instances composées des mêmes clients mais avec une composition de véhicules publics/privés différente. La colonne n indique le nombre de clients, N^T représente le nombre total de sommets, K le nombre de véhicules publics, K' le nombre de véhicules privés et TT le temps total en secondes pour résoudre l'instance. La colonne Obj représente le coût de la meilleure solution trouvée par chaque algorithme et T^* le temps nécessaire en secondes pour l'atteindre. Le pourcentage d'écart entre la meilleure solution de l'ELS et de l'algorithme génétique hybride a été calculé comme suit :

$$Gap\% = \frac{Obj_{GA} \times 100}{Obj_{ELS}} - 100$$

La méthode de résolution hybride fournie pour 33 instances sur 35 des solutions avec un objectif inférieur ou égal à l'algorithme ELS. Le gain sur la fonction objectif monte jusqu'à 15,49% sur l'instance PCD_120_20VP avec une valeur moyenne de 3,92%. Le temps nécessaire pour trouver la meilleure solution est également inférieur lors de l'utilisation de l'algorithme génétique hybride pour 27 instances sur 35.

Une comparaison de la qualité des solutions entre l'algorithme ELS du chapitre précédent et l'algorithme génétique hybride a été effectuée selon les trois critères définis par (Cordeau et Laporte, 2003) : TRT (*total riding time*), TWT (*total waiting time*) et TD (*total duration*). Comme le montre la *Figure 3.29*, les critères TRT et TD offrent des valeurs très proches selon la méthode de résolution, avec un avantage pour l'algorithme génétique hybride. En revanche, les solutions proposées par l'ELS fournissent des temps d'attentes inférieurs à celles de l'algorithme génétique : cependant, ces solutions sont d'un coût supérieur. Comme pour le chapitre précédent, on observe une amélioration de tous les critères lorsque les véhicules privés sont disponibles, et ceci quel que soit la méthode de résolution employée.

Tableau 3.1 : Comparaison de l'ELS et de la méthode hybride sur les nouvelles instances

						ELS		Hybrid GA		
Name	n	N^T	***K***	***K'***	***TT(s)***	***Obj***	***T*(s)***	***Obj***	***Gap%***	***T*(s)***
PCD_20_0VP	10	61	4	0	600.00	811.35	0.00	811.35	0.00	1.11
PCD_20_2VP	10	61	4	2	600.00	730.65	300.00	715.54	-2.07	3.87
PCD_20_4VP	10	61	4	4	600.00	730.65	300.00	715.54	-2.07	5.99
PCD_20_6VP	10	61	4	6	600.00	658.24	300.60	643.13	-2.29	4.49
PCD_20_10VP	10	61	4	10	600.00	435.52	330.60	388.96	-10.69	1.61
PCD_40_0VP	20	125	8	0	900.00	1109.59	6.00	1109.59	0.00	9.43
PCD_40_2VP	20	125	8	2	900.00	1028.41	455.40	1028.41	0.00	16.12
PCD_40_6VP	20	125	8	6	900.00	747.73	433.20	741.63	-0.82	10.15
PCD_40_8VP	20	125	8	8	900.00	667.43	454.80	667.43	0.00	20.65
PCD_40_10VP	20	125	8	10	900.00	627.54	679.20	614.79	-2.03	31.58
PCD_40_20VP	20	125	8	20	900.00	512.24	458.40	487.85	-4.76	11.13
PCD_60_0VP	30	191	20	0	1200.00	1525.62	846.60	1521.37	-0.28	1077.96
PCD_60_2VP	30	191	20	2	1200.00	1479.83	924.00	1479.82	0.00	115.82
PCD_60_8VP	30	191	20	8	1200.00	1127.55	957.00	1054.46	-6.48	92.17
PCD_60_10VP	30	191	20	10	1200.00	1133.38	663.60	1031.14	-9.02	89.75
PCD_60_16VP	30	191	20	16	1200.00	946.43	1101.00	899.72	-4.94	796.93
PCD_60_30VP	30	191	20	30	1200.00	911.15	612.60	775.40	-14.90	809.03
PCD_80_0VP	40	243	20	0	1500.00	2032.86	723.60	2009.90	-1.13	396.44
PCD_80_2VP	40	243	20	2	1500.00	1904.62	1257.60	1871.82	-1.72	447.45
PCD_80_10VP	40	243	20	10	1500.00	1792.16	1444.20	1674.83	-6.55	1053.71
PCD_80_14VP	40	243	20	14	1500.00	1679.32	1281.60	1624.63	-3.26	85.87
PCD_80_20VP	40	243	20	20	1500.00	1452.10	846.00	1421.29	-2.12	391.31
PCD_80_40VP	40	243	20	40	1500.00	1323.59	853.20	1198.17	-9.48	1432.43
PCD_100_0VP	50	307	50	0	1800.00	2384.21	1405.20	2475.91	3.85	911.05
PCD_100_4VP	50	307	50	4	1800.00	2303.97	1528.20	2247.32	-2.46	1353.06
PCD_100_14VP	50	307	50	14	1800.00	1991.67	1510.20	1925.30	-3.33	1017.70
PCD_100_18VP	50	307	50	18	1800.00	1921.00	1417.80	1903.03	-0.94	761.86
PCD_100_26VP	50	307	50	26	1800.00	1858.81	1323.00	1820.66	-2.05	903.25
PCD_100_50VP	50	307	50	50	1800.00	1842.80	1055.40	1597.78	-13.30	1044.82
PCD_120_0VP	60	371	50	0	2100.00	2817.98	2079.60	2946.83	4.57	1980.90
PCD_120_4VP	60	371	50	4	2100.00	2608.24	1240.80	2702.30	3.61	2148.09
PCD_120_16VP	60	371	50	16	2100.00	2300.61	2097.60	2134.50	-7.22	1047.46
PCD_120_20VP	60	371	50	20	2100.00	2453.28	1538.40	2073.20	-15.49	1830.19
PCD_120_30VP	60	371	50	30	2100.00	2174.94	1894.80	2042.93	-6.07	1085.00
PCD_120_60VP	60	371	50	60	2100.00	2114.03	1518.60	1820.70	-13.88	1980.90
Average									-3.92	

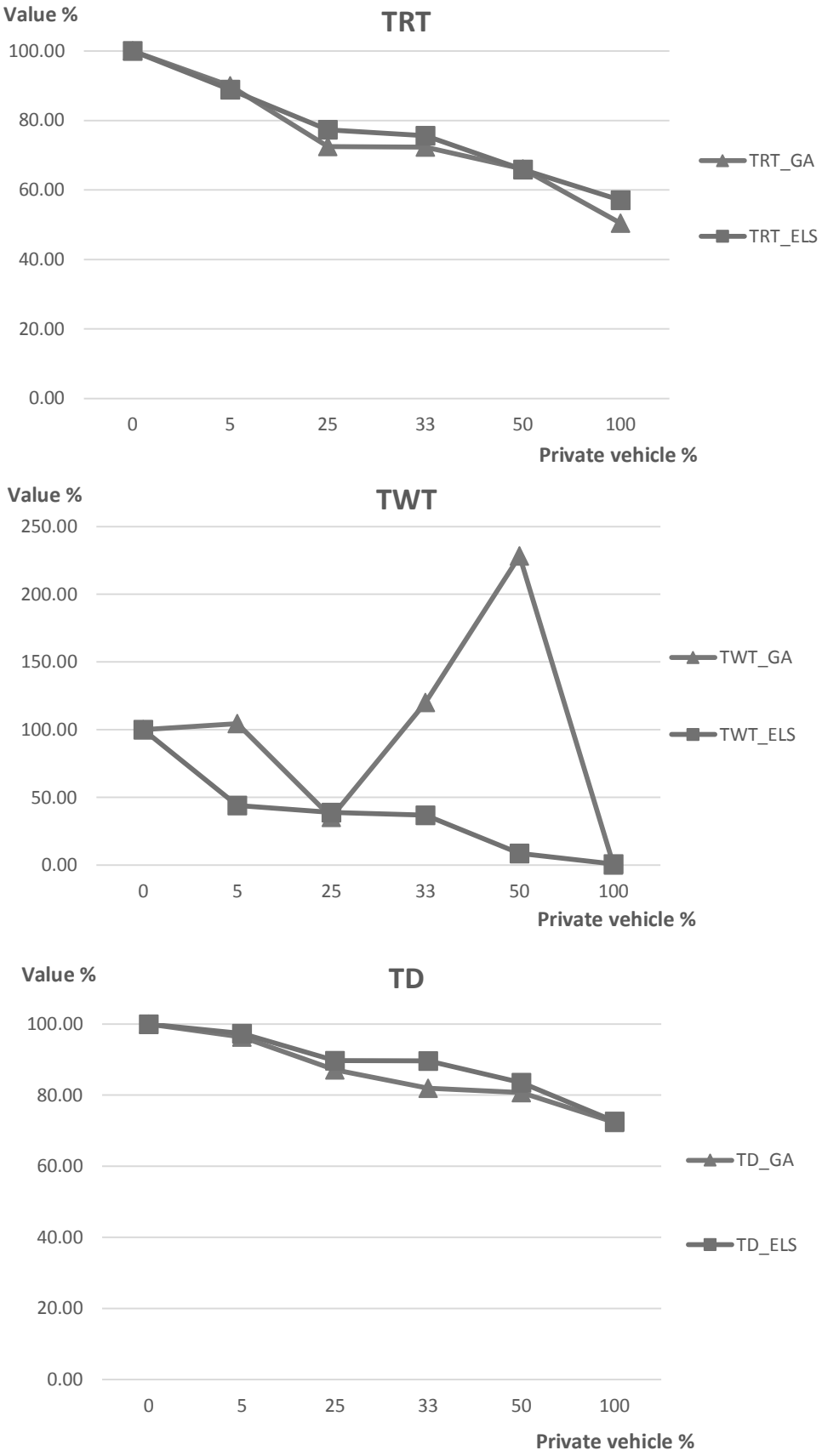


Figure 3.29 : Comparaison de la qualité de service moyenne selon la définition de (Cordeau et Laporte, 2003) suivant les méthodes de résolution (ELS ou algorithme génétique hybride)

Une seconde comparaison des deux algorithmes a été effectuée sur le nombre d'arcs utilisés en fonction du nombre de véhicules privés disponibles. Les résultats sont donnés sur la *Figure 3.30* : on peut constater une baisse du nombre d'arcs utilisés lors de l'utilisation de l'algorithme génétique hybride avec un écart allant jusqu'à 12,40% d'arcs en moins lorsque 100% des véhicules privés sont disponibles. Cette différence s'explique par la meilleure valeur des solutions obtenues par l'algorithme génétique hybride : ces dernières parcourent une distance plus courte que les solutions proposées par l'algorithme ELS et donc utilisent en moyenne moins d'arcs.

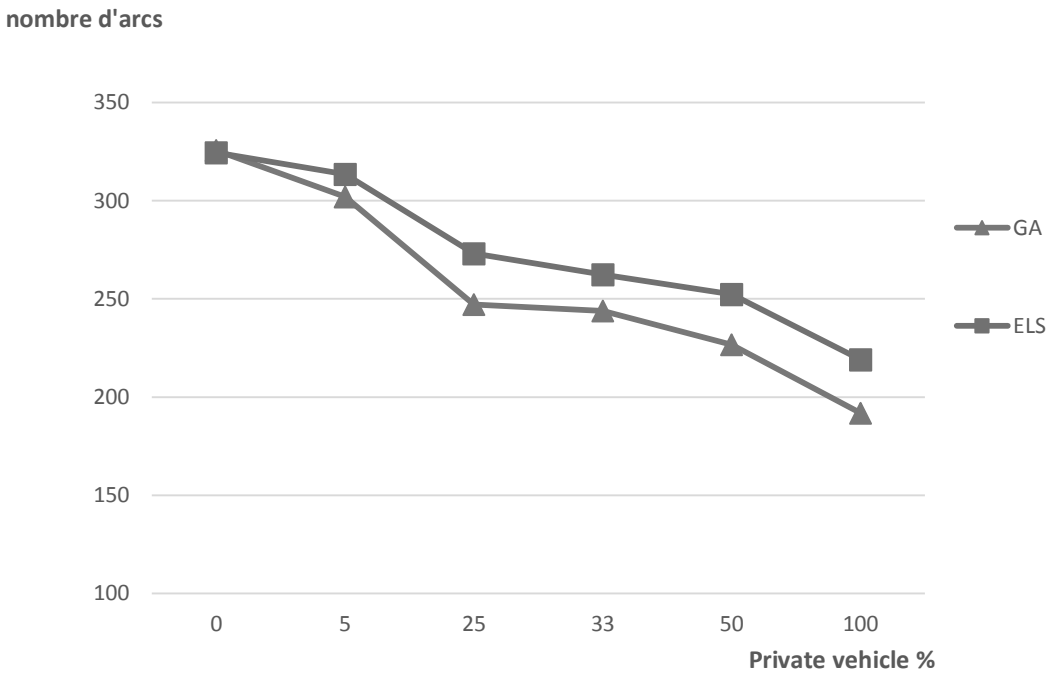


Figure 3.30 : Comparaison de la moyenne du nombre d'arcs utilisés

Comme dans le chapitre précédent, l'analyse suivante évalue dans chaque solution, la moyenne des écarts entre la distance de transport optimal d'un client (c'est-à-dire sans détour entre son *pickup* et son *delivery*) et la distance de transport réel du client. La *Figure 3.31* montre la différence entre les deux méthodes de résolution : on constate que l'algorithme génétique hybride fournit des solutions avec une déviation moyenne plus faible comparée aux solutions de l'algorithme ELS proposé dans le chapitre précédent. Lorsque 100% des véhicules privés sont disponibles, la longueur moyenne du transport d'un client correspond à 117,89% du chemin le plus court pour l'algorithme génétique hybride contre 129,29% pour l'ELS. Une nouvelle fois, cette différence s'explique par les valeurs des solutions trouvées par chaque algorithme : l'algorithme génétique hybride fournissant des solutions avec une distance plus courte, les déviations pour chaque client diminuent également.

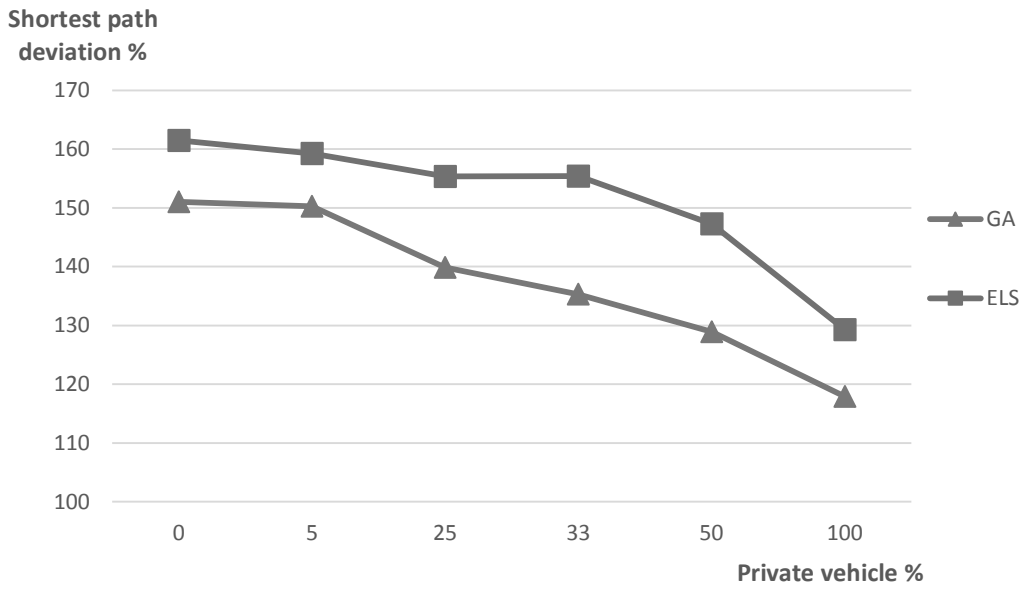


Figure 3.31 : Comparaison de la déviation moyenne du plus court chemin pour chaque requête

3.6.2 Comparaison des vitesses de convergence

L'analyse suivante porte sur la comparaison des vitesses de convergence des deux algorithmes : à chaque pourcentage de temps d'une instance est associé l'écart entre la meilleure solution courante et la meilleure solution finale de cette instance. Par exemple, sur une instance bénéficiant d'un temps de résolution de 60 secondes, et dont la meilleure solution trouvée aura une valeur de 100, si au temps 30 la solution courante à une valeur de 170 alors elle a une déviation de 70% à 50% du temps. Comme le montre la *Figure 3.32*, l'algorithme génétique hybride converge plus rapidement vers son optimum : les meilleures solutions de l'ELS sont atteinte en moyenne avec seulement 10% du temps par l'algorithme génétique. Les solutions initiales sont éloignées en moyenne de 68,44% pour l'ELS et de 38,19% pour l'algorithme génétique hybride.

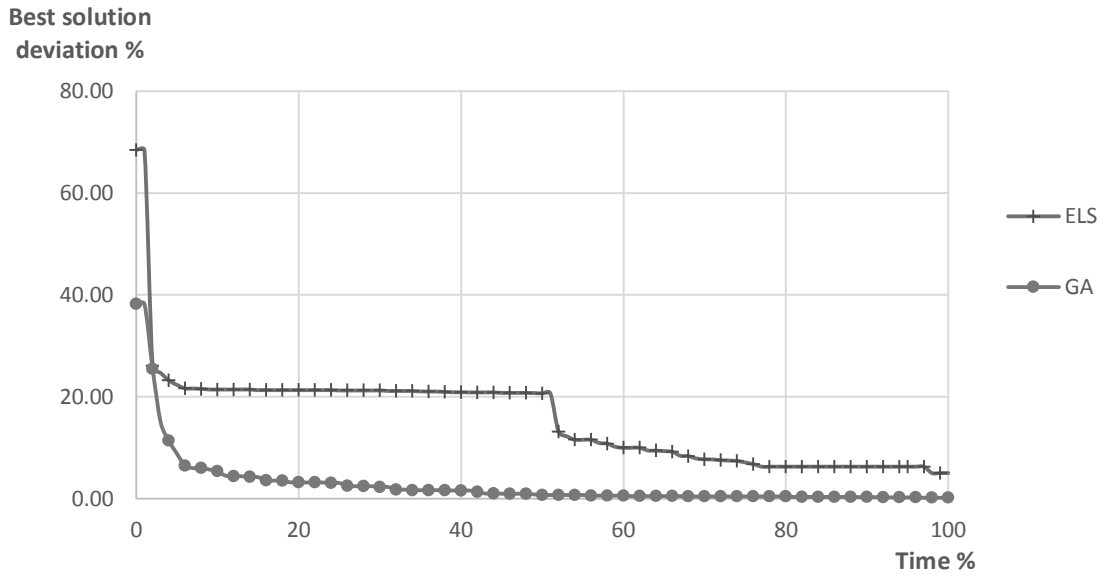


Figure 3.32 : Comparaison des vitesses de convergence des deux méthodes de résolution

3.6.3 Comparaison de l'algorithme génétique avec et sans enrichissement de population

La comparaison suivante montre l'utilité de la partie hybride de l'algorithme génétique présenté dans la section 3.5 visant à enrichir la population en sélectionnant les meilleures tournées créées depuis le lancement de l'algorithme jusqu'à la génération courante, afin de créer une nouvelle solution. Le but de cette méthode est d'améliorer les solutions trouvées mais également la vitesse de convergence de la métaheuristique hybride proposée. Ainsi, une comparaison avec et sans l'utilisation de cette méthode (en fixant le paramètre d'activation respectivement à *true* ou *false*) a été effectuée et représenté par la *Figure 3.33*. On observe une convergence plus rapide lors de l'utilisation de l'enrichissement de la population : l'écart lors des premières générations est très important (132,01% sans et 37,36% avec) et il faut en moyenne 12% du temps à la méthode avec enrichissement pour fournir de meilleures solutions que la version sans enrichissement. Les solutions fournies sans enrichissement sont également plus éloignées des solutions trouvées avec enrichissement, avec un écart moyen de 4,02% sur la fonction objectif.

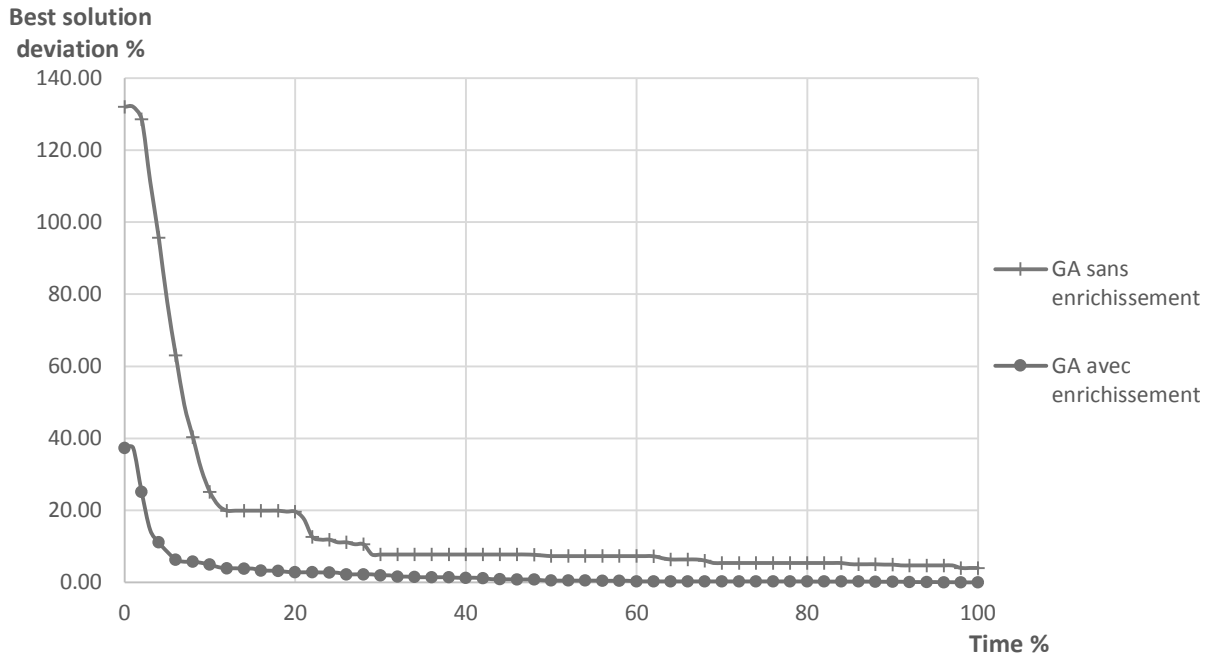


Figure 3.33 : Comparaison des vitesses de convergence de l'algorithme génétique avec et sans enrichissement de la population

Le *Tableau 3.2* illustre le temps moyen passé par CPLEX dans l'algorithme génétique hybride. La colonne $Name$ regroupe au sein d'un même nom les instances avec le même nombre de clients (PCD_20 représente les instances avec 10 clients), NB_{Turn} représente le nombre de tours moyen parmi lesquels CPLEX doit faire une sélection afin de déterminer une nouvelle solution, T_{CPLEX} est le temps moyen passé par CPLEX pour résoudre tous les problèmes de couverture par ensembles avec poids par instance et la colonne $T\%$ représente le pourcentage du temps total disponible pour résoudre une instance utilisé par CPLEX. Avec un temps moyen de 10,16% du temps total, le temps d'utilisation de CPLEX est faible dans l'algorithme génétique hybride. La vitesse de CPLEX pour résoudre un problème de couverture par ensembles avec poids est inférieur à 0,3 seconde sur une instance ayant jusqu'à 55000 tours. L'utilisation de l'enrichissement de population apporte donc une amélioration importante sur la vitesse de convergence et la valeur des solutions trouvées, tout en ayant un faible impact sur l'utilisation du temps octroyé pour la résolution d'une instance.

Tableau 3.2 : Temps utilisé par CPLEX dans l'algorithme génétique hybride

Name	*TT(s)*	NB_{Turn}	T_{CPLEX}*(s)*	*T%*
PCD_20	600.00	1873.40	53.89	8.98
PCD_40	900.00	12912.67	155.25	17.25
PCD_60	1200.00	14439.17	137.11	11.43
PCD_80	1500.00	29375.33	229.24	15.28
PCD_100	1800.00	20944.33	115.06	6.39
PCD_120	2100.00	17659.50	33.74	1.61

3.7 Conclusion

Pour résoudre le problème du transport à la demande avec véhicules privés et sommets alternatifs (DARP-PV-AN), une méthode en trois étapes a été proposée. La première étape consiste à affecter les clients aux véhicules avec une métaheuristique de type algorithme génétique. Ensuite, l'ordre optimal de visite pour chaque véhicule est calculé lors de la procédure d'évaluation par un algorithme de *Branch and Bound*. Enfin, une dernière étape consiste à enrichir la population de l'algorithme génétique en créant une nouvelle solution à chaque génération par résolution d'un modèle linéaire avec le solveur CPLEX. Ce modèle représente un problème de couverture par ensembles avec poids consistant à sélectionner un ensemble de tournées parmi celles générées depuis le lancement de l'algorithme, formant ainsi une nouvelle solution.

Pour tester les performances de cette approche, une comparaison a été effectuée avec l'algorithme ELS sur les nouvelles instances, tous deux proposés lors du chapitre précédent. Les résultats montrent que l'algorithme génétique hybride fourni de meilleures solutions dans des temps plus courts. La diminution de la valeur de la fonction objectif entraîne également une diminution du nombre d'arcs parcourus par les solutions de l'algorithme génétique tout comme un rapprochement du trajet de chaque client vers son trajet minimal théorique.

La comparaison des vitesses de convergence avec et sans la méthode d'enrichissement de la population par résolution du WSCP, montre l'impact important de cette dernière dans l'algorithme proposé. Son utilisation permet non seulement des solutions de meilleure qualité dès la première génération, mais également une convergence plus rapide de l'algorithme vers de meilleures solutions finales. De plus, son utilisation est très rapide et ne représente qu'un faible pourcentage de la totalité du temps de résolution pour chaque instance.

Des améliorations pourraient être apportées sur la partie *Branch and Bound* de l'algorithme notamment sur la recherche de bornes inférieures de meilleure qualité. De telles améliorations permettraient la résolution à l'optimum de tournées plus grandes, pour résoudre des instances composées de plus de 60 clients en un temps acceptable.

L'utilisation de véhicules privés apporte donc une amélioration des solutions mais également de leur qualité de service. Cependant, ces véhicules ne sont pas identiques aux moyens publics, et leur ratio d'utilisation coût/clients peut être potentiellement supérieur. Dans le chapitre suivant, l'estimation du coût monétaire d'une solution est prise en compte en plus de sa distance dans le cadre d'une approche bi-objectif. Le but étant est de proposer une méthode de résolution fournissant un ensemble de solutions intéressantes à la fois sur l'aspect financier, sur la distance parcourue et sur la qualité de service proposée aux clients.

Références

Beasley, J., 1983. Route first—Cluster second methods for vehicle routing. Omega 11, 403‑408. https://doi.org/10.1016/0305-0483(83)90033-6

Bodin, L., Golden, B., 1981. Classification in vehicle routing and scheduling. Networks 11, 97‑108. https://doi.org/10.1002/net.3230110204

Borůvka, O., 1926. O jisíém problému minimálním. 24.

Chassaing, M., 2015. Problèmes de transport à la demande avec prise en compte de la qualité de service. Clermont-Ferrand.

Chassaing, M., Duhamel, C., Lacomme, P., 2016. An ELS-based approach with dynamic probabilities management in local search for the Dial-A-Ride Problem. Eng. Appl. Artif. Intell. 48, 119‑133. https://doi.org/10.1016/j.engappai.2015.10.002

Chazelle, B., 2000. A minimum spanning tree algorithm with inverse-Ackermann type complexity. J. ACM 47, 1028‑1047. https://doi.org/10.1145/355541.355562

Chu, P.C., Beasley, J.E., 1998. A Genetic Algorithm for the Multidimensional Knapsack Problem. J. Heuristics 4, 63‑86. https://doi.org/10.1023/A:1009642405419

Clarke, G., Wright, J.W., 1964. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Oper. Res. 12, 568‑581. https://doi.org/10.1287/opre.12.4.568

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. https://doi.org/10.1016/S0191-2615(02)00045-0

Dantzig, T., 1930. Number: The Language of Science: A Critical Survey Written for the Cultured Non-Mathematician, Macmillan. ed.

Dongarra, J., Luszczek, P., Feautrier, P., Zee, F.G., Chan, E., Geijn, R.A., Bjornson, R., Dongarra, J., Luszczek, P., Philippe, B., Sameh, A., Philippe, B., Sameh, A., Dongarra, J., Luszczek, P., Steele, G.L., Gustafson, J.L., Dongarra, J., Luszczek, P., Becker, A., Zheng, G., Kalé, L.V., Pingali, K., Carro, M., Hermenegildo, M., Banerjee, U., Wismüller, R., 2011. LINPACK Benchmark, in: Padua, D. (Éd.), Encyclopedia of Parallel Computing. Springer US, Boston, MA, p. 1033‑1036. https://doi.org/10.1007/978-0-387-09766-4_155

Fogel, D.B., 2009. Artificial Intelligence through Simulated Evolution, in: Evolutionary Computation. IEEE. https://doi.org/10.1109/9780470544600.ch7

Goldberg, D.E., 1989. Genetic Algorithms in search, optimisation, and machine learning. Ed. Addison Wesley Publ. Co.

Goldberg, D.E., Lingle, R., 1985. Alleles, loci, and the traveling salesman problem. Proc. Int. Conf. Genet. Algorithms Their Appl. 154‑159.

Holland, J.H., 1975. Adaptation in natural and artificial systems. Ann Arbor Univ. Mich. Press.

Karp, R.M., 1972. Reducibility among Combinatorial Problems, in: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (Éd.), Complexity of Computer Computations. Springer US, Boston, MA, p. 85‑103. https://doi.org/10.1007/978-1-4684-2001-2_9

Kruskal, J.B., 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. Proc. Am. Math. Soc. 7, 48‑48. https://doi.org/10.1090/S0002-9939-1956-0078686-7

Lacomme, P., Prins, C., Ramdane-Cherif, W., 2004. Competitive Memetic Algorithms for Arc Routing Problems. Ann. Oper. Res. 131, 159‑185. https://doi.org/10.1023/B:ANOR.0000039517.35989.6d

Newton, R.M., Thomas, W.H., 1974. Bus routing in a multi-school system. Comput. Oper. Res. 1, 213‑222. https://doi.org/10.1016/0305-0548(74)90047-1

Prim, R.C., 1957. Shortest connection networks and some generalizations. Bell Syst. Tech. J. 1389‑1401.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. 31, 1985‑2002. https://doi.org/10.1016/S0305-0548(03)00158-8

Rechenberg, I., 1965. Cybernetic solution path of an experimental problem. R. Aircr. Establ. Libr. Transl. 1122.

Ulusoy, G., 1985. The fleet size and mix problem for capacitated arc routing. Eur. J. Oper. Res. 22, 329‑337. https://doi.org/10.1016/0377-2217(85)90252-8

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comput. Oper. Res. 40, 475‑489. https://doi.org/10.1016/j.cor.2012.07.018

CHAPITRE 4

Optimisation multi-objectif pour le problème du transport à la demande avec véhicules privés et sommets alternatifs

Objectifs du chapitre :

Ce chapitre concerne la résolution du problème de transport à la demande avec véhicules privés et sommets alternatifs (DARP-PV-AN) dans le cadre d'une approche multicritère. En effet, les moyens de transport privés et publics n'ont pas le même coût financier d'utilisation et de manière générale l'utilisation de moyens de transport privés est plus coûteuse que l'utilisation de moyen de transport public de type bus ou métro. L'objectif est de fournir un algorithme permettant d'obtenir un ensemble de solutions proposant des compromis différents entre la distance parcourue par les véhicules et le coût financier.
Une justification du second objectif est d'abord présentée suivie par une description de l'optimisation multi-objectif puis de l'algorithme hybride proposé. Ensuite, les différentes méthodes de résolution du problème de couverture par ensembles avec poids (WSCP) bi-objectif sont présentées. Enfin, la méthode hybride est testée et les différentes méthodes de résolution du WSCP bi-objectif sont comparées.

4.1 Objectifs pour le DARP-PV-AN

4.1.1 Optimisation multi-objectif dans les problèmes de transport

L'optimisation multi-objectif a été très étudiée au sein des problèmes de transports. Dans l'article de (Jozefowiez et al., 2007), les auteurs présentent un algorithme combinant une méthode évolutionnaire avec un *Branch and Cut* pour résoudre le problème de couverture par tournées (*Covering Tour Problem* – CTP). Le problème de tournées sur arcs avec contrainte de capacité (*Capacitated Arc Routing Problem* – CARP) dans sa version bi-objectif a été résolu par (Lacomme et al., 2006) en utilisant également un algorithme génétique couplé cette fois-ci avec une procédure de recherche locale. D'autres méthodes de résolutions ont été appliquées comme la recherche tabou liée avec une méthode de *Path Relinking* par (Pacheco et Martí, 2006) ou encore l'algorithme de colonies de fourmis adapté aux problèmes multi-objectif par (Pasia et al., 2007). Une étude sur les problèmes de tournées de véhicules multi-objectif a été publiée par (Jozefowiez et al., 2008).

Une des premières approches publiée pour le DARP dans un contexte multi-objectif a été proposée par (Madsen et al., 1995). Dans cet article, les auteurs proposent la minimisation des critères suivant en utilisant une méthode d'agrégation :

- Le temps total parcouru par l'ensemble de la flotte ;

- Le nombre de véhicules utilisés ;
- Le temps d'attente total des clients ;
- La déviation entre le temps de *pickup*/*delivery* demandé par le client et le temps proposé par la solution ;
- Le coût de l'ensemble des tournées.

Une approche plus récente a été proposée par (Parragh et al., 2009) pour la résolution du DARP bi-objectif. Dans cet article les auteurs minimisent la distance parcourue par l'ensemble des véhicules ainsi que le temps de trajet moyen des clients. Pour cela, deux algorithmes sont proposés :

- Le premier consiste en un *Branch and Cut* couplé à une méthode ϵ-contrainte permettant de générer l'ensemble des solutions optimales pour les petites/moyennes instances ;
- Le second est une métaheuristique en deux phases. La première utilise le principe d'agrégation en lançant plusieurs fois un algorithme VNS (*Variable Neighborhood Search*) avec des poids différents pour chaque objectif dans le but de générer les solutions supportées du front de Pareto. La deuxième phase utilise une méthode inspirée du *path-relinking* pour générer les solutions non-supportées.

En 2012, (Zidi et al., 2012) proposent un algorithme de recuit simulé multi-objectif pour le DARP en optimisant le coût et la qualité de service, elle-même composée du temps de trajet des clients et du nombre de sommets visités par le véhicules pendant le trajet d'un client. L'algorithme proposé n'utilise pas l'agrégation des deux objectifs mais une règle de dominance entre les solutions.

Enfin, (Guerriero et al., 2014) proposent également une méthode bi-objectif pour le DARP en optimisant le temps d'attente total de la solution et le temps de trajet maximal effectué par les véhicules. L'algorithme proposé est une méthode hybride cherchant d'abord à déterminer un ensemble de tournées réalisables grâce à un algorithme génétique. Ensuite, un problème de partitionnement bi-objectif est utilisé en couplant une méthode ϵ-contrainte avec un modèle linéaire résolu par le solveur CPLEX. Cette dernière phase assemble donc les meilleures tournées générées dans la première phase pour créer un ensemble de solutions formant un front de Pareto.

La majorité des méthodes de résolutions précédentes utilisent des algorithmes hybrides pour résoudre les problèmes abordés, afin d'obtenir des vitesses de convergence plus importantes et de meilleures solutions. Les approches évolutionnistes comme les algorithmes IBEA (Zitzler et Künzli, 2004), NSGA-II (Deb et al., 2002) ou encore SPEA2 (Zitzler et al., 2001) fournissent de bon résultats car elles permettent de gérer toutes les solutions représentant un compromis entre les différents objectifs en une seule exécution. Cependant, ces algorithmes sont plus lents et nécessitent souvent d'être couplés à des méthodes de recherche locale afin d'accélérer leur vitesse de convergence.

4.1.2 Choix des objectifs pour le DARP-PV-AN et modifications du modèle linéaire

Dans le cadre du problème de transport à la demande avec véhicules privés et sommets alternatifs, les résultats obtenus dans les chapitres précédents montrent l'utilité des véhicules privés pour obtenir des solutions avec une distance parcourue inférieure et une meilleure qualité de services pour les clients. En revanche, l'utilisation de véhicules supplémentaires engendre des coûts financiers additionnels non négligeables et ces derniers doivent être pris en compte lors de l'optimisation. Comme expliqué dans l'article de (Madsen et al., 1995),

l'utilisation de véhicules différents en plus de la flotte initiale, nécessite l'intégration de coûts spécifiques dans le calcul de la fonction objectif. Ainsi, le DARP-VP-AN étudié dans ce chapitre est défini avec deux objectifs à minimiser : la distance parcourue par l'ensemble des véhicules et le coût financier de l'ensemble des tournées. La distance parcourue correspond à l'objectif défini dans les chapitres précédents. Le coût financier est divisé pour chaque véhicule en deux parties : la première dépend de la distance parcourue par le véhicule, et la seconde correspond aux nombres de clients servis par le véhicule. Le modèle linéaire proposé dans la section 2.2 du chapitre 2 comporte les ajouts de nouvelles données, variables, contraintes et objectifs présentés ci-dessous :

a) Données

$fixe_k$ coût fixe par client transporté par le véhicule k si ce dernier est utilisé, $k \in K^T$

v_k coût par unité de distance parcourue par le véhicule k, $k \in K^T$

b) Variables

$dist_k$ distance parcourue par le véhicule k, $k \in K^T$

u_k nombre de clients transportés par le véhicule k, $k \in K^T$

c) Contraintes

La contrainte (35) somme la distance parcourue par le véhicule k dans la variable $dist_k$:

$$dist_k = \sum_{i \in N^T} \sum_{j \in N^T} c_{i,j}^k * x_{i,j}^k, \forall k \in K^T \quad (35)$$

La contrainte (36) utilise la technique du *big M* pour forcer u_k à la valeur 1 si le véhicule k est utilisé ($dist_k > 0$) :

$$u_k = \sum_{i \in N^T} \sum_{j \in N^T} x_{i,j}^k - 1, \forall k \in K^T \quad (36)$$

d) Objectifs

Le premier objectif correspond à la minimisation de la somme des distances parcourues par les véhicules :

$$\min \sum_{k \in K^T} dist_k \quad (37)$$

Le second objectif minimise les coûts financiers des tournées en additionnant les coûts fixes des clients aux coûts dépendant des trajets parcourus par chaque véhicule :

$$\min \sum_{k \in K^T} v_k * dist_k + u_k * fixe_k \quad (38)$$

L'algorithme développé dans ce chapitre vise à proposer un système d'aide à la décision fournissant un ensemble de plans de transport efficaces dans les deux objectifs proposés tout en prenant en compte la qualité de service des clients. La sélection d'un plan de transport dans l'ensemble proposé par l'algorithme est laissée à la discrétion du décideur.

4.2 Optimisation bi-objectif

Dans cette section, des précisions sur l'optimisation dans un contexte multi-objectif ainsi que certaines méthodes de résolutions sont apportées. Ces dernières sont issues du livre de (Ehrgott, 2005) ainsi que l'article de (Ehrgott et Gandibleux, 2000). Elles visent à expliquer la

formulation d'un problème bi-objectif et les notions d'espaces de décision et de solution. La définition d'optimalité dans un contexte multi-objectif et également abordée avec des rappels sur la notion de front de Pareto. Enfin, différentes méthodes de scalarisation utilisées dans ce chapitre sont expliquées.

4.2.1 Formulation d'un problème bi-objectif

Un problème multi-objectif est formalisé comme suit :

$$\min z(x)$$
$$\text{t. q. } g_j(x) \leq 0$$
$$\text{avec } x_i \in \mathbb{R}^n$$

- $x_i \in \mathbb{R}^n$ est composé de n variables avec $i \in \{1, \dots, n\}$;
- $g_j \in \mathbb{R}^n \to \mathbb{R}^m$ est composé de m contraintes avec $j \in \{1, \dots, m\}$;
- $z \in \mathbb{R}^n \to \mathbb{R}^p$ est composé de p fonctions objectif avec $z = (z_1, \dots, z_p)$ où les $z_i \in \mathbb{R}^n \to \mathbb{R}$.

Dans le cadre d'un problème bi-objectif, $z_k \in \mathbb{R}^n \to \mathbb{R}^2$ est composé de deux fonctions avec $k \in \{1,2\}$.

4.2.2 Espaces de décision et des objectifs

Dans le cadre d'un problème d'optimisation multi-objectif, il est important de distinguer l'espace de décision $\mathbb{R}^n$ (domaine des variables de décision) de l'espace des objectifs $\mathbb{R}^p$ (domaine des images des solutions par la fonction z).

a) Espace de décision

Dans le cadre d'un problème à n variables, l'espace de décision est défini sur $\mathbb{R}^n$. L'ensemble réalisable X dans $\mathbb{R}^n$ représente l'ensemble des solutions respectant les contraintes du problème et est défini par :

$$X = \{x \in \mathbb{R}^n : g(x) \leq 0\}$$

Une solution x est réalisable si $x \in X$.

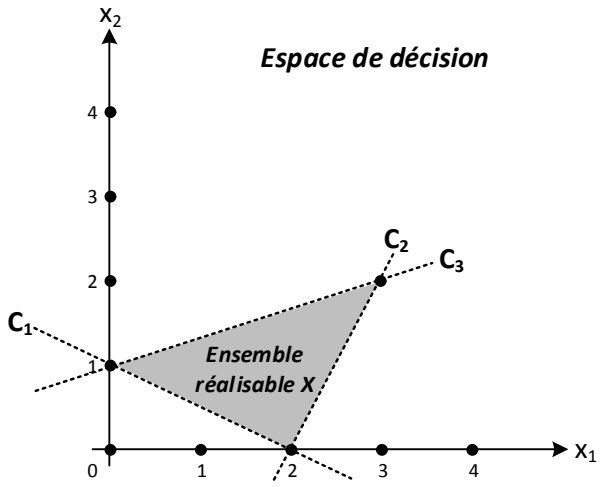


Figure 4.1 : Espace de décision à deux variables et ensemble réalisable respectant les contraintes C_1, C_2 et C_3

Dans l'exemple de la *Figure 4.1*, une illustration de l'espace de décision d'un problème à deux variables ainsi que l'ensemble réalisable X respectant les contraintes $-\frac{1}{2}x_1 - x_2 + 1 \leq 0\ (C_1), 2x_1 - x_2 - 4 \leq 0\ (C_2)$ et $-\frac{1}{3}x_1 + x_2 - 1 \leq 0\ (C_3)$ est proposée. Le point $(1; 1)$ est une solution réalisable du problème tandis que le point $(1; 2)$ est une solution non réalisable.

b) Espace des objectifs

Contrairement à l'optimisation mono-objectif, l'utilisation de plusieurs critères définit un espace des objectifs à plusieurs dimensions. Pour un problème composé de p critères, l'espace des objectifs est défini sur $\mathbb{R}^p$. L'ensemble des résultats Y dans $\mathbb{R}^p$ représente les images des solutions réalisables par z et est défini par :

$$Y = \{z(x) : x \in X\}$$

L'exemple de la *Figure 4.2* montre l'espace des objectifs ainsi que l'ensemble Y dans le cadre de la minimisation des fonctions $z_1 = 2x_1 + x_2$ et $z_2 = x_1 + 3x_2$ sur l'ensemble réalisable X défini précédemment sur la *Figure 4.1*.

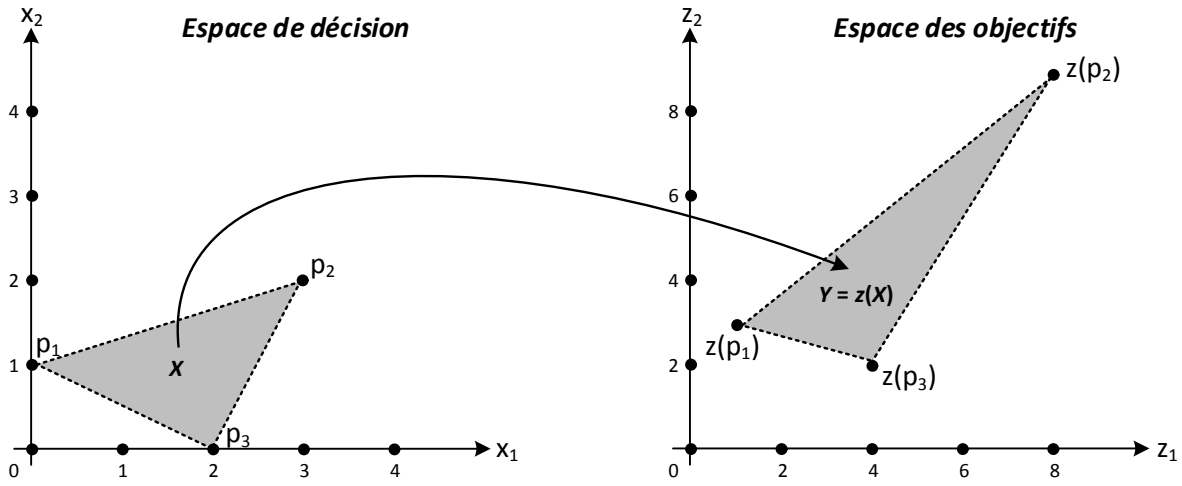


Figure 4.2 : Espace des objectifs avec deux critères pour l'espace de décision de la Figure 4.1

4.2.3 Définition de l'optimalité

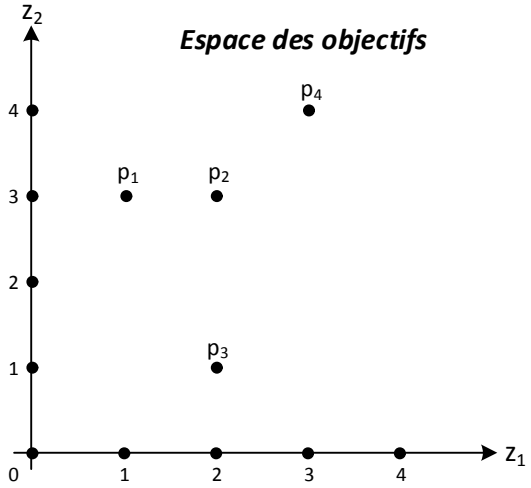
Dans le cadre mono-objectif, les solutions d'un problème peuvent être comparées facilement puisque la valeur associée à chaque solution est un nombre réel. Ainsi, dans le cas d'une minimisation, la solution respectant l'ensemble des contraintes avec la plus petite valeur est la solution optimale. Cependant, cette comparaison n'est plus possible dans le cadre multi-objectif. La notion de dominance a été introduite par Pareto dans les années 1900 et permet de comparer des solutions et de déterminer si l'une est meilleure que l'autre lorsque plusieurs objectifs sont présents :

Soit deux points y^1 et $y^2 \in \mathbb{R}^p$,

- y^1 domine y^2 au sens faible ($y^1 \leqq y^2$) si $\forall k \in \{1, \dots, p\}, y_k^1 \leq y_k^2$;
- y^1 domine y^2 au sens fort ($y^1 < y^2$) si $\forall k \in \{1, \dots, p\}, y_k^1 < y_k^2$;
- y^1 domine y^2 ($y^1 \leq y^2$) si $y^1 \neq y^2$ et $y^1 \leqq y^2$;

- y^1 et y^2 sont incomparables si aucune des trois relations de dominance précédentes ne sont vérifiées.

Dans l'exemple de la *Figure 4.3* les points $p_1 = (1;3), p_2 = (2;3), p_3 = (2;1)$ et $p_4 = (3;4)$ définis sur l'espace des objectifs $\mathbb{R}^p$ sont comparés. On constate que p_1 et p_3 dominent au sens fort p_4 et au sens faible p_2, p_2 domine au sens fort p_4 et p_1 et p_3 sont incomparables. On peut donc dire que p_1 et p_3 dominent p_2 et p_4.



$p_1 \prec p_4$
$p_1 \lessapprox p_2$
$p_2 \prec p_4$
$p_3 \prec p_4$
$p_3 \lessapprox p_2$
p_1 et p_3 sont incomparables

Figure 4.3 : Comparaison des relations de dominance entre différents points

Ces définitions sont étendues dans l'espace de décision par les notations suivantes :

Soit deux solutions x^1 et $x^2 \in \mathbb{R}^n$,

- x^1 domine x^2 au sens faible ($x^1 \lessapprox x^2$) si $z(x^1) \lessapprox z(x^2)$;
- x^1 domine x^2 au sens fort ($x^1 \prec x^2$) si $z(x^1) \prec z(x^2)$;
- x^1 domine x^2 ($x^1 \precsim x^2$) si $x^1 \neq x^2$ et $z(x^1) \lessapprox z(x^2)$;
- x^1 et x^2 sont incomparables si aucune des trois relations de dominance précédentes ne sont vérifiées.

L'ensemble des points non-dominés de l'espace des objectifs forment un front de Pareto, défini dans la section suivante.

4.2.4 Front de Pareto

Comme précisé dans la section 4.2.3 et illustré sur la *Figure 4.4*, l'ensemble des solutions non-dominées forment un front de Pareto. Il est ensuite possible de distinguer deux types de solutions sur ce dernier :

- Les solutions supportées dont l'image est située sur la partie convexe de Y ;
- Les solutions non-supportées dont l'image n'est pas située sur la partie convexe de Y, mais ne sont pas non plus des solutions dominées.

Les solutions supportées peuvent être divisées en deux ensembles :

- Les solutions supportées extrêmes dont les images correspondent à un point extrême de l'enveloppe convexe dans l'espace des objectifs ;
- Les solutions supportées non-extrêmes représentant l'ensemble des points restant.

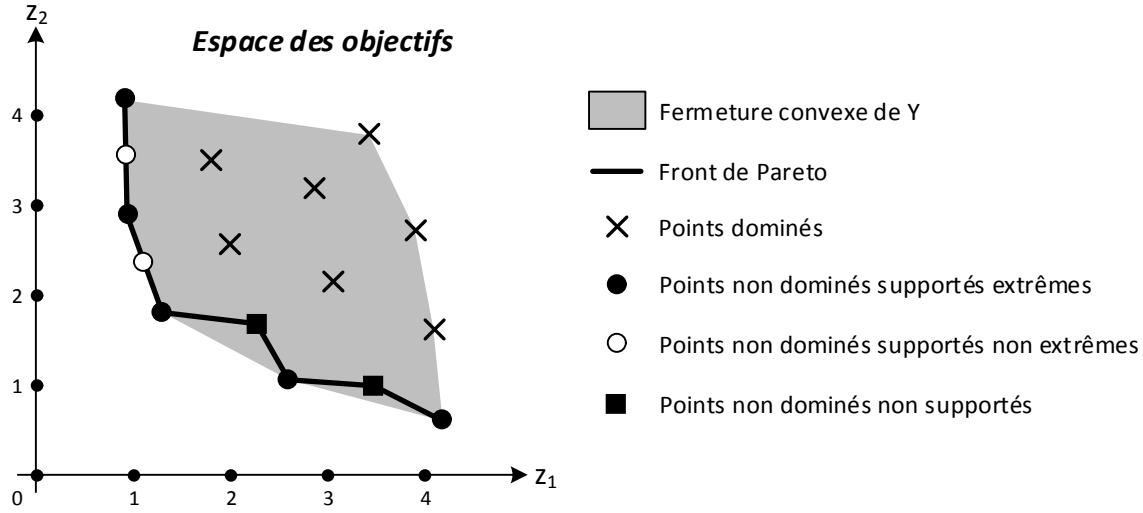


Figure 4.4 : Représentation des différents points non-dominées formant un front de Pareto (Cerqueus, 2015)

4.2.5 Méthodes de scalarisation

Le principe des méthodes de scalarisation est de convertir un problème multi-objectif en mono-objectif paramétré, puis de répéter sa résolution avec des paramètres différents. Dans l'idéal, chaque solution optimale au sens de Pareto dans le contexte multi-objectif doit être atteignable avec un ensemble de paramètres fixé lors de la scalarisation. Deux méthodes sont présentées dans les sections suivantes et utilisées plus loin dans le chapitre : la méthode dichotomique par somme pondérée et la méthode ϵ-contrainte.

a) Méthode dichotomique par somme pondérée

Introduite par (Geoffrion, 1968), la méthode de la somme pondérée consiste à agréger de façon linéaire l'ensemble des critères d'optimisation z_k au sein d'un même objectif tout en affectant à chacun un poids λ_k :

$$\min\left\{\sum_{k=1}^{p} \lambda_k z_k(x) : x \in X\right\}$$

Cette méthode a été utilisée par (Aneja et Nair, 1979), dans une approche dichotomique sur un problème de transport bi-objectif pour générer l'ensemble des solutions non-dominées supportées extrêmes (aucune garantie n'est attribuée à la génération de l'ensemble des points non-extrêmes). Elle ne génère donc que des solutions situées sur l'enveloppe convexe de l'espace des solutions. L'algorithme proposé dans leur article est rappelé dans l'*Algorithme 27* et une illustration de la méthode est proposée sur la *Figure 4.5*.

L'algorithme commence par chercher les deux solutions optimales sur le premier et le second objectif (lignes 3-4) avec respectivement les vecteurs de poids $\lambda^1 = (1,0)$ et $\lambda^2 = (0,1)$. La variable ϵ utilisée ici est une petite valeur strictement positive permettant d'éviter de générer une solution dominée. Si les deux solutions générées sont identiques, alors le problème P contient une unique solution non-dominée et la méthode s'arrête en renvoyant cette solution (lignes 6-7). En revanche, si les deux solutions sont différentes, la méthode va chercher successivement les différentes solutions non-dominées supportées extrêmes (lignes 13-25). Pour chaque paire de points (y^l, y^r), un vecteur de poids λ^i est calculé (ligne 16) donnant la

$i^{ème}$direction de recherche correspondant à la perpendiculaire de la droite (y^l, y^r). Ce vecteur permet d'obtenir lors de la résolution de P la solution x^{new} : si cette dernière est située sous la droite (y^l, y^r), alors deux nouveaux axes de recherche doivent être considérés, sinon la recherche ne génère pas de nouveaux axes (lignes 19-23).

Algorithme 27 ***Dichotomic weighted sum*** **(Aneja et Nair, 1979)**

Input parameter

P : A bi-objective combinatorial optimization problem

Output variable

res : Array of non-dominated supported (extremes) solutions

Local variables

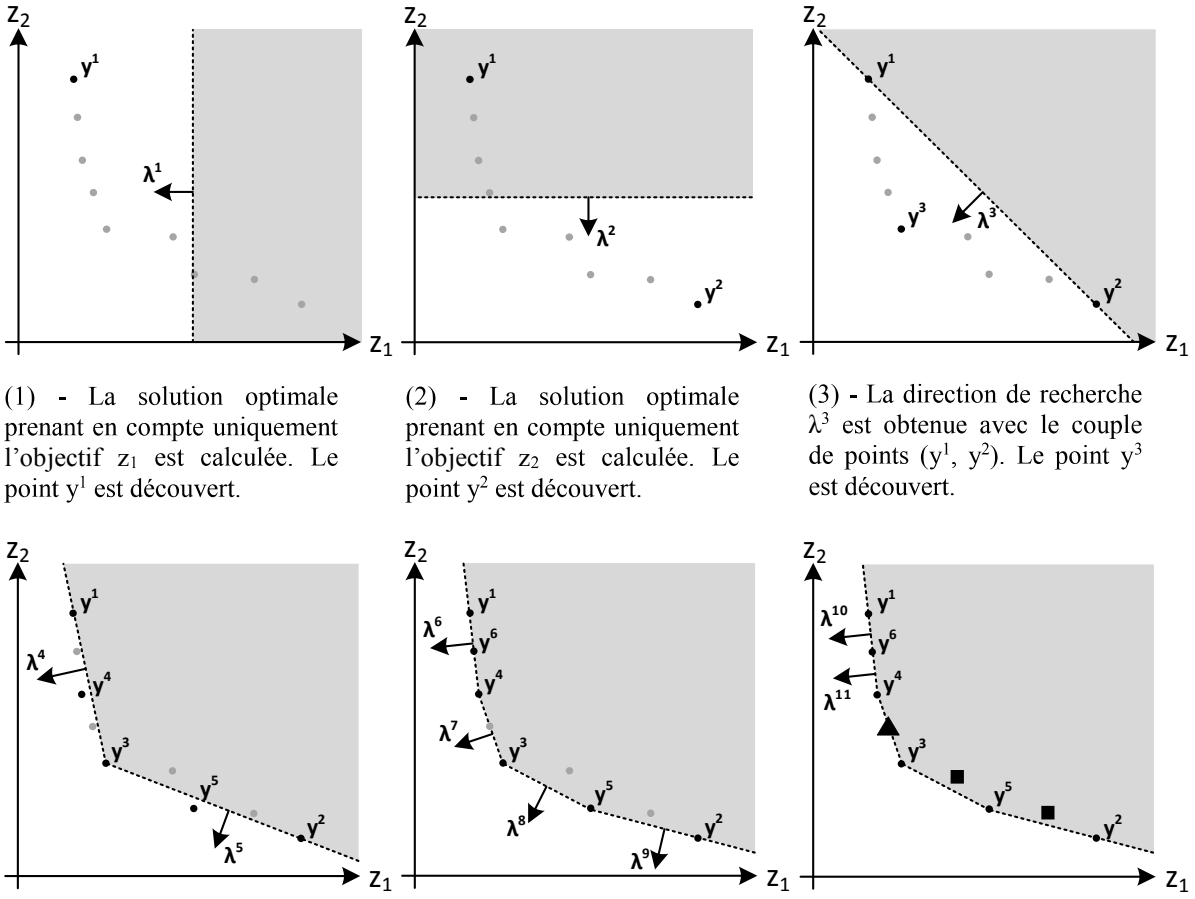
ϵ : A small value strictly greater than 0
λ : Weight vector used to aggregate objectives
$stack$: Stack used to store pair of points

Available functions

z(x) : Returns a point $y \in Y$ in the objective space of the solution $x \in X$
optimal(P, λ) : Returns an optimal solution of P whose objectives are aggregated with weight vector λ

Begin

1 res := ∅
2 $stack$:= ∅
3 x^1 := optimal(P,$(1, \epsilon)$)
4 x^2 := optimal(P,$(\epsilon, 1)$)
5
6 **if** z(x^1) = z(x^2) **then**
7 | res.add(x^1)
8 **else**
9 | res.add(x^1)
10 | res.add(x^2)
11 | $stack$.push((z(x^1), z(x^2)))
12 |
13 | **while** $stack \neq \emptyset$ **do**
14 | |
15 | | y^l, y^r := $stack$.pop()
16 | | λ := $(y_2^l - y_2^r, y_1^r - y_1^l)$
17 | | x^{new} := optimal(P,λ)
18 | |
19 | | **if** $\lambda x^{new} < \lambda y^l$ **then**
20 | | | res.add(x^{new})
21 | | | $stack$.push((y^l, z(x^{new})))
22 | | | $stack$.push((z(x^{new}), y^r))
23 | | **end if**
24 | |
25 | **end while**
26 |
27 **end if**
28
29 **return** { res }
End



(1) - La solution optimale prenant en compte uniquement l'objectif z_1 est calculée. Le point y^1 est découvert.

(2) - La solution optimale prenant en compte uniquement l'objectif z_2 est calculée. Le point y^2 est découvert.

(3) - La direction de recherche λ^3 est obtenue avec le couple de points (y^1, y^2). Le point y^3 est découvert.

(4) - Les directions λ^4 et λ^5 sont obtenues avec les couples (y^1, y^3) et (y^3, y^2). Les points y^4 et y^5 sont découverts.

(5) - Les directions λ^6, λ^7, λ^8 et λ^9 sont obtenues avec les couples (y^1, y^4), (y^4, y^3), (y^3, y^5) et (y^5, y^2). Seul le point supporté non-extrême y^6 est découvert.

(6) - Les directions λ^{10} et λ^{11} sont obtenues avec les couples (y^1, y^6), (y^6, y^4). Aucun nouveau point n'est découvert. Les points non-supportés (■) et le point non-extrême (▲) ne sont pas obtenus.

Figure 4.5 : Illustration de la méthode dichotomique par somme pondérée définie par (Geoffrion, 1968)

b) Méthode ϵ-contrainte

Une autre approche de scalarisation est la méthode ϵ-contrainte. Dans cette méthode introduite par (Haimes et al., 1971), une seule fonction objectif est minimisée : les autres fonctions objectif sont transformées en contraintes. Le nouveau problème d'optimisation s'écrit :

$$\begin{aligned} &\min z_l(x) \\ &\text{t.q. } z_k(x) \leq \varepsilon_k, \forall k \in \{1, \dots, p\} \backslash \{l\} \\ &\varepsilon_k \geq 0, \forall k \in \{1, \dots, p\} \backslash \{l\} \\ &x \in X \end{aligned}$$

Cette méthode permet de générer l'ensemble des solutions non-dominées en résolvant itérativement le problème d'optimisation ci-dessus avec différentes valeurs pour ε. Pour cela, à chaque itération de l'algorithme, le vecteur ε est modifié afin de rendre impossible la découverte des solutions trouvées aux itérations précédentes. Dans un contexte bi-objectif

où z_1 est le critère sélectionné, si à l'itération i le point y^i est l'optimum, alors à l'itération $i+1$ le problème à résoudre est le suivant :

$$\min z_1(x)$$
$$t.q.\ z_2(x) \leq y_2^i - \epsilon$$
$$x \in X$$

La valeur ϵ est utilisée lorsque la fonction objectif permet de fournir des résultats non entiers : elle représente l'écart minimal entre deux solutions sur l'objectif z_2. Plus ce dernier est petit et plus le front de Pareto généré comportera de solutions et favorisera ainsi une représentation précise de l'ensemble des points non-dominés, en revanche le nombre d'itérations de l'algorithme sera plus important. Si la valeur ϵ est trop grande, alors l'algorithme sera plus rapide mais le front généré ne sera pas nécessairement représentatif de la totalité des points non-dominés. Dans le cas où la fonction objectif est constituée de coefficients et de variables entières, alors la contrainte peut être modifiée comme suit : $z_2(x) \leq y_2^i - 1$ (ϵ est fixé à 1).

L'exemple de la *Figure 4.6* illustre la méthode ϵ-contrainte implémentée par l'*Algorithme 28* dans le contexte bi-objectif qui est utilisé dans ce chapitre. La zone colorée à chaque étape du schéma représente les solutions non réalisables à l'itération courante, tandis que la flèche représente l'objectif optimisé.

Algorithme 28 $\epsilon - constraint$

Input parameter

P : A bi-objective combinatorial optimization problem

Output variable

res : Array of non-dominated solutions

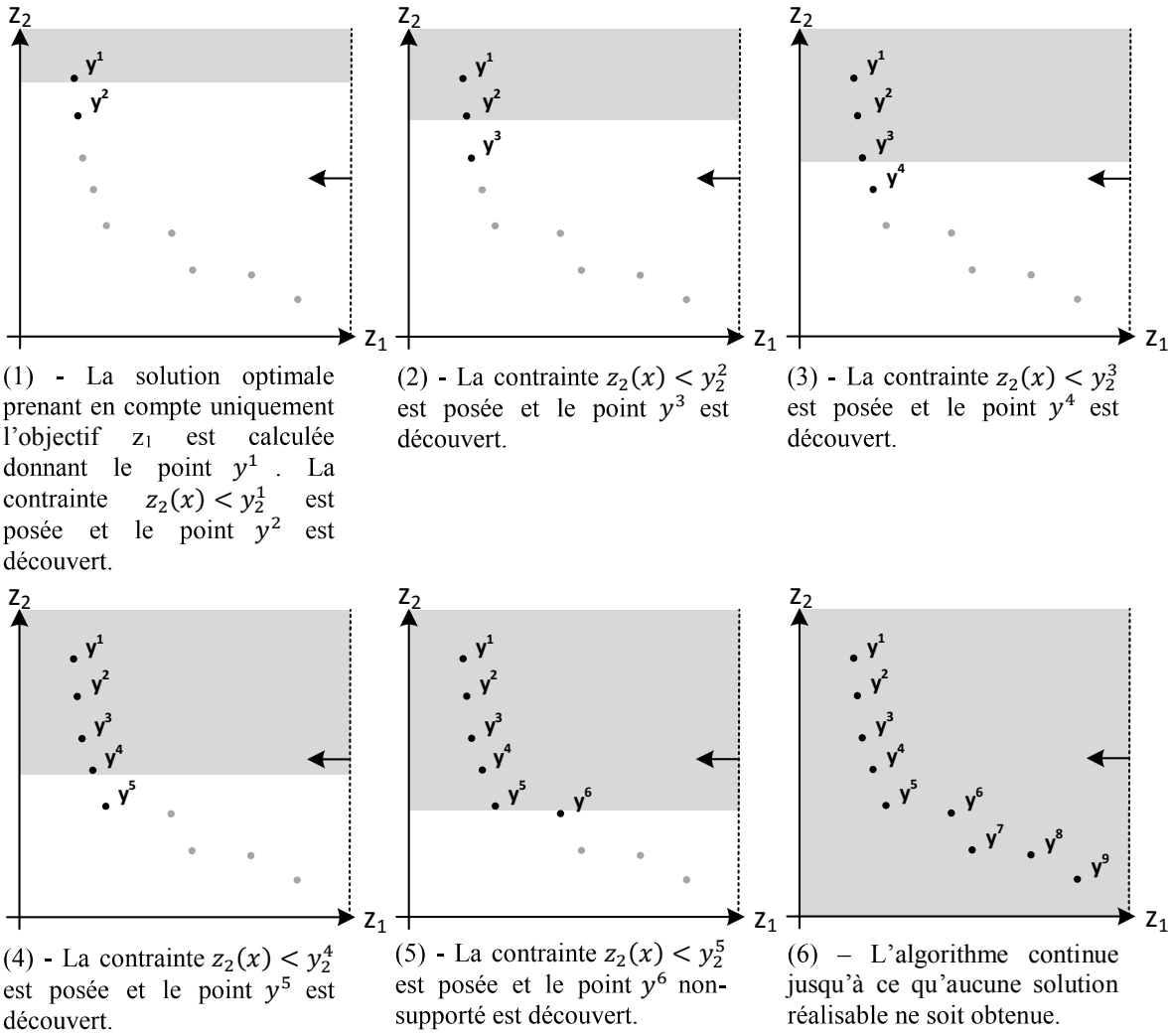
Available functions

$z_2(x)$	:	Returns the value of solution x in the second objective
optimalZ1(P)	:	Returns an optimal solution of P optimizing first objective
addConstraint(P, c)	:	Add the constraint $z_2(x) < c$ in P

Begin

1 $res := \emptyset$
2 x^{new} := optimalZ1(P)
3
4 **while** $x^{new} \neq \emptyset$ **do**
5 |
6 | res.add(x^{new})
7 | addConstraint(P,$z_2(x^{new})$)
8 | x^{new} := optimalZ1(P)
9 |
10 **end while**
11
12 **return** { res }

End



(1) - La solution optimale prenant en compte uniquement l'objectif z_1 est calculée donnant le point y^1 . La contrainte $z_2(x) < y_2^1$ est posée et le point y^2 est découvert.

(2) - La contrainte $z_2(x) < y_2^2$ est posée et le point y^3 est découvert.

(3) - La contrainte $z_2(x) < y_2^3$ est posée et le point y^4 est découvert.

(4) - La contrainte $z_2(x) < y_2^4$ est posée et le point y^5 est découvert.

(5) - La contrainte $z_2(x) < y_2^5$ est posée et le point y^6 non-supporté est découvert.

(6) – L'algorithme continue jusqu'à ce qu'aucune solution réalisable ne soit obtenue.

Figure 4.6 : Illustration de la méthode ϵ-contrainte proposée par (Haimes et al., 1971)

4.3 Métaheuristique hybride proposée

L'algorithme proposé pour résoudre le DARP-PV-AN multi-objectif est une évolution de la métaheuristique hybride de type *cluster-first route-second* proposée dans le chapitre précédent. L'algorithme est composé de 3 étapes illustrées sur la *Figure 4.9* :

- La première étape est effectuée par l'algorithme génétique bi-objectif NSGA-II : il détermine les véhicules utilisés et les clients qui y sont affectés. Les principes fondamentaux de l'algorithme ainsi que son implémentation pour le DARP-PV-AN bi-objectif sont proposés dans la section 4.4 ;
- La deuxième étape correspond à l'évaluation des individus créés par le NSGA-II avec un algorithme de *Branch and Bound.* Ce dernier calcule l'ordre de visite optimal de chaque véhicule en respectant l'ensemble des contraintes du DARP-PV-AN bi-objectif. Comme dans le chapitre précédent, les tournées sont sauvegardées pour éviter de recalculer des tournées rencontrées précédemment. La méthode est similaire à celle proposée dans le chapitre précédent et les modifications dans le cadre du DARP-PV-AN bi-objectif sont décrites dans la section 4.4.2 ;
- La dernière étape vise à enrichir la population du NSGA-II en résolvant le problème

de couverture par ensembles avec poids (WSCP) par programmation linéaire en nombre entier (PLNE) comme dans le chapitre 3. Cependant, la résolution du WSCP est effectuée dans un cadre bi-objectif : différentes méthodes de résolution sont mises en place et comparées. La sélection de la méthode de résolution du WSCP au lancement de la méthode hybride est un paramètre sélectionné par l'utilisateur. Le but de la résolution du WSCP bi-objectif est de fournir un ensemble d'individus représentant des solutions, et de les insérer dans la population pour la génération suivante du NSGA-II. Les différentes méthodes de résolution du WSCP bi-objectif sont décrites dans la section 4.5.

Donc, le but de la méthode hybride est tout d'abord de déterminer pour chaque véhicule, les clients qui leur sont affectés grâce au NSGA-II. Pour cela, un espace de codage indirect des solutions est utilisé comme illustré sur la *Figure 4.7*. Cet espace de codage se compose d'un tableau de véhicules ou la position i représente l'identifiant du client et la valeur de la case i, le véhicule qui lui est associé. Une fois l'ensemble des clients affectés aux véhicules, la méthode d'évaluation de *Branch and Bound* détermine, pour chaque véhicule, les tournées optimales respectant les contraintes du DARP-PV-AN. Ces dernières sont sauvegardées dans une structure de données de type *HashMap* pour éviter de les calculer si elles apparaissent dans de futurs individus.

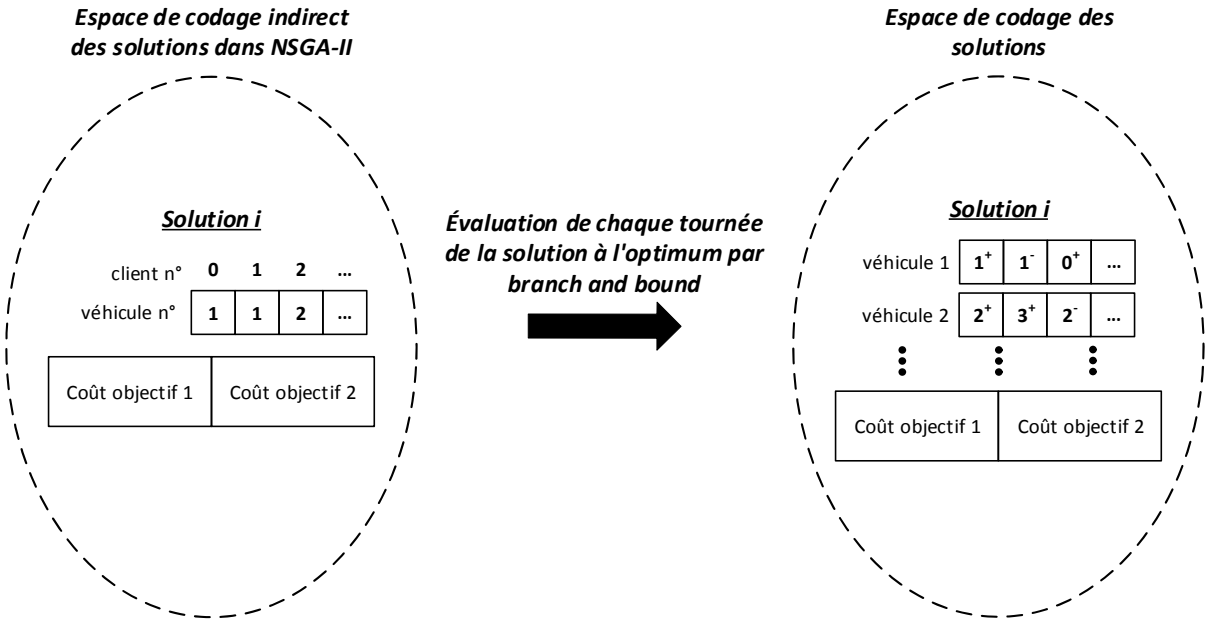


Figure 4.7 : Espace de codage indirect utilisé par le NSGA-II et décodage par la fonction d'évaluation

Ensuite, une méthode « d'assemblage » des tournées sauvegardées est appliquée à la fin de chaque génération du NSGA-II pour former de nouveaux individus représentant des solutions. Cette méthode est basée sur la résolution du problème WSCP bi-objectif par PLNE. Le contexte étant multi-objectif, des méthodes de scalarisation sont utilisées. La *Figure 4.8* illustre l'évolution de la population du NSGA-II lorsque cette méthode est ajoutée :

- Au début d'une nouvelle génération du NSGA-II, un ensemble d'individus P_i de taille n et un ensemble T de tournées optimales déjà calculées dans les générations précédentes sont fournies ;
- Un ensemble d'individus $P_i{}'$ de taille n est obtenu par l'application des méthodes de sélection/croisement/mutation/évaluation du NSGA-II sur la population P_i . Les

nouvelles tournées découvertes dans cette étape sont insérées dans T ;

- Un ensemble d'individus représentant des solutions P_i" de taille non fixée est obtenu avec la résolution du WSCP bi-objectif par PLNE via l'utilisation de méthodes de scalarisation ;
- Les ensembles P_i, $P_i{}'$ et P_i" sont fusionnés et une méthode de filtrage est appliquée par le NSGA-II pour garder les n meilleurs individus, formant ainsi la nouvelle population P_{i+1} pour la génération suivante.

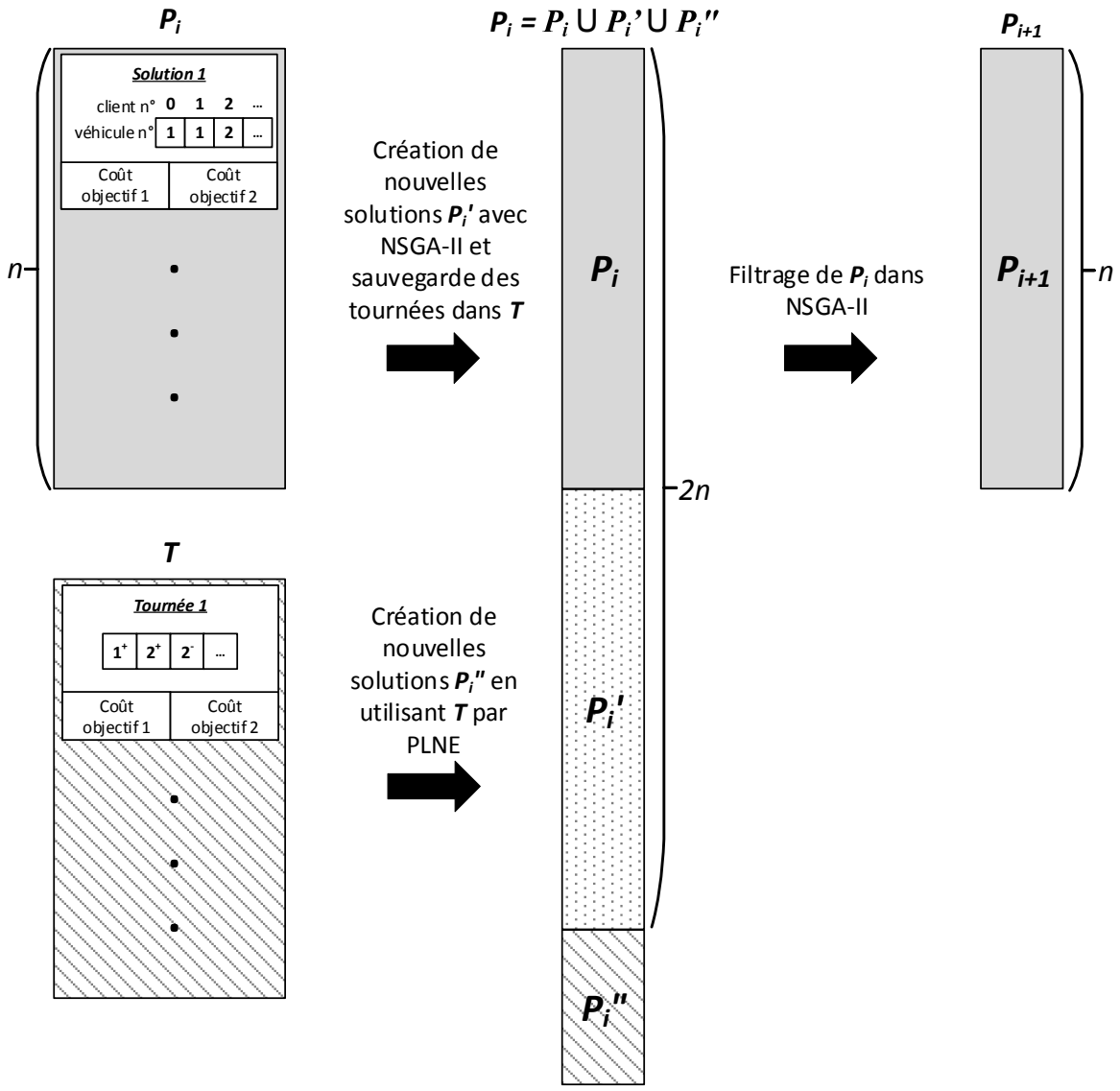


Figure 4.8 : Évolution de la population au cours d'une génération du NSGA-II hybride

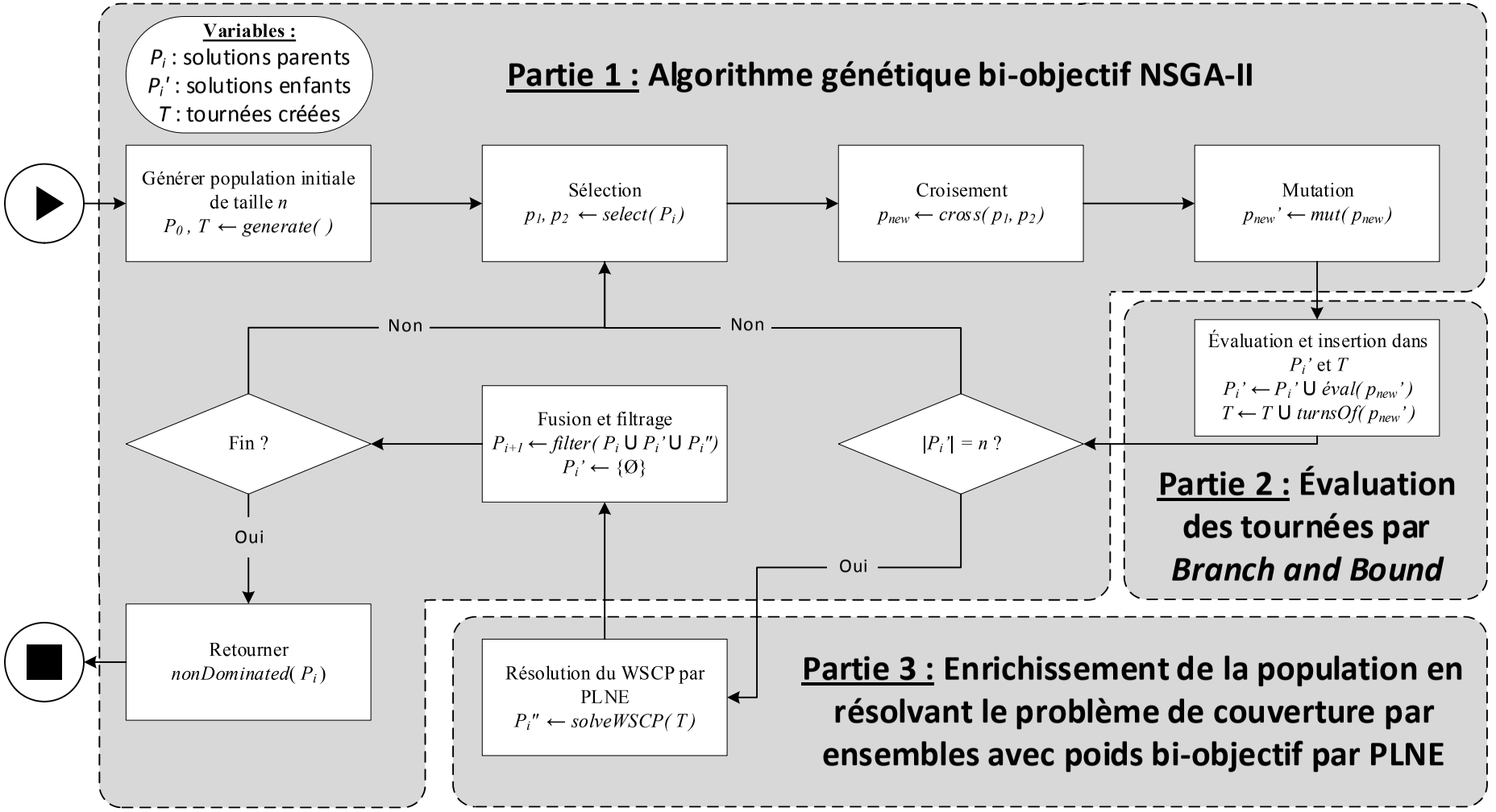


Figure 4.9 : Schéma global de la méthode de résolution proposée

4.4 Algorithme génétique bi-objectif NSGA-II

Cette section est divisée en deux parties : la première explique le fonctionnement global de l'algorithme NSGA-II proposé par (Deb et al., 2002) et la seconde montre son implémentation dans le cadre de la résolution du DARP-PV-AN.

4.4.1 Principes fondamentaux de l'algorithme

L'algorithme NSGA-II est un algorithme évolutionniste : c'est une méthode de choix pour échantillonner le front de Pareto d'un problème multi-objectif. En effet, travailler avec une population d'individus permet d'approximer l'ensemble Pareto-optimal tout au long de l'exécution de l'algorithme. Dans un cadre multi-objectif, ces algorithmes sont appelés MOGA (*Multi-Objective Genetic Algorithm*) ou MOEA (*Multi-Objective Evolutionary Algorithm*).

Comme tout algorithme évolutionniste, le NSGA-II est composé d'une phase de génération d'individus initiale, suivie par une répétition des phases : sélection de parents, croisement, mutation, évaluation et filtrage de la population. Contrairement à un algorithme génétique mono-objectif, le NSGA-II ne renvoie pas une seule solution mais un ensemble de solutions non-dominées.

Dans tout MOGA, la présence d'une méthode permettant de comparer les individus entre eux est obligatoire. En effet, lorsque l'on compare les parents lors de la phase de sélection ou encore que l'on filtre la population avant la génération suivante, il est nécessaire de pouvoir déterminer quels individus représentent des solutions non dominées. Dans un cadre mono-objectif, cette comparaison est très souvent effectuée sur l'objectif optimisé : un individu avec une meilleure valeur qu'un autre est favorisé. Dans l'algorithme NSGA-II, la comparaison est basée sur deux mesures affectées à chaque individu : son rang déterminé par le tri par non-domination, et sa marge représentant l'isolement de la solution qu'il représente dans l'espace des objectifs.

a) Tri par non-domination et affectation du rang

Le tri par non-domination (*non-dominated sorting*) permet de partitionner la population d'individus en des fronts successifs, chaque front étant composé d'individus représentant des solutions non-dominées par les fronts suivants. Le principe consiste à trouver dans la population l'ensemble des individus non-dominés : ceux-ci appartiennent au front de rang 1 et sont temporairement retirés de la population. Le nouvel ensemble d'individus non-dominés parmi les individus restants de la population va permettre de créer le front numéro 2. Le processus est itéré aussi longtemps que des individus ne sont pas affectés à un front, comme représenté sur la *Figure 4.10*. La méthode implémentée est détaillée dans l'*Algorithme 29* et a été proposée par (Deb et al., 2002) avec une complexité globale en $O(nc \times ns^2)$, nc représentant le nombre de critères à optimiser et ns la taille de l'ensemble à trier. L'ensemble des individus est représenté par un tableau Pop. Le vecteur $front$ contient des ensembles d'indices d'individus représentant leurs rangs : si $i \in front[k]$ alors l'individu Pop[i] est sur le front de rang k. La fonction dom(A, B) est une fonction booléenne renvoyant $true$ si A domine B. Le tableau $nbBetter$ contient à l'indice i le nombre d'individus dominant $Pop[i]$ et $setWorse$ contient à l'indice i les indices des individus dominés par $Pop[i]$.

Algorithme 29 $non\_dominated\_sort$ **(Deb et al., 2002)**

Input parameter

Pop : Array of individuals

Local variables

ns : Size of Pop
nf : Rank/Front number
$nbBetter$: Array of integer where $nbBetter[i]$ is the number of individuals dominating $Pop[i]$
$setWorse$: Array of sets of individuals where $setWorse[i]$ is the set of individuals dominated by $Pop[i]$
$front$: Array of sets of integers where $front[i]$ is the set of indices of individuals in Pop of rank i

Available functions

dom(A,B) : Return $true$ if individual A dominate individual B, $false$ otherwise
setRank(i): Set the rank i to an individual

Begin

```
1   front[0] := ∅
2
3   for i := 0 to ns − 1 do
4   |   nbBetter[i] := 0
5   |   setWorse[i] := ∅
6   |   for j := 0 to ns − 1 do
7   |   |   if dom(Pop[i],Pop[j]) then
8   |   |   |   setWorse[i].add(j)
9   |   |   else if dom(Pop[j],Pop[i])) then
10  |   |   |   nbBetter[i] := nbBetter[i] + 1
11  |   |   end if
12  |   end for
13  |   if nbBetter[i] = 0 then
14  |   |   front[0].add(i)
15  |   end if
16  end for
17
18  nf := 0
19
20  do
21  |   for all i in front[nf] do
22  |   |   Pop[i].setRank(nf)
23  |   |   for all j in setWorse[i] do
24  |   |   |       nbBetter[j] := nbBetter[j] − 1
25  |   |   |   if nbBetter[j] = 0 then
26  |   |   |   |   front[nf + 1].add(j)
27  |   |   |   end if
28  |   |   end for
29  |   end for
30  |   nf := nf + 1
31  while front[nf + 1] ≠ ∅
End
```

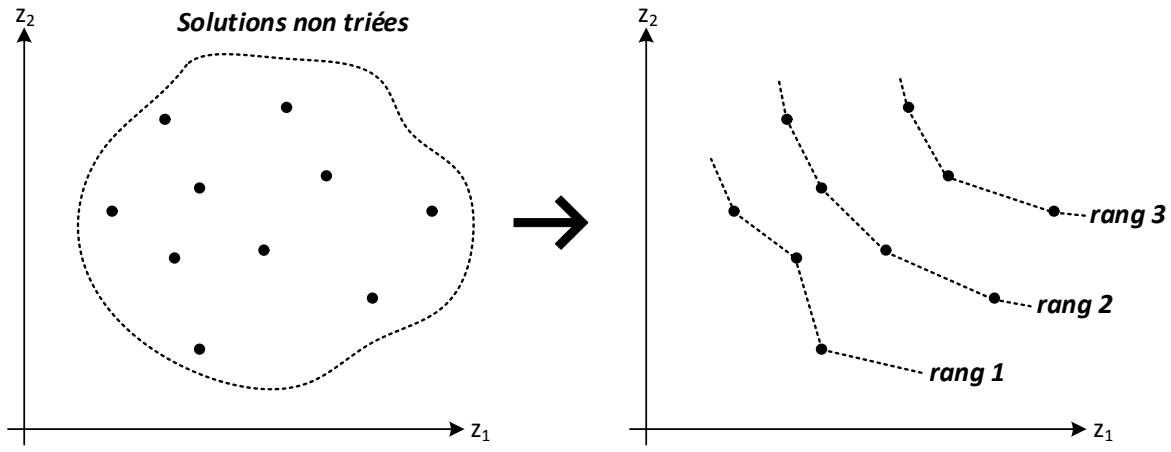


Figure 4.10 : Tri par non-domination

b) Calcul de la marge

Une deuxième valeur appelée marge est affectée à chaque individu. Cette dernière est une représentation de « l'isolement » de la solution correspondant à l'individu vis-à-vis des autres et se calcule au sein de chaque front obtenu lors du tri par non-domination. Dans le contexte du DARP-PV-AN bi-objectif, le calcul s'effectue comme suit :

Soit R un ensemble d'individus de front identique, trié par valeurs croissantes du 1er critère et R_k l'individu en position k dans l'ensemble trié. Les valeurs z_c^{min} et z_c^{max} représentent les valeurs minimales et maximales de l'objectif c dans l'ensemble R. Pour $k = 1$ ou $k = |R|$, alors $marge(R_k) = \infty$, dans le but d'élargir le front. Si $1 < k < |R|$:

$$marge(R_k) = \frac{z_1(R_{k+1}) - z_1(R_{k-1})}{z_1^{max} - z_1^{min}} + \frac{z_2(R_{k-1}) - z_2(R_{k+1})}{z_2^{max} - z_2^{min}}$$

Une illustration du calcul de la marge est proposée sur la *Figure 4.11*. Plus la marge est grande, plus le point est isolé.

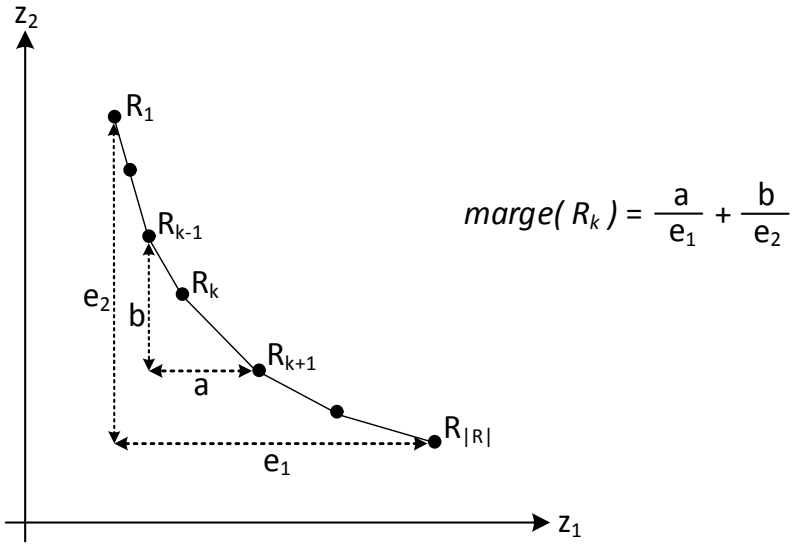


Figure 4.11 : Calcul de la marge associée à un individu représentant une solution (Deb et al., 2002)

c) Comparaison des individus

Dans l'algorithme NSGA-II, la comparaison des individus est effectuée grâce aux rangs et aux marges de ces derniers. Un individu A est jugé meilleur qu'un individu B s'il est de rang inférieur. Si A et B appartiennent au même front, on les départage en favorisant celui ayant la plus grande marge : ce choix permet de maintenir un front de Pareto bien dispersé en favorisant les individus isolés. L'idée principale est d'avantager les individus représentant les meilleures solutions (rang) tout en gardant un front de Pareto étendu (marge) empêchant ainsi la formation de groupes de solutions très similaires sur leurs objectifs.

4.4.2 Implémentation du NSGA-II

Comme dans le chapitre précédent, l'algorithme génétique (ici NSGA-II) est responsable de la partie *clustering*, c'est-à-dire de l'affectation des clients dans les véhicules. L'implémentation multi-objectif (*Algorithme 30*) étant proche de la version mono-objectif proposée dans le chapitre 3, seules les modifications apportées aux différentes parties de l'algorithme mono-objectif sont ici présentées :

- Les modifications concernent l'encodage d'un individu, la fonction de sélection par tournoi binaire ($bin\_tournament\_selection$), la fonction d'évaluation ($evaluation$), l'enrichissement de la population en créant de nouvelles solutions par résolution du WSCP bi-objectif par PLNE avec CPLEX ($launch\_CPLEX\_tour\_selection$) et la méthode de filtrage pour la génération suivante ($next\_iteration\_selection$) ;
- Enfin, une fonction calculant le rang et la marge de chaque individu est ajoutée, basée sur l'implémentation de (Lacomme et al., 2006).

a) Encodage d'un individu

L'encodage d'un individu pour le NSGA-II reprend les concepts développés dans le cadre mono-objectif du DARP-PV-AN : il représente l'affectation pour chaque client $c \in C$ du véhicule $k \in K^T$ par lequel il est transporté. Pour cela, un vecteur α est utilisé ou α_i représente le véhicule affecté au client $i \in C$. Dans le cadre mono-objectif, ce vecteur est lié à une valeur entière représentant le coût de la fonction objectif de la solution potentielle représentée par cet individu.

Lors de l'optimisation bi-objectif, trois nouvelles valeurs sont ajoutées à chaque individu formant ainsi une structure composée de :

- α un vecteur représentant l'affectation des clients dans les véhicules ;
- $dist$ un entier représentant la distance parcourue par les véhicules ;
- $cost$ un entier représentant le coût financier ;
- $rank$ le rang de l'individu ;
- $crowding$ la marge de l'individu.

b) Calcul du rang et de la marge

Le rang et la marge de chaque individu sont calculés avant l'appel à la fonction de filtrage de la population. Une mise à jour de la marge est effectuée après le filtrage. Ce calcul est répété car la valeur de la marge de chaque individu peut changer suite à l'appel de la fonction filtrant la population.

Algorithme 30 $NSGA-II$

Input parameters

t : Time limit of the algorithm
m : Mutation's probability
$sizePop$: Number of individuals in the population

Output variable

$Pareto$: Array of non-dominated individual

Local variables

$parents$: Array of individuals
$firstParent$: First parent for crossover
$secondParent$: Second parent for crossover
$child$: Child created by crossover
$children$: Array of current generation's individuals
$elapsedTime$: Time since algorithm's launch

Begin

1 **// Step 1. Generate initial population**
2 $parents$:= generate_initial_solutions($sizePop$)
3 compute_rank_crowding($parents$)
4 **while** $elapsedTime < t$ **do**
5 | **for** i := 0 **to** $sizePop - 1$ **do**
6 | |
7 | | **// Step 2. Parents selection before crossover**
8 | | $\{firstParent, secondParent\}$:= bin_tournament_selection($parents$)
9 | |
10 | | **// Step 3. Child creation by crossover**
11 | | $child$:= one_cut_crossover($firstParent, secondParent$)
12 | |
13 | | **// Step 4. Mutation on child**
14 | | $child$:= mutation($child, m$)
15 | |
16 | | **// Step 5. Keep private vehicles or not**
17 | | private_vehicle_keeping($child$)
18 | |
19 | | **// Step 6. Individual evaluation**
20 | | evaluation($child$)
21 | |
22 | | $children$.add($child$)
23 | **end for**
24 |
25 | **// Step 7. CPLEX tour selection (described section 4.5)**
26 | launch_cplex_tour_selection($children$)
27 |
28 | **// Step 8. Compute rank and crowding**
29 | compute_rank_crowding($parents \cup children$)
30 |
31 | **// Step 9. Selection for next iteration**
32 | next_iteration_selection($parents, children$)
33 |
34 | **// Step 10. Update crowding**
35 | update_crowding($parents$)
36 **end while**
37 $Pareto$ **:=** get_first_rank($parents$)
38 **return** { $Pareto$ }

End

La *Figure 4.12* illustre la nécessité de recalculer la marge après le filtrage. Dans une population de 5 individus maximum, un individu doit être retiré dans l'ensemble $P = \{A, B, C, D, E, F\}$ suite à la génération de nouveaux individus. Pour cela, le rang et la marge des individus sont calculés. Comme les individus sont tous de même rang, l'individu supprimé sera celui ayant la plus petite valeur de marge (C ou D) : soit l'individu D dans l'exemple. La marge est ensuite calculée de nouveau car la suppression de l'individu D engendre des individus C et E meilleurs que l'individu B. Si la marge n'avait pas été recalculée, l'individu B aurait été considéré comme meilleur que l'individu C et équivalent à l'individu E, ce qui est faux dans l'ensemble $P \backslash D$. Ces résultats faux de la méthode de comparaison entraineraient de mauvaises sélections dans la méthode du tournoi binaire présentée ci-après, ce qui prouve la nécessité de recalculer la marge après filtrage de la population.

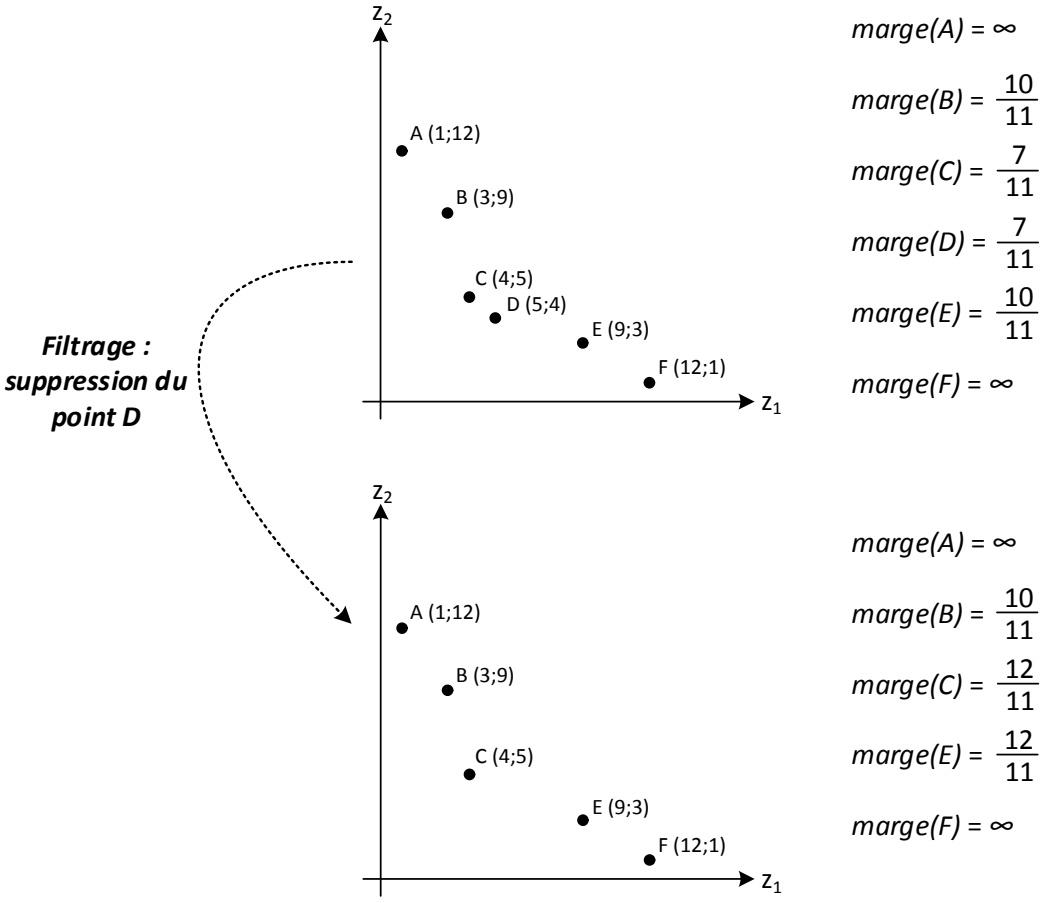


Figure 4.12 : Nécessité de recalculer la marge après la procédure de filtrage : la suppression du point D (moins bonne marge parmi les individus présentés) entraîne un point C meilleur que le point B

c) Sélection des couples

La méthode de sélection des individus pour le croisement utilise l'algorithme de tournoi binaire : deux individus sont choisis aléatoirement parmi la population de parents puis comparés et le meilleur est conservé comme premier parent. Un deuxième parent différent du premier est sélectionné de façon identique.

Contrairement au cadre mono-objectif, les parents ne sont pas comparés sur leurs fonctions objectifs mais sur la valeur de leur rang et de leur marge : l'individu avec le rang le plus petit

est privilégié et dans le cas où les individus sont de rang identique, celui avec la marge la plus importante est sélectionné. Cette méthode de sélection permet de favoriser le développement des individus représentant des solutions situées sur le front (rang le plus faible) tout en élargissant ce dernier en avantageant les plus isolés (marge la plus grande).

d) Évaluation

Comme dans le chapitre précédent, l'évaluation d'un individu consiste à déterminer pour chaque véhicule l'ordre de visite optimal des clients qui lui sont affectés en respectant l'ensemble des contraintes du DARP-PV-AN. Cet algorithme est effectué par une méthode de *Branch and Bound*, cette dernière fournissant l'ordre de visite avec la plus petite distance en respectant l'ensemble des contraintes. Cet ordre est également optimal sur l'objectif financier. Ce dernier se compose d'un coût proportionnel à la distance parcourue, et d'un coût dépendant du nombre de clients servis par le véhicule. Le nombre de clients servis n'étant pas choisi par la procédure de *Branch and Bound* mais par le NSGA-II, optimiser la distance donne la tournée optimale sur les deux objectifs.

En revanche, la fonction d'évaluation globale lançant la procédure de *Branch and Bound* sur chaque tournée d'un individu doit correctement modifier les valeurs des deux fonctions objectifs optimisées. Celle-ci est illustrée par l'*Algorithme 31*. Pour chaque tournée de S, une vérification est effectuée dans la *HashMap* $toursCreated$ contenant l'ensemble des tournées déjà rencontrées et calculées (ligne 7). Si cette dernière est présente, alors le *Branch and Bound* n'est pas lancé et la tournée précédemment calculée est utilisée (ligne 8). Dans le cas contraire, le *Branch and Bound* est lancé fournissant l'ordre de visite optimal et la tournée est sauvegardée dans $toursCreated$ (lignes 10-15). Les valeurs des deux objectifs de S sont ensuite affectées lorsque toutes les tournées ont été évaluées (lignes 27, 28).

e) Enrichissement de la population

Les tournées étant sauvegardées lors de l'évaluation afin d'éviter de recalculer des ordres de visites optimaux déjà rencontrés, il est possible de sélectionner dans cet ensemble des tournées afin de créer de nouvelles solutions comme dans le chapitre précédent. Contrairement au cadre mono-objectif ou une seule solution est générée et l'individu la représentant est inséré dans la population courante, le cadre bi-objectif voit l'insertion d'un ensemble d'individus représentant des solutions non-dominées.

Pour cela, différentes méthodes de scalarisation sont utilisées afin de réduire le problème bi-objectif à un problème mono-objectif et ainsi résoudre le problème de couverture par ensembles avec poids (WSCP) grâce à la PLNE et le solveur CPLEX. La sélection de la méthode de scalarisation est un paramètre à fournir lors de l'exécution de l'algorithme hybride. Pour chacune des méthodes de scalarisation, la réduction peut s'effectuer avec différents paramètres et peut être relancée plusieurs fois afin d'obtenir des solutions non-dominées différentes. Les nouveaux individus représentant ces solutions sont ensuite insérés dans la population courante avant le filtrage pour la génération suivante.

L'implémentation de ces différentes méthodes est décrite section 4.5 et leurs résultats sont comparés dans la section 4.7.

Algorithme 31 $evaluation$

Input parameter

S : Individual that will be evaluated, with $S.tours$ an array of tour

Local variables

$sumDistance$: Objective function distance value
$sumCost$: Objective function cost value
$allToursValid$: Boolean value saying if all tour are valid
$tourKey$: Key value of a given tour

Global variables

MAX_VALUE : Big M value
$toursCreated$: Map assigning to each computed tour key its created tour
$precedenceGraph$: Computed precedence graph necessary for evaluation of each tour

Begin

```
1   sumDistance := 0
2   sumCost := 0
3   allToursValid := true
4
5   for all tour in S.tours do
6   |   tourKey := key_tour(tour)
7   |   if toursCreated.contains(tourKey) then
8   |   |   tour := toursCreated.get(tourKey)
9   |   else
10  |   |   evaluation_tour(tour, precedenceGraph)
11  |   |   if not tour.is_valid() then
12  |   |   |   tour.distance := MAX_VALUE
13  |   |   |   tour.cost := MAX_VALUE
14  |   |   end if
15  |   |   toursCreated.put(tourKey, tour)
16  |   end if
17  |   sumDistance := sumDistance + tour.distance
18  |   sumCost := sumCost + tour.cost
19  |   allToursValid := allToursValid and tour.is_valid()
20  end for
21
22  if not allToursValid then
23  |   sumDistance := MAX_VALUE
24  |   sumCost := MAX_VALUE
25  end if
26
27  S.distance := sumDistance
28  S.cost := sumCost
End
```

f) Filtrage de la population

Le filtrage de la population consiste à sélectionner les individus les plus prometteurs parmi les parents et les enfants générés, pour constituer l'ensemble des parents de la génération suivante. Dans l'algorithme NSGA-II, les meilleurs individus sont conservés suivant leur rang et leur marge : ce n'est donc pas une comparaison directe des critères d'optimisation comme dans le cadre mono-objectif.

Le fonctionnement de la méthode de filtrage est décrit dans l'*Algorithme 32*. La première étape fusionne les individus des tableaux $parents$ et $children$ dans le tableau $parents$. Ensuite, le tableau est trié par ordre croissant de rang et, si des individus sont de même rang,

par ordre décroissant de marge. Les individus les plus prometteurs sont donc situés au début du tableau. L'étape suivante supprime les individus présents en double grâce au système de clé unique associée à chaque individu présenté section 3.3.1. Cette suppression dans la population permet d'éviter une convergence vers des solutions identiques et des optima locaux. Enfin la dernière étape sélectionne les $sizePop$ premiers individus pour former la population de la génération suivante.

Une illustration de la méthode de filtrage est proposée *Figure 4.13*.

Algorithme 32 $next\_iteration\_selection$

Input parameters

$parents$: Array of individuals that were used to create children
$children$: Array of individuals created by individuals in $parents$

Local variables

$random$: Generator of random values
$randPos$: Randomly selected position
$it1$: First iterator on $children$
$it2$: Second iterator on $children$
$indKey$: Temporary variable saving individual key

Global variable

$sizePop$: Size of the population

Begin

```
1   // Step 1. Put all individuals in children and sort them by ascending order of their rank and if
2   // they have the same rank, by descending order of their crowding
3   children.add_all(parents)
4   parents.clear()
5   sort(children)
6
7   // Step 2. Remove identical individuals
8   it1 := 0
9   while it1 < children.size() −1 do
10  |   indKey := key_ind(children[it1])
11  |   it2 := it1 + 1
12  |   while it2 < children.size() do
13  |   |   |   if indKey = key_ind(children[it2]) then
14  |   |   |   |   children.remove(it2)
15  |   |   |   else
16  |   |   |   |   it2 := it2 + 1
17  |   |   |   end if
18  |   end while
19  |   it := it + 1
20  end while
21
22  // Step 3. Keep the first half individuals of children
23  for i := 0 to sizePop − 1 do
24  |   parents.add(children[i])
25  end for
26
27  children.clear()
End
```

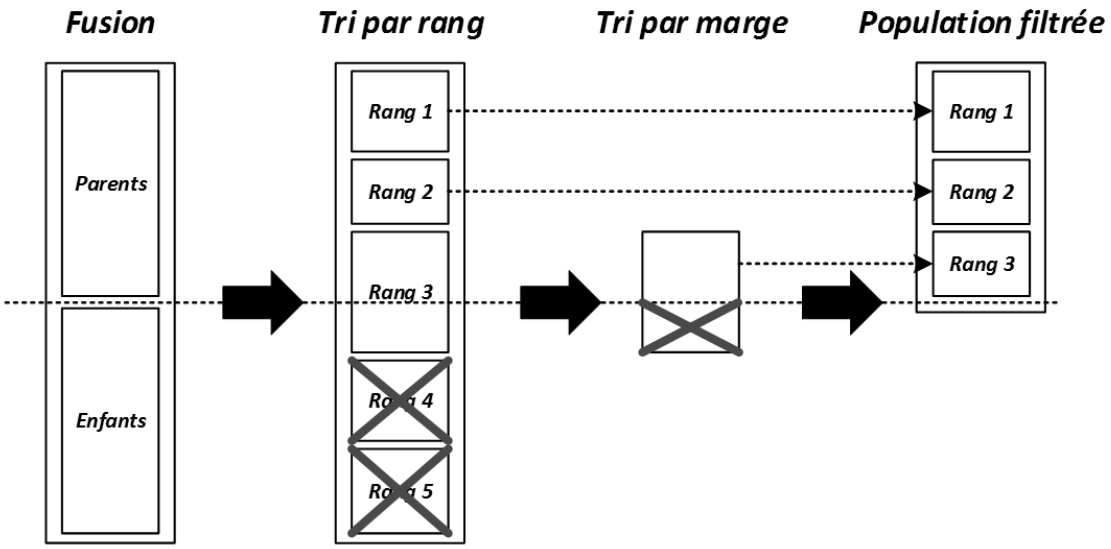


Figure 4.13 : Procédure de filtrage de l'algorithme NSGA-II par (Deb et al., 2002)

4.5 Enrichissement de la population en résolvant le problème de couverture par ensembles avec poids bi-objectif par PLNE

La dernière étape de la méthode hybride proposée comporte l'ajout, à la fin de chaque génération du NSGA-II, d'un ensemble d'individus représentant des solutions non-dominées créées grâce à l'ensemble des tournées calculées et sauvegardées lors de la partie *Branch and Bound*. Contrairement au cadre mono-objectif où cette solution est unique à chaque génération, l'approche bi-objectif favorise la génération d'un ensemble de solutions non-dominées.

Pour générer ces nouvelles solutions, le problème de couverture par ensembles avec poids (WSCP) est toujours utilisé et nécessite de transformer les deux objectifs en un seul pour pouvoir résoudre le problème avec le solveur CPLEX par PLNE. Lors d'une génération du NSGA-II, le problème est résolu plusieurs fois en mono-objectif avec une même méthode de scalarisation et des paramétrages différents, donnant ainsi un ensemble de solutions non-dominées par génération.

En plus des avantages présentés dans le cadre mono-objectif, l'utilisation de la méthode proposée offre des atouts supplémentaires lors de l'optimisation de plusieurs critères :

- Création de solutions obligatoirement non-dominées lors de la génération courante ;
- Enrichissement du front en termes de nombre de solutions ;
- Élargissement du front proposant ainsi une meilleure couverture de l'ensemble des solutions à proposer au décideur ;
- Accélération de la vitesse de convergence de l'algorithme.

Les différentes méthodes de scalarisation implémentées sont au nombre de trois :

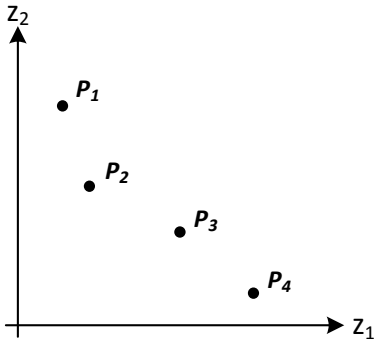
1. La recherche dichotomique par somme pondérée introduite par (Geoffrion, 1968) et utilisée dans un contexte bi-objectif par (Aneja et Nair, 1979) ;
2. La méthode ϵ-contrainte présentée par (Haimes et al., 1971) ;
3. La méthode d'agrégation proposée par (Murata et al., 2003) et utilisée par (Lacomme et al., 2006) sur un problème bi-objectif de tournées sur arcs avec capacité.

L'implémentation et l'utilisation de ces différentes méthodes sont présentées dans les sections suivantes. Les résultats de ces méthodes sont comparés dans la section 4.7.

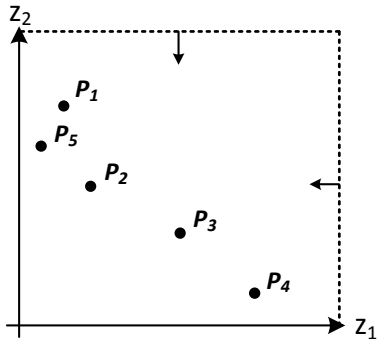
4.5.1 Recherche dichotomique par somme pondérée

Introduite dans la section 4.2.5, la méthode de recherche dichotomique par somme pondérée consiste à agréger de façon linéaire l'ensemble des critères d'optimisation z_k au sein d'un même objectif tout en affectant à chacun un poids λ_k. La méthode vise à générer l'ensemble des solutions non-dominées supportées extrêmes du WSCP bi-objectif à la génération i.

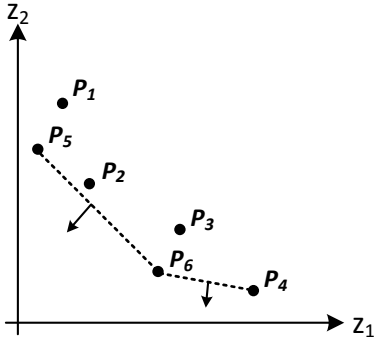
La *Figure 4.14* illustre le fonctionnement de la méthode implémentée, appelée avant la fonction de filtrage à chaque génération de l'algorithme NSGA-II. Avant le lancement de la recherche, les points P_1, P_2, P_3 et P_4 forment l'ensemble des solutions non dominées (1). La première étape consiste à recherche les solutions optimales sur le premier et sur le second objectif (2). Ensuite, la recherche dichotomique est appliquée générant de nouvelles solutions grâces aux nouvelles tournées découvertes depuis la génération précédente du NSGA-II. Enfin, l'ensemble des individus représentant les solutions générées est ajouté dans la population actuelle avant la méthode de filtrage (4). On remarque que la résolution successive du WSCP avec cette méthode au fil des générations du NSGA-II permet de découvrir des solutions non-dominées non-supportées : en effet, une solution supportée à l'itération i peut devenir une solution non-supportée à l'itération $i+1$ suite à l'apparition de nouvelles tournées.



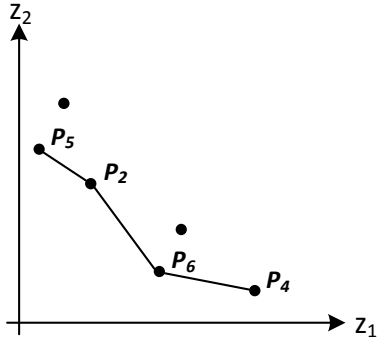
(1) - Solutions non-dominées avant le lancement de la recherche dichotomique à la génération i



(2) – Recherche des solutions optimales sur le premier et le second objectif et obtention du nouveau point P_5



(3) – Suite à la recherche dichotomique, les nouveaux points P_5 et P_6 ont été obtenus



(4) – Les individus représentant les nouvelles solutions sont insérés dans la population à la fin de la génération i formant ainsi un nouveau front

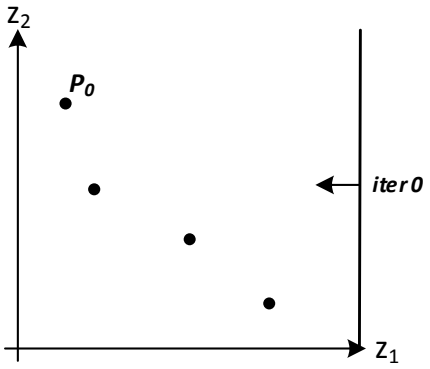
Figure 4.14 : Illustration de la recherche dichotomique par somme pondérée lors d'une génération au sein de l'algorithme NSGA-II pour le WSCP bi-objectif

4.5.2 Méthode ϵ-contrainte

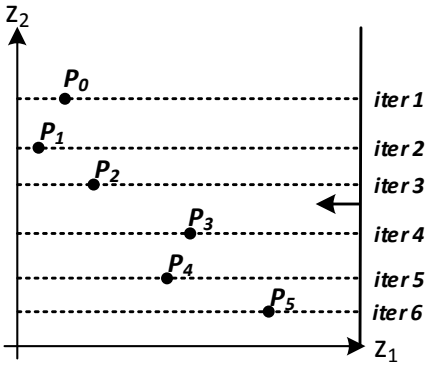
Une seconde approche implémentée est la méthode ϵ-contrainte décrite dans la section 4.2.5. Cette dernière permet de générer l'ensemble des solutions non-dominées du WSCP bi-objectif à l'itération i, contrairement à la recherche dichotomique par somme pondérée qui ne génère que des solutions non-supportées.

L'algorithme consiste à répéter la résolution du WSCP en minimisant seulement la distance : à chacun solution trouvée, une nouvelle contrainte sur le coût est ajoutée dans le modèle, forçant le prochain traitement à trouver une solution meilleure sur ce critère.

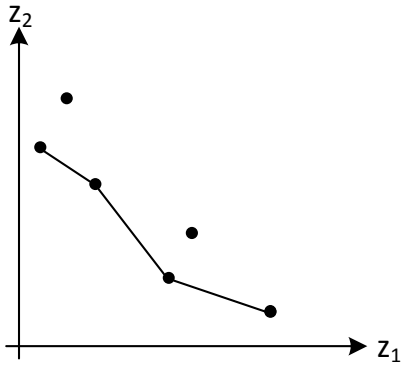
La *Figure 4.15* illustre le fonctionnement de la méthode implémentée, appelée avant la fonction de filtrage à chaque génération de l'algorithme NSGA-II. Lors de la première étape (1), une première itération du WSCP est résolue en optimisant uniquement l'objectif z_1. La solution P_0 obtenue est utilisée pour l'itération suivante ou le WSCP est résolu avec la contrainte $z_2 < P_0^2$ (P_i^2 représentant la valeur du deuxième objectif du point i). Les itérations continuent en fixant à l'itération i la contrainte $z_2 < P_{i-1}^2$, jusqu'à ce qu'aucune solution valide ne soit trouvée (2). Les nouveaux individus représentant les solutions non-dominées découvertes sont ensuite ajoutés à la population courante du NSGA-II avant la fonction de filtrage (3).



(1) – Résolution du WSCP en optimisant uniquement l'objectif z_1 sans contraintes supplémentaires : le point P_0 est obtenu



(2) – Résolution successive du WSCP en optimisant z_1 : à l'itération i la contrainte $z_2 < P_{i-1}^2$ est ajoutée dans le modèle du WSCP



(3) – Les nouvelles solutions sont insérées dans la population à la fin de la génération i formant ainsi un nouveau front

Figure 4.15 : Illustration de la méthode ϵ-contrainte lors d'une génération au sein de l'algorithme NSGA-II pour le WSCP bi-objectif

4.5.3 Implémentation de la méthode d'agrégation proposée par (Murata et al., 2003)

Contrairement aux deux méthodes précédentes, la recherche proposée par (Murata et al., 2003) nommée dans la suite du manuscrit *M-Aggregation*, et utilisée par (Lacomme et al., 2006) est un algorithme de recherche locale. Elle ne vise pas à générer l'ensemble des solutions supportées non-extrêmes comme la recherche dichotomique par somme pondérée ou encore la totalité des solutions non dominées comme la méthode ϵ-contrainte : son but est d'appliquer une recherche locale sur chaque individu représentant une solution non-dominée de l'itération courante. Pour cela, une agrégation des deux critères d'optimisation est effectuée avant d'appliquer les différentes méthodes d'exploration dépendant du problème étudié (dans leur article, (Lacomme et al., 2006) utilisent des mouvements de suppression et réinsertion de tâches et des mouvements de type 2-Opt).

Cependant l'utilisation de recherche locale pour le WSCP n'est pas privilégiée car la résolution peut être faite à l'optimum par PLNE en un temps rapide avec le solveur CPLEX. Ainsi, pour chaque individu représentant une solution non-dominée à l'itération i, une agrégation des deux objectifs est effectuée avec un vecteur de poids donnant une direction de recherche, et un WSCP est résolu à l'optimum.

La méthode d'agrégation des critères d'optimisation proposée par (Murata et al., 2003), affecte une direction de recherche à chaque solution non-dominée en fonction de sa position dans le front.

Le but est de garder une bonne couverture de ce dernier en maintenant un bon espacement entre les solutions. La fonction objectif pondérée du WSCP est la suivante :

$$\min \sum_{i \in S} w \times dist_i \times x_i + (1 - w) \times cost_i \times x_i$$

avec S l'ensemble des tournées disponibles, $dist_i$ la valeur de l'objectif distance de la tournée i, $cost_i$ le valeur de l'objectif financier de la tournée i et x_i la variable de décision déterminant si la tournée est sélectionnée ou non. Le poids w déterminé pour chaque solution s non-dominée est calculé selon la formule suivante :

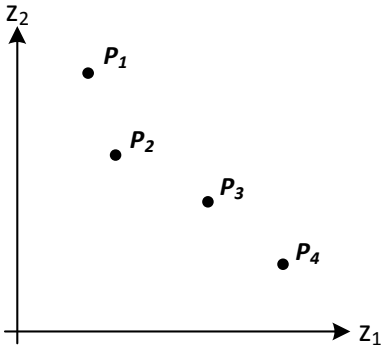
$$w = \left(\frac{dist_s - dist_{min}}{dist_{max} - dist_{min}}\right) / \left(\frac{dist_s - dist_{min}}{dist_{max} - dist_{min}} + \frac{cost_s - cost_{min}}{cost_{max} - cost_{min}}\right)$$

avec $dist_{min/max}$ la plus petite/grande valeur de distance parmi les solutions non-dominées de l'itération courante et de manière similaire, $cost_{min/max}$ la plus petite/grande valeur de coût.

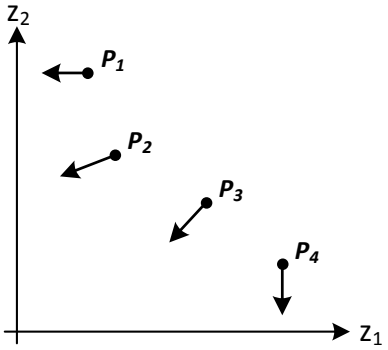
Bien que cette méthode semble avoir des similitudes avec la recherche dichotomique par somme pondérée (notamment par l'agrégation des deux objectifs avec un vecteur de poids) une différence importante est présente : la recherche dichotomique retourne toutes les solutions supportées extrêmes et appel un plus grand nombre de fois le solveur pour résoudre un WSCP au sein d'une même génération. En revanche, la méthode présentée dans cette section résout moins de modèle linéaires : moins de solutions sont trouvées mais le temps économisé permet de consacrer plus de temps au NSGA-II et au *Branch and Bound* pour

découvrir de nouvelles tournées potentiellement plus intéressantes, et donc de meilleures solutions.

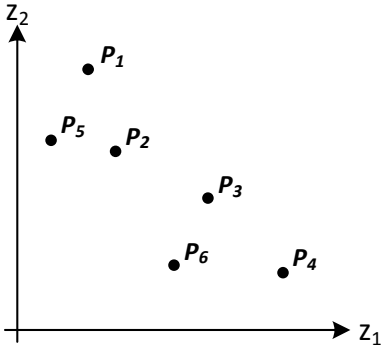
La *Figure 4.16* montre le fonctionnement de la méthode implémentée, appelée avant la fonction de filtrage à chaque génération de l'algorithme NSGA-II. Lors de la première étape (1), les individus représentant des solutions non-dominées sont marqués dans la population courante. À chaque individu marqué est affecté une direction de recherche représentée par un vecteur de poids déterminé par la formule présentée précédemment (2). Pour chaque individu marqué représentant une solution non-dominée, un WSCP est résolu en agrégeant les deux objectifs avec le vecteur de poids correspondant (3). Les individus représentant les nouvelles solutions non-dominées sont ajoutés à la population courante du NSGA-II avant la fonction de filtrage (4).



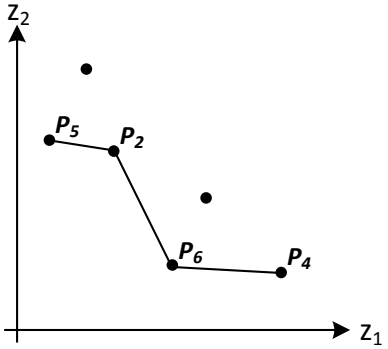
(1) - Solutions non-dominées à l'itération i avant le lancement de la méthode



(2) – Détermination des directions de recherche pour chaque solution non-dominée



(3) – Découvertes des points P_5et P_6 suite aux résolutions des WSCP pour chaque vecteur de poids déterminé en (2)



(4) – Les nouvelles solutions sont insérées dans la population à la fin de la génération i formant ainsi un nouveau front

Figure 4.16 : Illustration de la méthode M-Aggregation appelée à chaque génération du NSGA-II

a) Détermination des vecteurs de poids

La fonction $weight\_vector\_computation$ de l'*Algorithme 33* montre le calcul des différents vecteurs de poids pour chaque individu représentant une solution non-dominée. Les lignes 6-19 montrent la recherche des valeurs min/max des deux objectifs parmi S. Puis pour chaque

individu, un vecteur de poids est calculé et ajouté dans $weights$ (ligne 21-24). La fonction retourne un tableau contenant l'ensemble des vecteurs de poids.

Algorithme 33 $weight\_vector\_computation$

Input parameters

S : Array of non-dominated individuals at iteration i of NSGA-II
$sizeS$: Number of individuals in S

Output variable

$weights$: Array of weight vectors

Local variables

$dist_{min}$: Minimal distance value of individuals in S
$dist_{max}$: Maximal distance value of individuals in S
$cost_{min}$: Minimal cost value of individuals in S
$cost_{max}$: Maximal cost value of individuals in S
MAX_VALUE : Big M value
w : Computed weight

Available functions

computeW($s, dist_{min}, dist_{max}, cost_{min}, cost_{max}$) : Compute w value of individual s

Begin

1 $dist_{min}$:= MAX_VALUE
2 $dist_{max}$:= 0
3 $cost_{min}$:= MAX_VALUE
4 $cost_{max}$:= 0
5
6 **for** i := 0 **to** $sizeS - 1$ **do**
7 | **if** $dist_{min} > S[i].dist$ **then**
8 | | $dist_{min} := S[i].dist$
9 | **end if**
10 | **if** $dist_{max} < S[i].dist$ **then**
11 | | $dist_{max} := S[i].dist$
12 | **end if**
13 | **if** $cost_{min} > S[i].cost$ **then**
14 | | $cost_{min} := S[i].cost$
15 | **end if**
16 | **if** $cost_{max} < S[i].cost$ **then**
17 | | $cost_{max} := S[i].cost$
18 | **end if**
19 **end for**
20
21 **for** i := 0 **to** $sizeS - 1$ **do**
22 | w := computeW($S[i], dist_{min}, dist_{max}, cost_{min}, cost_{max}$)
23 | $weights$.add($(w, 1 - w)$)
24 **end for**
25
26 **return** { $weights$ }
End

b) Résolution du WSCP pour chaque vecteur de poids

L'*Algorithme 34* résout un problème WSCP pondéré pour chaque vecteur de poids calculé dans la fonction $weight\_vector\_computation$. Il renvoie un tableau contenant l'ensemble

des solutions générées. Les individus représentant ces solutions sont intégrés à la population en cours avant la fonction de filtrage de l'algorithme génétique.

Algorithme 34 $aggregated\_SCP\_computation$

Input parameters

$weights$: Array of weight vectors
$sizeW$: Number of vectors in $weights$
$turns$: Set of all turn created since the launch of the algorithm

Output variable

$results$: Array of non-dominated solutions obtained by solving WSCP by ILP

Local variable

obj : Aggregated objective function of WSCP

Available function

solveWSCP($turns, obj$) : Return the optimal solution of WSCP with objective obj using ILP with CPLEX

Begin

1 $results := \emptyset$
2
3 **for** $i := 0$ **to** $sizeW - 1$ **do**
4 | $obj := \sum_{t \in turns} weights[i] * dist_t * x_t + (1 - weights[i]) * cost_t * x_t$
5 | $results :=$ solveWSCP($turns, obj$)
6 **end for**
7
8 **return** { $results$ }

End

4.6 Instances

L'algorithme NSGA-II hybride a été testé sur les instances proposées dans le chapitre 2. Ces dernières ont été complétées par des données supplémentaires permettant l'optimisation du coût financier en plus de la distance. Ces modifications ont été effectuées en 4 étapes :

1. Choix du coût fixe par client transporté pour les véhicules publics ;
2. Choix du coût fixe par client transporté pour les véhicules privés ;
3. Choix du coût dépendant du trajet pour les véhicules publics ;
4. Choix du coût dépendant du trajet pour les véhicules privés.

Les modifications doivent prendre en compte les cas les plus défavorables pour l'utilisation des véhicules privés. Par conséquent, si dans les meilleures solutions fournies lors de la résolution des instances, ces derniers sont utilisés dans les cas les moins propices à leur implantation, leur utilité est démontrée.

Ainsi, le coût fixe par client transporté pour les véhicules publics (1) est fixé à 0€ car l'entreprise de transport n'a pas de dépenses fixes engendrées par chaque client supplémentaire dans ses tournées.

Le coût fixe des clients transporté par les véhicules privés (2) représente un montant versé par l'entreprise à un conducteur privé pour chaque client que ce dernier va transporter et qui n'aura donc pas besoin d'être pris en charge par un véhicule public. Pour chaque instance résolue, 5 scénarios différents ont été envisagés avec chacun un coût fixe différent : 10, 15, 20, 25 et 30€.

Contrairement au coût fixe, le coût dépendant du trajet pour les véhicules publics (3) n'est pas nul : ce dernier est calculé uniquement sur le salaire des chauffeurs (les véhicules pouvant être électriques, le coût de consommation est négligeable vis-à-vis du carburant fossile). Ainsi, le salaire moyen d'un chauffeur de la RATP est utilisé (1878€/mois), soit sur une base française de 35h/semaine, un coût de 0,21€/min $\left(1878 \times 12 \times \frac{1}{52} \times \frac{1}{35} \times \frac{1}{60}\right)$. Comme la matrice des distances est représentée par le temps de trajet entre chaque sommet, cette valeur permet de déterminer le coût d'un trajet effectué par un véhicule public.

Dans le cadre des véhicules privés, le coût dépendant du trajet (4) est calculé comme un remboursement des frais de transport du conducteur du véhicule. Le rapport publié par (Hugrel et Joumard, 2004), montre la répartition des véhicules privés en France (urbain, route et autoroute) ainsi que leur vitesse moyenne.

Tableau 4.1 : Répartition de la circulation des véhicules privés et vitesse moyenne (Hugrel et Joumard, 2004)

	Urbain	**Route**	**Autoroute**	**Moyenne**
Répartition %	32	46	22	-
Vitesse moyenne km/h	23	56	108	56,88

La vitesse moyenne d'un véhicule privé est estimée à 56,88km/h, soit 0,948km/min. De plus, la consommation moyenne du parc automobile français en 2017 est de 6,69L/100km (bilan annuel réalisé par le Comité des Constructeurs Français d'Automobile – CCFA, http://ccfa.fr/wp-content/uploads/2018/09/analyse_statistiques_2018_fr.pdf) avec un coût moyen du carburant de 1,45€/L. Ainsi, pour 100km le coût en carburant est de 9,7005€ $(1{,}45 \times 6{,}69)$, soit 0,097005€/km. Le coût dépendant du trajet pour les véhicules privés est donc estimé à 0,091€/min $(0{,}097005 \times 0{,}948)$.

Le but de ce chapitre étant d'étudier l'impact des véhicules privés sur les coûts économiques et les distances parcourues, seules les instances autorisant l'ensemble des clients à utiliser leur propre véhicule sont utilisées (PCD_20_10VP, PCD_40_20VP, PCD_60_30VP, PCD_80_40VP, PCD_100_50VP et PCD_120_60VP). Pour chacune de ces instances, les cinq scénarios sont envisagés pour le coût fixe des véhicules privés, le coût fixe des véhicules publics est de 0, le coût dépendant du trajet des véhicules publics est de 0,21€/min et le coût dépendant du trajet pour les véhicules privés est de 0,091€/min.

Toutes les instances pour le DARP-PV-AN bi-objectif sont disponibles sur la page « https://perso.limos.fr/~davbreve/DARP-PV-AN_bi-objective.html ».

4.7 Résultats

Les résultats présentés dans cette section ont été réalisés sur les instances proposées dans la section 4.6, dans le cas de la résolution du DARP-PV-AN bi-objectif. Différentes versions de la méthode de résolution hybride proposé dans ce chapitre sont utilisées et comparées.

Puisque l'objectif consiste à trouver l'ensemble Pareto-optimal pour chaque instance, une mesure appelée hypervolume est associée à chaque front. Introduite par (Zitzler et Thiele,

1999), cette indice nécessite de définir un point maximal dans l'espace des objectifs dont les valeurs sont supérieures à toutes les solutions des fronts comparés. L'hypervolume représente l'espace dominé couvert par le front de Pareto : plus la valeur mesurée est grande, plus l'espace couvert est important et donc le front est de meilleure qualité. Une autre possibilité est de mesurer l'espace non dominé par le front : dans ce cas plus l'espace est faible, plus le front est de bonne qualité. Les mesures effectuées pour le DARP-PV-AN utilisent la première proposition (maximisation de l'hypervolume) comme illustré sur la *Figure 4.17*.

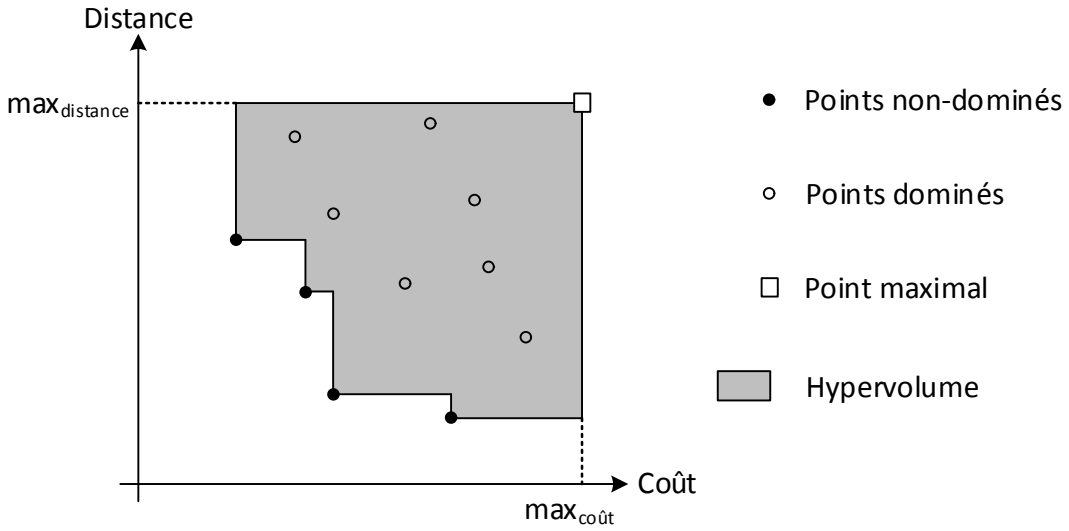


Figure 4.17 : Exemple de front de Pareto avec sa mesure d'hypervolume

Les expériences ont été réalisées dans des conditions similaires au chapitre précédent : un serveur de calcul sous Linux équipé de 3To de ram et un Intel Xeon E7-8890v3@2,5GHz avec un seul thread activé. Le solveur CPLEX en version 12.6 a été utilisé pour chaque résolution du problème de couverture par ensembles avec poids. D'après la méthode d'évaluation de (Dongarra et al., 2011), la vitesse de la machine est d'environ 3,33 GFlops et l'algorithme hybride proposé a été codé en Java.

4.7.1 Utilisation individuelle des méthodes d'enrichissement de la population

La première évaluation de la méthode hybride a été effectuée en comparant individuellement les trois méthodes proposées dans la section 4.5. Chacune d'entre elle est responsable de la résolution du problème de couverture par ensembles avec poids bi-objectif effectué à chaque itération du NSGA-II, afin d'enrichir la population pour la génération suivante. Pour cette comparaison, la méthode hybride a été lancée 4 fois sur l'ensemble des instances :

- Enrichissement de la population par la méthode de recherche dichotomique par somme pondérée uniquement (colonne *Dichotomic Search*) ;
- Enrichissement de la population par la méthode ϵ-contrainte uniquement (colonne *Epsilon Constraint*) ;
- Enrichissement de la population par la méthode *M-Aggregation* uniquement (colonne *M-Aggregation*) ;
- Non utilisation de l'enrichissement de la population (colonne *No CPLEX*).

Les résultats sont présentés dans le *Tableau 4.2* divisé en 5 parties principales, chacune correspondant à un scénario de coût fixe des clients transportés par les véhicules privés (10, 15, 20, 25 et 30€). La colonne *#ND* représente le nombre de solutions non-dominées obtenues

Tableau 4.2 : Comparaison individuelle des méthodes de résolution du WSCP bi-objectif

		Dichotomic Search			Epsilon Constraint			M-Aggregation			No CPLEX		
Name		#ND	HV	Gap%	#ND	HV	Gap%	#ND	HV	Gap%	#ND	HV	Gap%
Scenario 1	PCD_20_10VP	2	4.54E+07	0.00	2	4.54E+07	0.00	2	4.54E+07	0.00	2	4.54E+07	0.00
	PCD_40_20VP	7	4.41E+07	0.04	6	4.41E+07	0.06	7	4.41E+07	0.04	7	4.41E+07	0.04
	PCD_60_30VP	8	4.16E+07	0.00	14	4.15E+07	0.12	11	4.15E+07	0.31	4	4.12E+07	0.99
	PCD_80_40VP	13	3.81E+07	0.00	8	3.79E+07	0.57	11	3.80E+07	0.18	7	3.75E+07	1.63
	PCD_100_50VP	4	3.45E+07	0.00	5	3.45E+07	0.04	1	3.44E+07	0.21	7	3.39E+07	1.72
	PCD_120_60VP	5	3.26E+07	0.08	14	3.25E+07	0.26	1	3.24E+07	0.67	1	3.12E+07	4.25
Scenario 2	PCD_20_10VP	8	4.52E+07	0.00	8	4.52E+07	0.00	8	4.52E+07	0.00	8	4.52E+07	0.00
	PCD_40_20VP	16	4.41E+07	0.00	18	4.41E+07	0.00	18	4.41E+07	0.00	18	4.41E+07	0.00
	PCD_60_30VP	22	4.15E+07	0.27	27	4.13E+07	0.86	16	4.16E+07	0.01	20	4.07E+07	2.32
	PCD_80_40VP	21	3.80E+07	0.00	32	3.77E+07	0.72	18	3.79E+07	0.20	20	3.66E+07	3.48
	PCD_100_50VP	30	3.43E+07	0.61	40	3.39E+07	1.69	12	3.44E+07	0.30	24	3.27E+07	5.20
	PCD_120_60VP	13	3.11E+07	4.07	42	3.13E+07	3.25	9	3.23E+07	0.31	8	2.93E+07	9.45
Scenario 3	PCD_20_10VP	8	4.52E+07	0.00	8	4.52E+07	0.00	8	4.52E+07	0.00	8	4.52E+07	0.00
	PCD_40_20VP	20	4.40E+07	0.00	21	4.40E+07	0.00	19	4.40E+07	0.00	21	4.40E+07	0.00
	PCD_60_30VP	24	4.13E+07	0.27	36	4.11E+07	0.79	23	4.14E+07	0.00	29	4.02E+07	3.10
	PCD_80_40VP	14	3.77E+07	0.32	45	3.72E+07	1.42	20	3.78E+07	0.00	13	3.60E+07	4.62
	PCD_100_50VP	28	3.40E+07	1.12	45	3.36E+07	2.24	15	3.43E+07	0.38	33	3.14E+07	8.83
	PCD_120_60VP	31	3.11E+07	3.30	50	3.09E+07	4.13	16	3.21E+07	0.50	10	2.79E+07	13.29
Scenario 4	PCD_20_10VP	8	4.51E+07	0.00	8	4.51E+07	0.00	8	4.51E+07	0.00	8	4.51E+07	0.00
	PCD_40_20VP	19	4.40E+07	0.00	21	4.40E+07	0.00	20	4.40E+07	0.00	22	4.40E+07	0.00
	PCD_60_30VP	27	4.14E+07	0.00	42	4.10E+07	0.85	17	4.13E+07	0.15	32	4.05E+07	2.22
	PCD_80_40VP	20	3.76E+07	0.06	37	3.75E+07	0.29	16	3.76E+07	0.10	24	3.65E+07	3.04
	PCD_100_50VP	31	3.42E+07	0.27	42	3.36E+07	1.82	23	3.42E+07	0.16	16	3.15E+07	8.01
	PCD_120_60VP	24	3.13E+07	2.71	45	3.04E+07	5.23	18	3.19E+07	0.75	18	2.62E+07	18.57
Scenario 5	PCD_20_10VP	9	4.51E+07	0.00	9	4.51E+07	0.00	9	4.51E+07	0.00	9	4.51E+07	0.00
	PCD_40_20VP	21	4.40E+07	0.00	21	4.40E+07	0.00	17	4.40E+07	0.00	21	4.40E+07	0.00
	PCD_60_30VP	33	4.13E+07	0.15	38	4.11E+07	0.51	19	4.13E+07	0.03	31	4.04E+07	2.31
	PCD_80_40VP	22	3.77E+07	0.00	40	3.74E+07	0.89	28	3.76E+07	0.46	19	3.58E+07	5.07
	PCD_100_50VP	29	3.41E+07	0.42	49	3.32E+07	2.94	25	3.42E+07	0.10	31	3.11E+07	9.29
	PCD_120_60VP	24	3.06E+07	4.54	43	3.01E+07	6.06	20	3.16E+07	1.20	16	2.71E+07	15.26

à la fin du temps imparti, HV représente l'hypervolume du front de Pareto et $Gap\%$ représente l'écart entre l'hypervolume obtenu et le meilleur hypervolume de cette instance parmi toutes les méthodes testées.

La méthode *M-Aggregation* fourni les meilleurs résultats avec un écart moyen de 0,20% contre 0,61% pour la recherche dichotomique et 1,16% pour la méthode ϵ-contrainte. L'écart de la méthode n'utilisant pas l'enrichissement de population est plus important (4,09%) montrant l'intérêt de l'insertion de nouvelles solutions par résolution du WSCP, comme dans le cadre mono-objectif.

Le nombre de solutions non-dominées est plus important sur la méthode ϵ-contrainte avec une moyenne de 27,20 contre 18,03 pour la recherche dichotomique, 14,50 pour la méthode *M-Aggregation* et 16,23 pour celle n'utilisant pas l'enrichissement. Le nombre plus important de solutions non-dominées générées par la méthode ϵ-contrainte est justifiée par la génération de solutions non-supportées en plus des solutions supportées. En revanche, un plus grand nombre de solutions non-dominées n'indique pas nécessairement un meilleur hypervolume comme illustré sur la *Figure 4.18*.

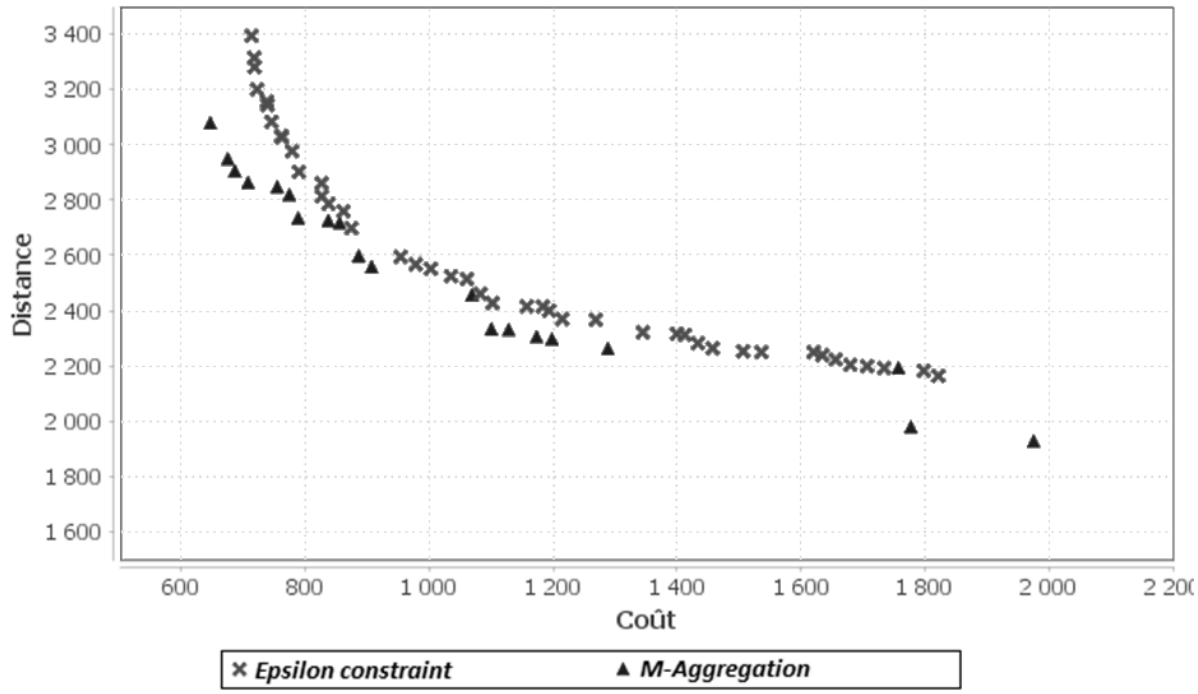


Figure 4.18 : Fronts de Pareto des méthodes ϵ-contrainte et M-Aggregation sur l'instance PCD_120_60VP avec le scénario 5 : le front obtenu par la méthode ϵ-contrainte est composé de plus de solutions mais possède une moins bonne mesure d'hypervolume

La vitesse de convergence de l'hypervolume a également été mesurée pour chacune des quatre méthodes, les résultats sont reportés sur la *Figure 4.19*. L'axe des abscisses représentant le temps écoulé depuis le début de l'algorithme et l'axe des ordonnées, l'écart entre l'hypervolume du front de Pareto courant et le meilleur front trouvé. La représentation du graphique utilise une échelle logarithmique.

On constate une meilleure qualité du front dès la première génération avec la recherche dichotomique (8,62%) contrairement à la méthode ϵ-contrainte (46,45%) et la méthode *M-Aggregation* (61,40%). Cependant, cette dernière offre une convergence plus rapide et des meilleurs résultats dès lors que 10% du temps est dépassé. On remarque également l'écart

important entre les trois méthodes utilisant l'enrichissement de la population et celle ne l'utilisant pas : les solutions de la première génération offrent un front de Pareto de moins bonne qualité (71,50%), la convergence est plus lente et le front final est très éloigné (4,21% contre 1,30% pour la méthode ϵ-contrainte, 0,72% pour la recherche dichotomique et 0,32% pour la méthode *M-Aggregation*).

Une autre approche abordée dans la section suivante, est d'utiliser différentes méthodes de manière conjointe : au sein du même algorithme, les méthodes de recherche dichotomique, ϵ-contrainte et *M-Aggregation* sont utilisées simultanément. Pour cela, deux nouveaux tests sont effectués sur chaque instance :

- Recherche dichotomique et *M-Aggregation* ;
- *M-Aggregation* et ϵ-contrainte.

Les autres combinaisons ne sont pas étudiées car elles utilisent la méthode de recherche dichotomique et la méthode ϵ-contrainte en même temps. Or la méthode ϵ-contrainte contient nécessairement l'ensemble des solutions supportées fournies par la méthode de recherche dichotomique : il est donc inutile de lancer les deux en même temps.

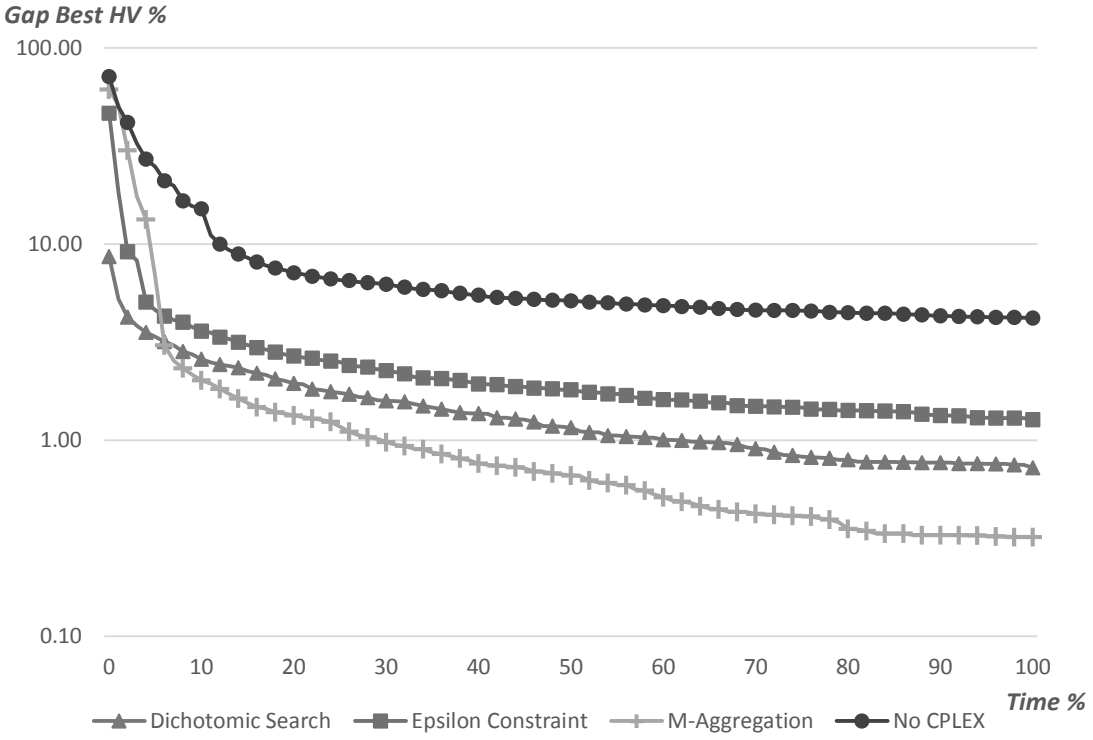


*Figure 4.19 : Convergence de l'hypervolume des méthodes de recherche dichotomique, ϵ-contrainte, M-Aggregation et sans enrichissement (*No CPLEX*)*

4.7.2 Utilisation conjointe des méthodes d'enrichissement de la population

Une seconde évaluation de la méthode hybride a été effectuée en utilisant conjointement plusieurs méthodes de résolution du WSCP bi-objectif à chaque génération de l'algorithme NSGA-II, comme précisé dans la section précédente. L'ensemble des combinaisons de méthodes ont été testées et les résultats sont reportés dans le *Tableau 4.3*.

Tableau 4.3 : Comparaison des combinaisons de méthodes de résolution du WSCP bi-objectif

		Dichotomic Search + M-Aggregation			***M-Aggregation + Epsilon Constraint***		
Name		***#ND***	***HV***	***Gap%***	***#ND***	***HV***	***Gap%***
Scenario 1	PCD_20_10VP	2	4.54E+07	0.00	2	4.54E+07	0.00
	PCD_40_20VP	7	4.41E+07	0.04	6	4.41E+07	0.06
	PCD_60_30VP	8	4.16E+07	0.00	14	4.15E+07	0.12
	PCD_80_40VP	13	3.81E+07	0.04	8	3.79E+07	0.57
	PCD_100_50VP	4	3.45E+07	0.00	3	3.45E+07	0.04
	PCD_120_60VP	5	3.26E+07	0.00	9	3.25E+07	0.22
Scenario 2	PCD_20_10VP	8	4.52E+07	0.00	8	4.52E+07	0.00
	PCD_40_20VP	17	4.41E+07	0.00	18	4.41E+07	0.00
	PCD_60_30VP	24	4.14E+07	0.58	37	4.14E+07	0.54
	PCD_80_40VP	19	3.79E+07	0.16	33	3.78E+07	0.37
	PCD_100_50VP	25	3.42E+07	0.79	38	3.39E+07	1.78
	PCD_120_60VP	12	3.11E+07	4.07	38	3.12E+07	3.77
Scenario 3	PCD_20_10VP	8	4.52E+07	0.00	8	4.52E+07	0.00
	PCD_40_20VP	19	4.40E+07	0.00	21	4.40E+07	0.00
	PCD_60_30VP	15	4.13E+07	0.26	39	4.09E+07	1.22
	PCD_80_40VP	22	3.77E+07	0.13	37	3.74E+07	1.07
	PCD_100_50VP	26	3.39E+07	1.52	49	3.36E+07	2.34
	PCD_120_60VP	27	3.14E+07	2.38	51	3.09E+07	4.10
Scenario 4	PCD_20_10VP	8	4.51E+07	0.00	8	4.51E+07	0.00
	PCD_40_20VP	20	4.40E+07	0.00	21	4.40E+07	0.00
	PCD_60_30VP	23	4.13E+07	0.12	40	4.12E+07	0.51
	PCD_80_40VP	23	3.74E+07	0.71	35	3.75E+07	0.55
	PCD_100_50VP	29	3.40E+07	0.65	48	3.34E+07	2.67
	PCD_120_60VP	25	3.09E+07	3.86	52	3.06E+07	4.77
Scenario 5	PCD_20_10VP	9	4.51E+07	0.00	9	4.51E+07	0.00
	PCD_40_20VP	18	4.40E+07	0.00	21	4.40E+07	0.00
	PCD_60_30VP	24	4.11E+07	0.68	40	4.10E+07	0.89
	PCD_80_40VP	20	3.75E+07	0.65	36	3.72E+07	1.36
	PCD_100_50VP	29	3.39E+07	1.09	50	3.32E+07	3.04
	PCD_120_60VP	29	3.07E+07	4.19	44	2.99E+07	6.51

L'activation des méthodes de recherche dichotomique et *M-Aggregation* fournie les meilleurs résultats avec un écart moyen de 0,73% contre 1,22% pour la combinaison de *M-Aggregation* et ϵ-contrainte. Cependant, ces résultats ont un écart plus important que l'utilisation de la méthode *M-Aggregation* seule vue précédemment (0,20%).

Le nombre de solutions non-dominées est plus important dans la combinaison utilisant la méthode ϵ-contrainte et *M-Aggregation* avec un front de Pareto comportant en moyenne 27,43 solution contre 17,27 pour l'autre combinaison. Cet écart s'explique encore une fois par la présence de la méthode ϵ-contrainte dans seulement une des deux combinaisons testées : cette dernière permet de générer un front de Pareto beaucoup plus dense. L'exemple de

l'instance PCD_120_60VP avec le scénario 5 de la *Figure 4.20* illustre cette différence : la méthode combinée recherche dichotomique et *M-Aggregation* contient 29 solutions sur son front de Pareto contre 44 pour l'autre combinaison, mais à cependant un meilleur hypervolume. Ce dernier est dû à l'étirement plus important du front et à un grand nombre de solutions dominant celles de la combinaison *M-Aggregation* et ϵ-contrainte.

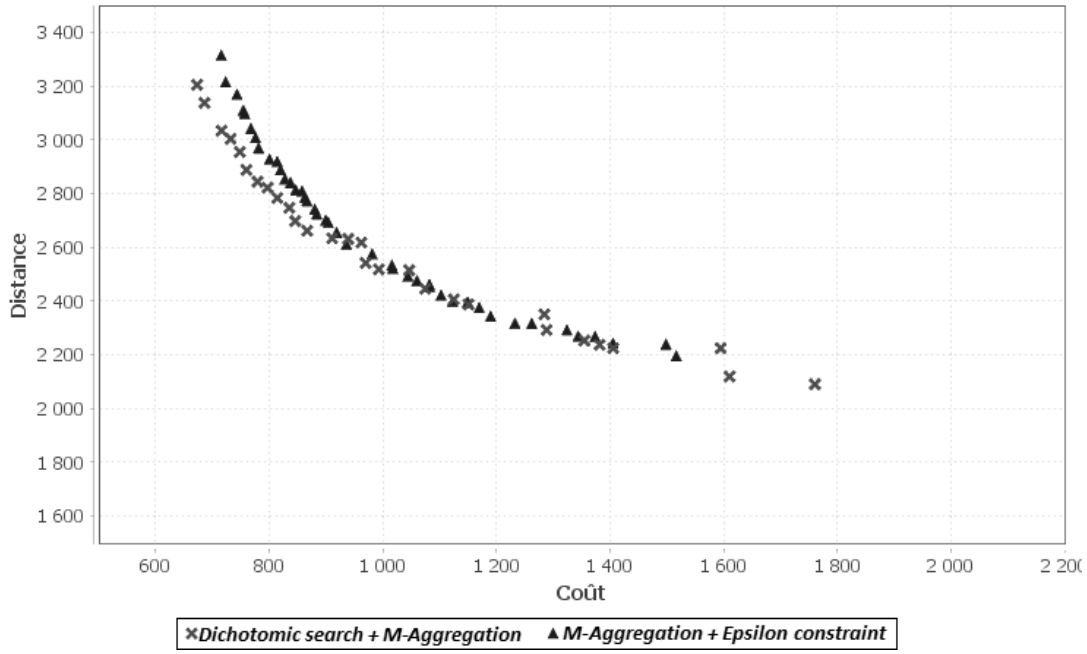


Figure 4.20 : Fronts de Pareto des méthodes combinées recherche dichotomique + M-Aggregation et ϵ-contrainte + M-Aggregation sur l'instance PCD_120_60VP avec le scénario 5

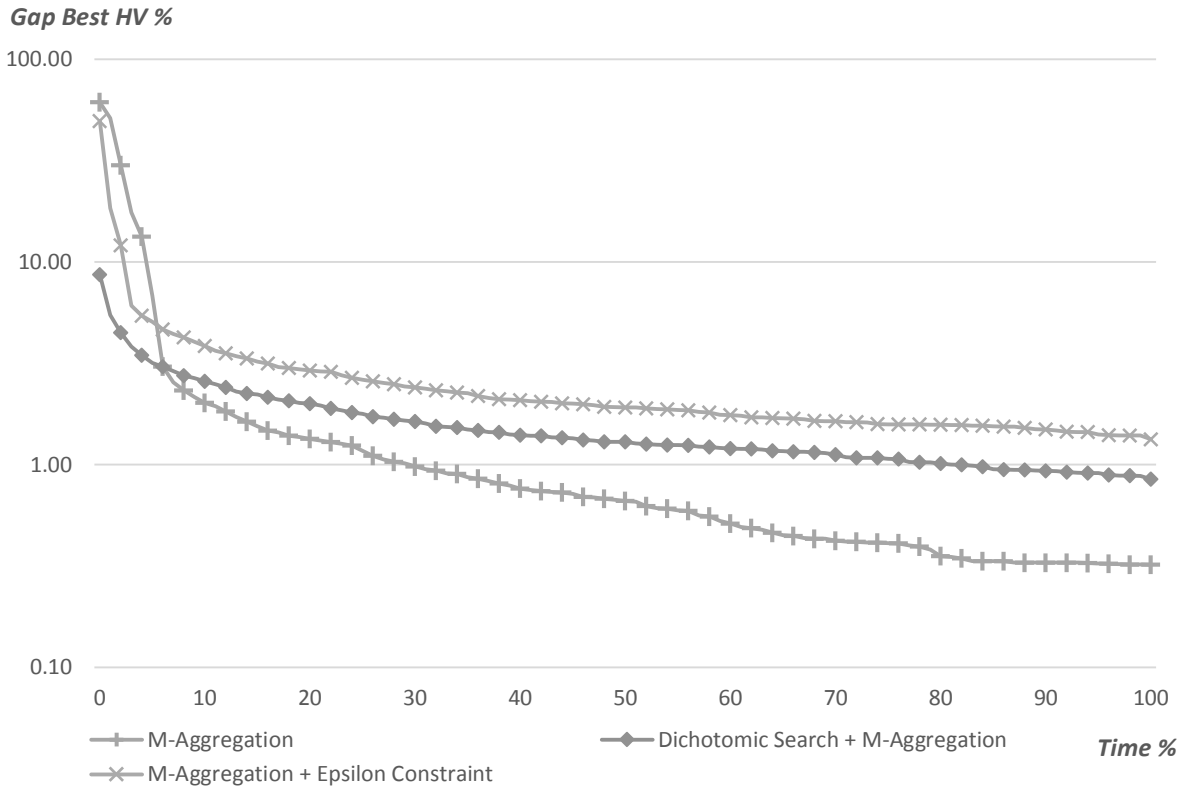


Figure 4.21 : Convergence de l'hypervolume des méthodes combinées (la méthode individuelle M-Aggregation est également ajoutée pour comparer les meilleurs résultats)

Comme dans la section précédente, la vitesse de convergence de l'hypervolume a été mesurée et les résultats sont reportés sur la *Figure 4.21*. Cette dernière illustre la domination de la combinaison recherche dichotomique et *M-Aggregation* sur la combinaison *M-Aggregation* et ϵ-contrainte : elle offre en moyenne un meilleur hypervolume de la première à la dernière génération du NSGA-II. En revanche, bien que meilleure que la méthode individuelle *M-Aggregation* sur les premières générations, elle est rapidement dépassée par cette dernière.

Les méthodes combinées ne sont donc pas aussi efficaces que la méthode individuelle *M-Aggregation.* La prochaine section apporte une réponse aux différences de résultats des méthodes présentées jusqu'à maintenant, basée sur les temps de calcul. Cette réponse permet également de proposer une autre méthode offrant des fronts de Pareto avec un meilleur hypervolume.

4.7.3 Temps de calcul CPLEX et utilisation finale de la méthode ϵ-contrainte

La répartition du temps de calcul a été étudiée et comparée suivant les différentes méthodes d'enrichissement de population activées. Cette répartition est divisée en deux parties. La première est l'algorithme NSGA-II avec le *Branch and Bound* responsable de calculer de nouvelles solutions et de découvrir de nouvelles tournées. La seconde partie est l'algorithme résolvant l'enrichissement de la population. Si le temps alloué à cette seconde partie est important, alors un ensemble plus conséquent de bonnes solutions sera créé et inséré dans la population courante à chaque génération, mais le nombre de nouvelles tournées découvertes par le NSGA-II/*Branch and Bound* sera plus faible. À l'inverse, un temps important pour la première partie permettra une plus grande découverte de nouvelles tournées, mais le nombre de solutions créées lors de l'enrichissement de la population sera plus succinct. Le but étant de trouver l'équilibre lors de la répartition du temps pour chaque partie, offrant les meilleures solutions.

Le *Tableau 4.4* reporte les comparaisons de répartition du temps des différentes méthodes. La colonne $Avg\ T(s)$ donne le temps moyen consacré à l'enrichissement de la population lors d'une génération de l'algorithme NSGA-II et la colonne $TT\%$ montre le pourcentage de temps total de la méthode passé à enrichir la population. Les algorithmes utilisant la méthode ϵ-contrainte (individuellement ou en combinaison avec une autre méthode) sont les plus longs avec un temps moyen dépassant les 12 secondes à chaque génération. La méthode de recherche dichotomique est plus rapide avec un temps moyen de 1,72 seconde. La méthode *M-Aggregation* est la plus rapide avec un temps moyen de 0,95 seconde. Cette différence de temps est liée au nombre de solutions générées : la méthode ϵ-contrainte génère l'ensemble des solutions non-dominées alors que la recherche dichotomique ne génère que les solutions supportées extrêmes.

On constate que les méthodes de combinaison et celles utilisant ϵ-contrainte sont trop lourdes en temps de calcul pour la résolution du WSCP. Les méthodes de recherche dichotomique et *M-Aggregation* laissent un temps plus important au NSGA-II/*Branch and Bound* pour découvrir de nouvelles tournées, ce qui explique la différence de qualité de front proposé.

Tableau 4.4 : Répartition du temps de calcul selon les différentes méthodes de résolution du WSCP bi-objectif

		Dichotomic Search		Epsilon Constraint		M-Aggregation		Dichotomic Search + M-Aggregation		M-Aggregation + Epsilon Constraint	
Name		Avg T(s)	TT %	Avg T(s)	TT%	Avg T(s)	TT%	Avg T(s)	TT%	Avg T(s)	TT%
Scenario 1	PCD_20_10VP	0.01	17.38	0.02	25.13	0.01	12.28	0.02	24.24	0.03	32.08
	PCD_40_20VP	0.09	29.51	0.28	39.37	0.13	36.92	0.17	41.65	0.37	50.96
	PCD_60_30VP	0.17	8.38	1.38	48.17	0.20	7.54	0.34	15.18	1.51	50.50
	PCD_80_40VP	0.36	31.92	1.28	63.40	0.44	36.37	0.83	47.87	1.48	65.25
	PCD_100_50VP	0.12	4.42	0.63	25.24	0.04	5.15	0.20	7.11	0.73	28.86
	PCD_120_60VP	0.45	3.61	2.81	22.77	0.09	1.50	0.46	2.00	1.28	12.79
Scenario 2	PCD_20_10VP	0.05	48.27	0.09	58.63	0.03	34.96	0.09	59.26	0.12	65.86
	PCD_40_20VP	0.31	44.76	1.45	76.10	0.45	55.76	0.70	68.24	1.79	76.69
	PCD_60_30VP	1.06	23.56	4.92	65.21	1.00	45.77	1.70	25.06	7.21	81.11
	PCD_80_40VP	1.55	44.21	9.90	79.21	1.56	45.10	2.48	40.08	10.56	67.58
	PCD_100_50VP	2.73	40.46	18.49	83.19	1.17	23.88	6.39	62.53	14.12	72.15
	PCD_120_60VP	1.82	3.64	29.06	71.95	0.57	6.34	3.02	6.76	16.57	30.77
Scenario 3	PCD_20_10VP	0.05	48.44	0.09	62.39	0.03	37.80	0.09	60.01	0.13	67.75
	PCD_40_20VP	0.36	58.70	2.42	70.77	0.45	58.99	0.81	70.24	2.60	78.21
	PCD_60_30VP	1.76	45.00	8.11	71.65	1.06	26.08	3.74	75.04	9.35	74.03
	PCD_80_40VP	2.08	32.34	15.69	91.00	1.16	38.85	4.33	63.48	13.61	88.01
	PCD_100_50VP	5.04	52.31	34.06	88.94	2.72	43.20	8.37	64.63	36.88	90.16
	PCD_120_60VP	4.57	13.91	50.91	65.46	0.65	3.86	6.02	10.89	53.88	51.31
Scenario 4	PCD_20_10VP	0.06	49.31	0.10	62.87	0.03	36.65	0.08	58.89	0.14	69.43
	PCD_40_20VP	0.40	58.01	1.72	77.30	0.51	64.46	0.90	71.08	2.15	77.30
	PCD_60_30VP	1.93	53.95	9.99	73.28	1.83	51.77	2.93	48.41	10.70	67.79
	PCD_80_40VP	2.08	23.34	9.71	71.88	0.75	19.85	3.75	33.24	12.03	54.56
	PCD_100_50VP	4.98	51.72	25.37	71.89	2.46	30.25	8.31	68.76	30.13	87.05
	PCD_120_60VP	6.10	11.62	38.34	42.00	2.23	16.37	6.45	12.90	49.96	54.72
Scenario 5	PCD_20_10VP	0.06	52.43	0.12	66.16	0.04	37.94	0.10	62.32	0.14	70.84
	PCD_40_20VP	0.45	61.10	1.75	77.56	0.51	63.56	0.84	73.03	2.45	84.51
	PCD_60_30VP	2.27	57.63	8.37	70.46	1.44	43.02	4.03	79.35	13.55	81.28
	PCD_80_40VP	2.21	24.80	16.39	89.62	1.80	35.70	5.21	79.49	14.69	76.39
	PCD_100_50VP	4.27	45.50	29.20	66.51	2.72	34.58	8.62	64.15	35.36	82.50
	PCD_120_60VP	4.36	9.33	50.40	48.00	2.34	15.70	9.76	26.50	35.76	30.65

Suite à ces résultats, une dernière méthode a été implémentée, combinant les avantages des méthodes ϵ-contrainte et *M-Aggregation* :

- La méthode *M-Aggregation* fournit les meilleurs résultats car son temps de calcul est plus faible que les autres méthodes testées, tout en offrant une bonne couverture du front de Pareto ;
- La méthode ϵ-contrainte est soumise à des temps de calcul plus important mais génère l'ensemble des solutions du front.

La méthode implémentée (*M-Aggregation + Final ϵ-constraint*) active la méthode *M-Aggregation* pour être utilisée lors du déroulement de l'algorithme et active la méthode ϵ-contrainte en l'utilisant une seule fois après la dernière génération. Pour confirmer l'utilité d'enrichir la population au cours de l'algorithme, cette méthode est comparée à une approche similaire ou la méthode ϵ-contrainte est lancée une seule fois après la dernière génération, mais aucun enrichissement de la population n'est utilisée auparavant (*No CPLEX + Final ϵ-constraint*). Les résultats sont reportés dans le *Tableau 4.5*.

L'activation de la méthode ϵ-contrainte en fin de l'algorithme utilisant la méthode *M-Aggregation* améliore les résultats, passant l'écart moyen de 0,20% à 0,07%. Il en est de même pour la méthode n'utilisant pas l'enrichissement au cours de l'algorithme mais uniquement à la fin avec la méthode ϵ-contrainte, passant de 4,09% à 1,24%. On constate également que l'enrichissement de la population au cours du NSGA-II guide la recherche vers de meilleurs fronts de Pareto (0,07% contre 1,24%). L'utilisation finale de la méthode ϵ-contrainte n'est donc pas suffisante pour trouver les meilleurs résultats.

La *Figure 4.22* illustre le « saut » de l'hypervolume lors de l'utilisation de la méthode ϵ-contrainte en fin d'algorithme : ce dernier est visible par la forte descente de la courbe lors de la dernière itération.

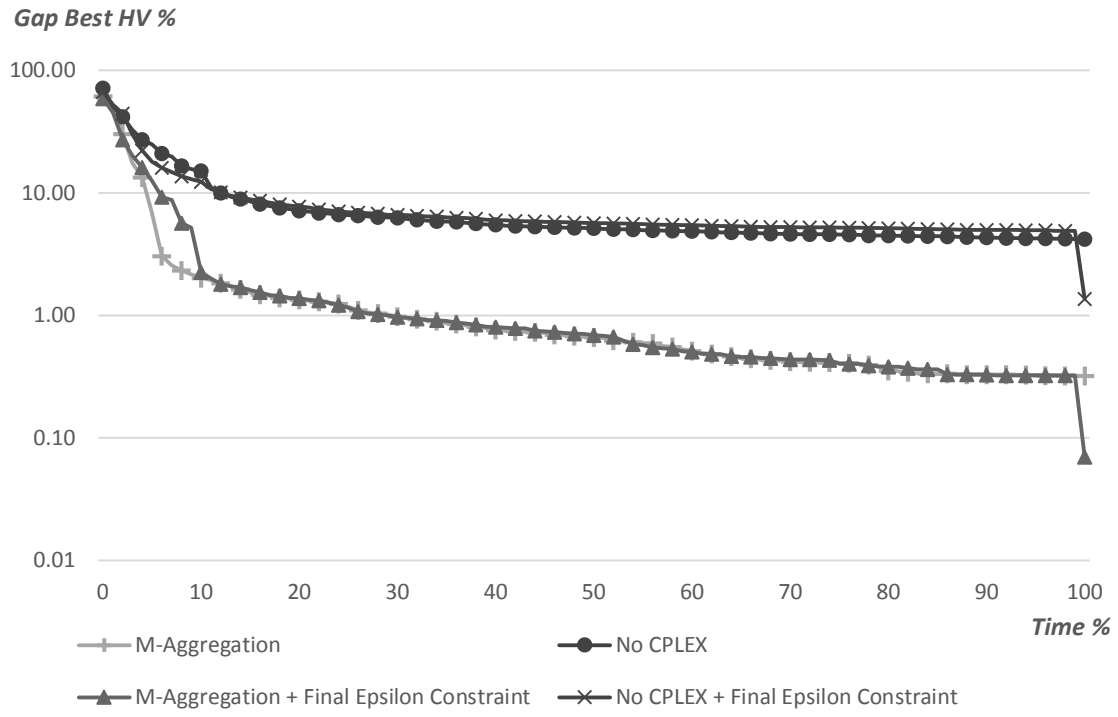
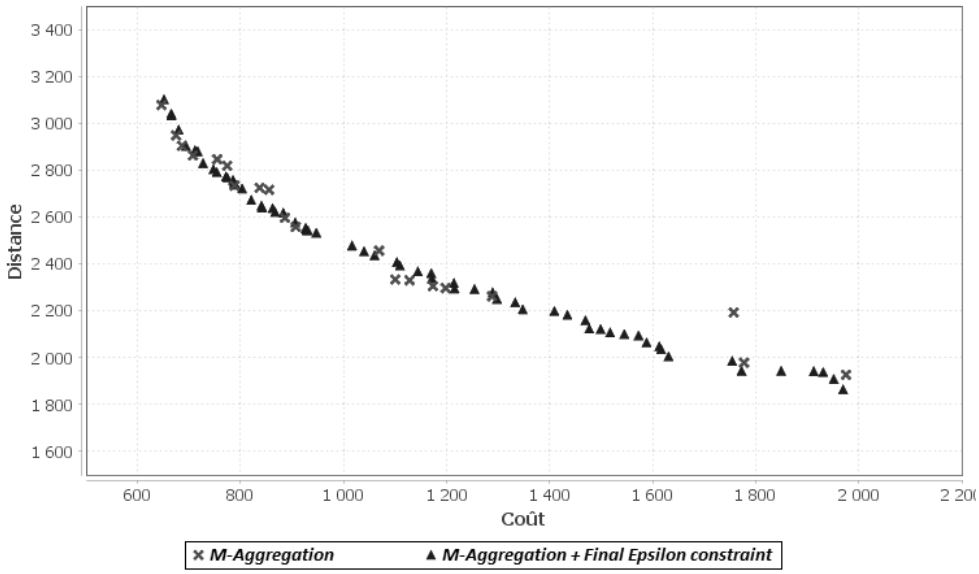


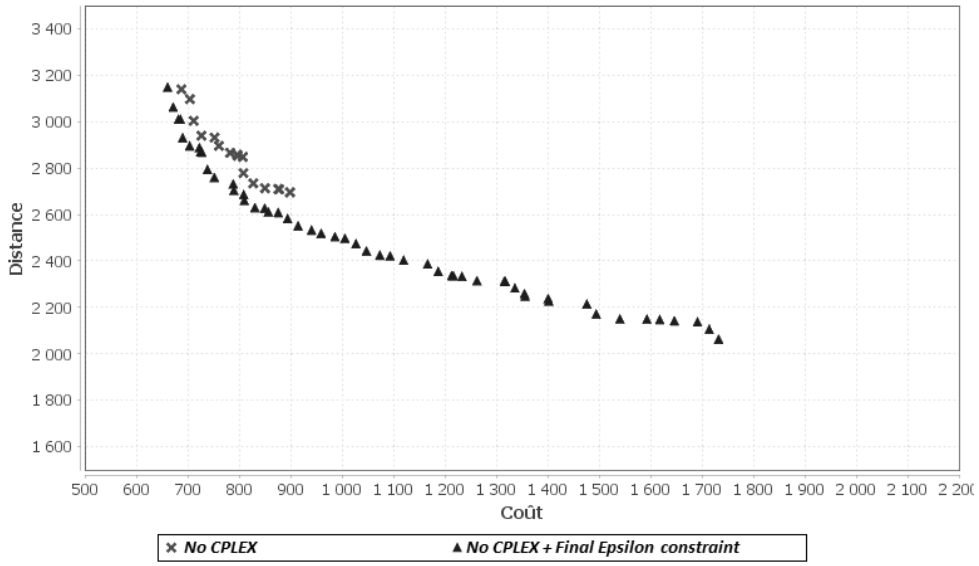
Figure 4.22 : Convergence de l'hypervolume des approches utilisant la méthode ϵ-contrainte en fin d'algorithme

Tableau 4.5 : Comparaison des approches utilisant la méthode ϵ-contrainte une seule fois en fin d'algorithme

		M-Aggregation + Final Epsilon Constraint			***No CPLEX + Final Epsilon Constraint***		
Name		***#ND***	***HV***	***Gap%***	***#ND***	***HV***	***Gap%***
Scenario 1	PCD_20_10VP	2	4.54E+07	0.00	2	4.54E+07	0.00
	PCD_40_20VP	7	4.41E+07	0.04	9	4.42E+07	0.00
	PCD_60_30VP	16	4.15E+07	0.31	4	4.13E+07	0.76
	PCD_80_40VP	18	3.80E+07	0.18	11	3.77E+07	1.08
	PCD_100_50VP	3	3.45E+07	0.17	6	3.42E+07	0.97
	PCD_120_60VP	2	3.24E+07	0.66	3	3.19E+07	2.30
Scenario 2	PCD_20_10VP	8	4.52E+07	0.00	8	4.52E+07	0.00
	PCD_40_20VP	18	4.41E+07	0.00	18	4.41E+07	0.00
	PCD_60_30VP	45	4.16E+07	0.00	41	4.12E+07	1.09
	PCD_80_40VP	38	3.79E+07	0.19	27	3.74E+07	1.38
	PCD_100_50VP	48	3.45E+07	0.00	45	3.36E+07	2.58
	PCD_120_60VP	59	3.24E+07	0.00	39	3.11E+07	3.99
Scenario 3	PCD_20_10VP	8	4.52E+07	0.00	8	4.52E+07	0.00
	PCD_40_20VP	21	4.40E+07	0.00	21	4.40E+07	0.00
	PCD_60_30VP	46	4.14E+07	0.00	43	4.13E+07	0.31
	PCD_80_40VP	41	3.78E+07	0.01	36	3.74E+07	0.95
	PCD_100_50VP	65	3.44E+07	0.00	64	3.40E+07	1.17
	PCD_120_60VP	68	3.22E+07	0.00	50	3.09E+07	4.02
Scenario 4	PCD_20_10VP	8	4.51E+07	0.00	8	4.51E+07	0.00
	PCD_40_20VP	21	4.40E+07	0.00	21	4.40E+07	0.00
	PCD_60_30VP	41	4.13E+07	0.13	46	4.09E+07	1.17
	PCD_80_40VP	50	3.77E+07	0.00	42	3.74E+07	0.79
	PCD_100_50VP	52	3.43E+07	0.00	58	3.39E+07	1.03
	PCD_120_60VP	55	3.21E+07	0.00	59	3.03E+07	5.76
Scenario 5	PCD_20_10VP	9	4.51E+07	0.00	9	4.51E+07	0.00
	PCD_40_20VP	21	4.40E+07	0.00	21	4.40E+07	0.00
	PCD_60_30VP	42	4.13E+07	0.00	43	4.09E+07	1.17
	PCD_80_40VP	44	3.76E+07	0.39	33	3.71E+07	1.68
	PCD_100_50VP	61	3.42E+07	0.00	51	3.37E+07	1.69
	PCD_120_60VP	64	3.20E+07	0.00	52	3.09E+07	3.41

La *Figure 4.23* illustre les fronts de Pareto avec l'activation finale de la méthode ϵ-contrainte. La méthode *M-Aggregation* voit le nombre de solutions non-dominées augmenter ainsi qu'une meilleure répartition de ces dernières sur l'ensemble du front. L'algorithme n'utilisant pas l'enrichissement voit son front s'étendre considérablement, sachant que l'ensemble des solutions ajoutées par la méthode ϵ-contrainte en fin d'algorithme n'étaient pas encore découvertes. Enfin l'enrichissement de la population par la méthode *M-Aggregation* au cours de l'algorithme apporte un gain non négligeable : comme illustré sur la figure, un nombre important de solutions dominent les solutions générées par la méthode n'utilisant pas l'enrichissement pendant l'algorithme NSGA-II.





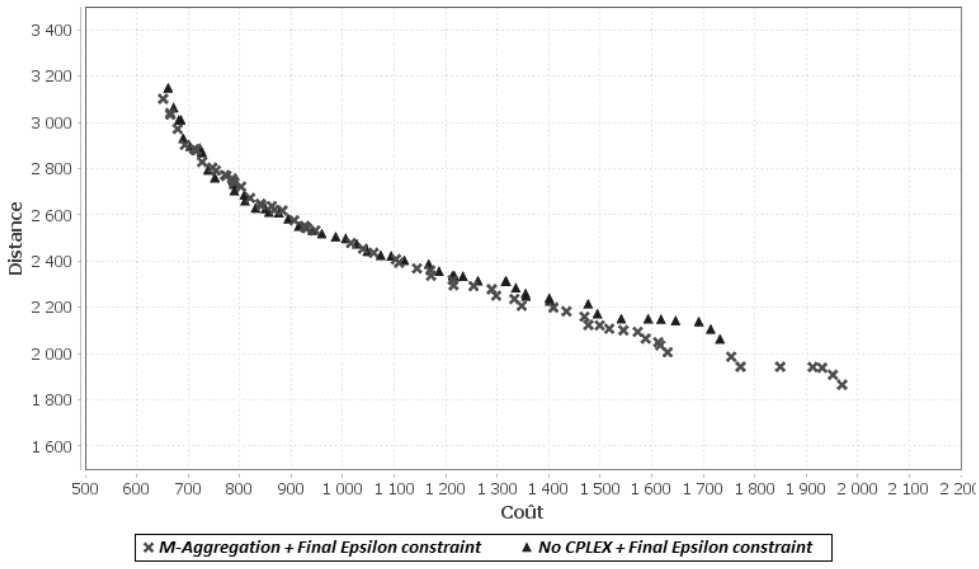


Figure 4.23 : Comparaison des fronts de Pareto sans et avec la méthode ϵ-contrainte en fin d'algorithme sur l'instance PCD_120_60VP avec le scénario 5

4.7.4 Impact des scénarios sur le coût et l'utilisation des véhicules privés

Une dernière étude a été effectuée pour évaluer l'impact du coût fixe des clients lié à chaque scénario, sur l'utilisation des véhicules privés ainsi que sur l'augmentation du coût des solutions. Les résultats sont reportés sur la *Figure 4.24*.

Pour effectuer ces comparaisons, des solutions ne comportant que des véhicules publics ont été calculées pour servir de référence. Ainsi, une solution ayant 0% de clients dans les véhicules privés, aura une distance à 100% et un coût à 100%.

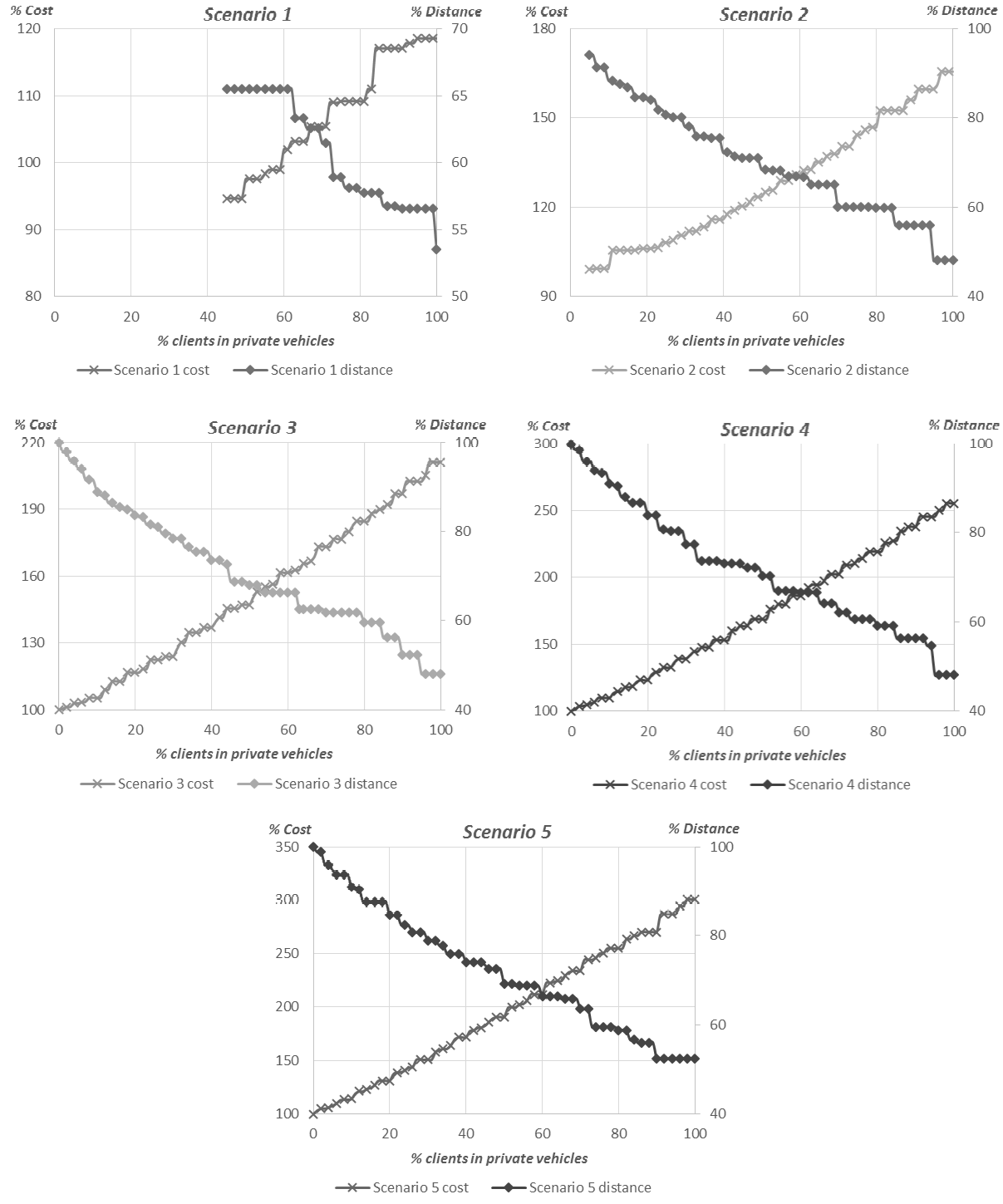


Figure 4.24 : Représentation des gains/pertes sur les coûts/distances suivant le pourcentage de clients dans les véhicules privés pour les scénarios 1 à 5

Les solutions comportant uniquement des véhicules publics ne sont pas présentes dans les scénarios 1 et 2. Ces dernières sont dominées par des solutions hybrides (comportant des véhicules publics et privés) sur la distance mais également sur le coût. Ainsi, avec un coût fixe par client peu élevé, utiliser des véhicules privés améliore les solutions sur la distance parcourue mais également sur le coût financier. Par exemple dans le scénario 1, les solutions avec le coût le plus faible (94,64%) servent en moyenne 45% des clients avec des véhicules privés pour une distance parcourue de 65,51%. En revanche, si tous les clients sont transportés par des véhicules privés, le coût grimpe à 118,55% pour une distance parcourue de 53,51%.

Une autre comparaison consiste à fixer le pourcentage de clients dans les véhicules privés et observer les différences de coût suivant le scénario. Par exemple, avec 50% de clients dans les véhicules privés, le coût est de 97,53% avec le scénario 1, 123,49% avec le scénario 2, 147,11% avec le scénario 3, 168,59% avec le scénario 4 et 190,57% avec le scénario 5, le tout pour une distance parcourue moyenne de 68%.

Choisir la valeur du coût fixe par client transporté, versé au conducteur est donc important : un coût faible sera rentable sur le plan financier mais également sur la distance parcourue par les véhicules (et donc le temps passé par les clients sur leur trajets) cependant il sera moins attrayant pour les conducteurs de véhicules privés. À l'inverse, un coût plus élevé sera plus attirant mais les coûts supplémentaires engendrés sont très importants.

4.8 Conclusion

La résolution du problème de transport à la demande avec véhicules privés et sommets alternatifs (DARP-PV-AN) bi-objectif a été effectuée. Le premier objectif représente la distance parcourue par les véhicules et le second le coût financier de chaque solution. Pour résoudre ce problème, la méthode en trois étapes proposée au chapitre précédent a été adaptée. La première étape consistant à affecter les clients dans les véhicules est effectuée par la métaheuristique NSGA-II. Chaque ordre de visite optimal des véhicules est ensuite calculé avec un algorithme de *Branch and Bound* lors de la deuxième étape. La dernière étape consiste à ajouter un ensemble de solutions à la fin de chaque génération du NSGA-II via la résolution du problème de couverture par ensembles avec poids bi-objectif. Pour cela, différentes méthodes ont été testées de manière individuelle ou combinée : recherche dichotomique, ϵ-contrainte et *M-Aggregation*.

Pour comparer les performances de ces différentes approches, les instances proposées dans le chapitre 2 ont été agrémentées de nouvelles données permettant la gestion du coût financier. Les résultats montrent que la méthode *M-Aggregation* est plus performante que les méthodes de recherche dichotomique et ϵ-contrainte. Avec des temps de résolution plus rapides, elle laisse au couple NSGA-II/*Branch and Bound* une durée plus importante pour découvrir de nouvelles tournées, et guide donc plus efficacement l'algorithme vers un front de Pareto de bonne qualité. L'utilisation de la méthode ϵ-contrainte de manière unique à la fin de l'algorithme appliquant la méthode *M-Aggregation* améliore également les solutions trouvées.

La comparaison de la méthode *M-Aggregation* avec celle n'utilisant pas l'enrichissement de population montre son impact sur la qualité des solutions proposées. Son utilisation permet une convergence plus rapide de l'hypervolume vers de meilleures valeurs. Cette comparaison prouve également l'importance de l'utilisation de la méthode au sein du NSGA-II et non uniquement à la fin de ce dernier. Contrairement au cadre mono-objectif ou son utilisation ne représentait qu'une faible partie du temps total de l'algorithme, l'enrichissement de

population par résolution du WSCP bi-objectif est plus long car l'appel au solveur CPLEX est effectué plusieurs fois à chaque itération du NSGA-II.

Comme dans le contexte mono-objectif, l'utilisation de véhicules privés améliore la distance parcourue dans les solutions et la qualité de service proposée aux clients. En revanche, une forte utilisation de ces derniers favorise des solutions avec un coût économique important. Cependant, si le coût fixe par client établit dans chaque scénario n'est pas élevé, alors les solutions ne comportant que des véhicules publics n'appartiennent plus au front de Pareto : des solutions hybrides utilisant à la fois les véhicules publics et privés auront de meilleurs valeurs sur la distance parcourue mais également sur le coût financier. Ainsi, dans un cadre particulier défini par une entreprise souhaitant faire appel à ses clients, il semble théoriquement possible d'obtenir un gain financier.

Plusieurs perspectives d'évolutions du problème peuvent être envisagées. La première serait d'optimiser la qualité de service des clients non pas une fois la distance pour chaque tournée calculée et fixée, mais indépendamment au sein d'un nouvel objectif. Une variante du problème en trois objectifs permettrait de tester l'extension de la méthode de résolution hybride proposée. Cet objectif supplémentaire engendrerait également des modifications de la méthode de *Branch and Bound* devant optimiser deux objectifs simultanément (contrairement à ce chapitre ou le coût financier n'avait aucun impact sur la tournée optimale).

Une autre approche consisterait à résoudre le DARP-PV-AN dans un contexte stochastique. Les temps de trajet sont soumis à des variations non connues durant la phase d'optimisation. L'objectif étant de générer des solutions respectant les contraintes du problème dans un contexte déterministe mais sont également robustes aux variations des temps de transport. Ce problème serait plus réaliste et constituerait une amélioration sensible des solutions appliquées dans le monde actuel.

Références

Aneja, Y.P., Nair, K.P.K., 1979. Bicriteria Transportation Problem. Manag. Sci. 25, 73‑78. https://doi.org/10.1287/mnsc.25.1.73

Cerqueus, A., 2015. Bi-objective branch-and-cut algorithms applied to the binary knapsack problem : surrogate bound sets, dynamic branching strategies, generation and exploitation of cover inequalities. Université de Nantes, Nantes.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6, 182‑197. https://doi.org/10.1109/4235.996017

Dongarra, J., Luszczek, P., Feautrier, P., Zee, F.G., Chan, E., Geijn, R.A., Bjornson, R., Dongarra, J., Luszczek, P., Philippe, B., Sameh, A., Philippe, B., Sameh, A., Dongarra, J., Luszczek, P., Steele, G.L., Gustafson, J.L., Dongarra, J., Luszczek, P., Becker, A., Zheng, G., Kalé, L.V., Pingali, K., Carro, M., Hermenegildo, M., Banerjee, U., Wismüller, R., 2011. LINPACK Benchmark, in: Padua, D. (Éd.), Encyclopedia of Parallel Computing. Springer US, Boston, MA, p. 1033‑1036. https://doi.org/10.1007/978-0-387-09766-4_155

Ehrgott, M., 2005. Multicriteria optimization, Springer Science & Business Media.

Ehrgott, M., Gandibleux, X., 2000. A survey and annotated bibliography of multiobjective combinatorial optimization. Spectr. 22, 425‑460. https://doi.org/10.1007/s002910000046

Geoffrion, A.M., 1968. Proper efficiency and the theory of vector maximization. J. Math. Anal. Appl. 22, 618‑630. https://doi.org/10.1016/0022-247X(68)90201-1

Guerriero, F., Pezzella, F., Pisacane, O., Trollini, L., 2014. Multi-objective Optimization in Dial-a-ride Public Transportation. Transp. Res. Procedia 3, 299‑308. https://doi.org/10.1016/j.trpro.2014.10.009

Haimes, Y.Y., Lasdon, L.S., Wismer, D.A., 1971. On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization. IEEE Trans. Syst. Man Cybern. SMC-1, 296‑297. https://doi.org/10.1109/TSMC.1971.4308298

Hugrel, C., Joumard, R., 2004. Transport routier - parc, usage et émissions des véhicules en France de 1970 à 2025 140.

Jozefowiez, N., Semet, F., Talbi, E.-G., 2008. Multi-objective vehicle routing problems. Eur. J. Oper. Res. 189, 293‑309. https://doi.org/10.1016/j.ejor.2007.05.055

Jozefowiez, N., Semet, F., Talbi, E.-G., 2007. The bi-objective covering tour problem. Comput. Oper. Res. 34, 1929‑1942. https://doi.org/10.1016/j.cor.2005.07.022

Lacomme, P., Prins, C., Sevaux, M., 2006. A genetic algorithm for a bi-objective capacitated arc routing problem. Comput. Oper. Res. 33, 3473‑3493. https://doi.org/10.1016/j.cor.2005.02.017

Madsen, O.B.G., Ravn, H.F., Rygaard, J.M., 1995. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. Ann. Oper. Res. 60, 193‑208. https://doi.org/10.1007/BF02031946

Murata, T., Nozawa, H., Ishibuchi, H., Gen, M., 2003. Modification of Local Search Directions for Non-dominated Solutions in Cellular Multiobjective Genetic Algorithms for Pattern Classification Problems, in: Fonseca, C.M., Fleming, P.J., Zitzler, E., Thiele, L., Deb, K. (Éd.), Evolutionary Multi-Criterion Optimization. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 593‑607. https://doi.org/10.1007/3-540-36970-8_42

Pacheco, J., Martí, R., 2006. Tabu search for a multi-objective routing problem. J. Oper. Res. Soc. 57, 29‑37. https://doi.org/10.1057/palgrave.jors.2601917

Parragh, S.N., Doerner, K.F., Hartl, R.F., Gandibleux, X., 2009. A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. Networks 54, 227‑242. https://doi.org/10.1002/net.20335

Pasia, J.M., Doerner, K.F., Hartl, R.F., Reimann, M., 2007. Solving a Bi-objective Vehicle Routing Problem by Pareto-Ant Colony Optimization, in: Stützle, T., Birattari, M., H. Hoos, H. (Éd.), Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 187‑191. https://doi.org/10.1007/978-3-540-74446-7_15

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. https://doi.org/10.1016/j.engappai.2012.03.012

Zitzler, E., Künzli, S., 2004. Indicator-Based Selection in Multiobjective Search, in: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (Éd.), Parallel Problem Solving from Nature - PPSN VIII. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 832‑842. https://doi.org/10.1007/978-3-540-30217-9_84

Zitzler, E., Laumanns, M., Thiele, L., 2001. SPEA2: Improving the strength pareto evolutionary algorithm. https://doi.org/10.3929/ethz-a-004284029

Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. 3, 257‑271. https://doi.org/10.1109/4235.797969

Conclusion et perspectives

Cette thèse aborde le problème du transport à la demande (DARP) : ce dernier consiste à planifier des tournées pour satisfaire l'ensemble des demandes des clients voulant être transportés de leurs origines vers leurs destinations. Une variante du DARP nommée DARP-PV-AN a été proposée. Cette dernière permet l'utilisation de manière conjointe des flottes de véhicules publics et privés, tout en préservant les adresses des utilisateurs. L'une des principales difficultés est la gestion de l'ensemble des contraintes du problème : capacité, fenêtres de temps, durée de trajet maximale, sélection du lieu de *pickup* et de *delivery* parmi ceux proposés par le client si ce dernier est conduit par un véhicule privé, etc. Le problème proposé a tout d'abord été étudié dans un cadre mono-objectif et deux méthodes de résolution approchées ont été proposées.

La première consiste en une extension de l'algorithme ELS proposée par (Chassaing et al., 2016), enrichie et adaptée pour résoudre le DARP-PV-AN. Elle est composée d'une fonction d'évaluation qui étend l'algorithme de (Firat et Woeginger, 2011) : cette dernière permet de vérifier que chaque tournée est valide et fournit les dates de service pour chaque client. Une méthode de recherche locale adaptative a également été utilisée au sein de l'algorithme ELS.

La deuxième méthode proposée consiste à diviser l'aspect combinatoire du DARP-VP-AN en trois parties également combinatoires. Deux d'entre elles sont résolues à l'optimum et une de manière approchée :

- L'ordre de visite effectué par les véhicules au sein des tournées est résolu à l'optimum par un algorithme de *Branch and Bound* ;
- La sélection des tournées parmi toutes celles déjà calculées pour former une nouvelle solution est également résolu à l'optimum par résolution du problème de couverture par ensembles avec poids (WSCP) par PLNE ;
- L'affectation des clients dans les véhicules est résolu de manière approchée par une métaheuristique de type algorithme génétique.

L'utilisation de cette méthode hybride assure l'optimalité des solutions proposées sur deux aspects : les tournées proposées dans la solution sont optimales, mais également la sélection de l'ensemble des tournées proposées dans la meilleure solution est optimale parmi toutes les tournées découvertes lors du calcul de l'algorithme. La sélection des tournées, effectuée en résolvant le WSCP par PLNE permet également une convergence plus rapide de l'algorithme.

Dans le quatrième chapitre, le problème est étudié dans un contexte bi-objectif. Pour cela, la notion de coût financier est introduite dans le DARP-PV-AN formant ainsi les deux objectifs : la minimisation de la distance parcourue ainsi que la minimisation du coût (la qualité de service étant toujours optimisée pour une distance parcourue donnée). Pour résoudre ce problème, l'algorithme génétique hybride proposé dans le chapitre 3 est adapté :

- L'ordre de visite effectué par les véhicules est toujours résolu à l'optimum par un algorithme de *Branch and Bound* ;
- La méthode de sélection de tournées créant une nouvelle solution, est modifiée. Il n'est en effet plus possible de générer une seule solution optimale dans un cadre bi-objectif avec la prise en compte de la notion de dominance. Pour cela différentes méthodes d'agrégation ont été testées. Toutes résolvent le problème WSCP bi-objectif à l'optimum mais fournissent des fronts de Pareto différents. La méthode ϵ-contrainte

donne la totalité des solutions formant le front tandis que la méthode de recherche dichotomique ne fournit que l'ensemble des solutions supportées extrêmes. Enfin, la méthode *M-Aggregation* ne fournit qu'un sous-ensemble des solutions supportées extrêmes mais est plus rapide et laisse donc plus de temps pour découvrir de nouvelles tournées. En post-optimisation, la méthode ϵ-contrainte est lancée en fin d'algorithme pour obtenir un front de Pareto de meilleure qualité ;

- L'affectation des clients dans les véhicules est toujours effectuée par une métaheuristique évolutionniste. Dans un contexte bi-objectif, le NSGA-II a été sélectionné.

L'utilisation de cette méthode hybride dans un cadre multi-objectif montre la capacité d'adaptation de cette dernière. Comme dans le cadre mono-objectif, les solutions proposées possèdent des tournées optimales mais également une sélection optimale des tournées parmi celles créées. Les instances du DARP-PV-AN ont été adaptées et réutilisées pour prouver l'efficacité des méthodes proposées.

Au cours de cette thèse, les travaux ont été portés sur le problème de transport à la demande avec véhicules privés et nœuds alternatifs, résolu par des méthodes hybrides combinant résolution approchées et exactes. Ces études offrent plusieurs perspectives.

À court terme, une amélioration de la fonction d'évaluation de la méthode hybride lors de la résolution du DARP-PV-AN mono ou bi-objectif est à tester via différentes approches :

- Nouvelles coupes dans l'algorithme de *Branch and Bound* ;
- Utilisation de la programmation par contraintes (PPC) pour calculer l'ordre de visite optimal des tournées ;
- Utilisation de méthodes heuristiques pour calculer un ordre de visite des tournées proche de l'optimum en un temps plus court.

Il est également possible de s'intéresser à l'insertion d'un troisième objectif à optimiser pour le DARP-PV-AN représentant la qualité de service introduite par exemple par (Cordeau et Laporte, 2003).

Sur le long terme, différents axes de recherches poursuivant les travaux actuels sont envisagés, se basant sur la méthode hybride proposée, sur les enjeux liés aux problèmes de transport à la demande, ou plus généralement, des problèmes de tournées de véhicules.

L'un de ces axes consisterait à tester la méthode hybride proposée sur d'autres problèmes. Les méthodes de couverture par ensemble ou de partition sont utilisées dans l'optimisation exacte (Baldacci et Mingozzi, 2009) pour résoudre certains problèmes de recherche opérationnelle, mais sont peu utilisées lors de l'utilisation de méthodes approchées. La méthode hybride peut être utilisée sur l'ensemble des problèmes de tournées de véhicules, mais également sur des problèmes d'ordonnancement ou encore de *bin-packing*. La condition à respecter pour utiliser la méthode étant la présence dans le problème à résoudre d'un sous-problème de partitionnement.

Un autre axe de recherche pourrait prendre en compte l'aspect robuste abordé dans les problèmes de tournées de véhicules. Ce dernier peut apparaître sous forme de variations de temps de trajet suivant le moment de la journée ou bien sur la présence ou non des véhicules privés lorsque ceux-ci sont utilisés. En effet, la fiabilité de ces derniers est moindre que les moyens de transport publics (annulation au dernier moment par exemple). Pour cela, des approches mettant en avant la robustesse des solutions proposées sont envisageables,

notamment en prévoyant la prise en charge des clients par des véhicules publics si le véhicule privé qui devait normalement les prendre en charge n'est plus disponible.

Finalement, un axe de recherche pourrait porter sur la notion d'équité pour la qualité de service entre les différents clients. Actuellement, celle-ci est principalement mesurée par la durée du trajet ainsi que des temps d'attentes. Cependant, la mesure effectuée est une moyenne, ne prenant pas en compte chaque client individuellement. Il est donc possible que la meilleure solution fournisse une qualité de service parfaite pour un ensemble de clients alors que certains subissent un temps de trajet bien supérieur couplé à de nombreuses attentes. Fournir des solutions équitables pour l'ensemble des clients pourrait donc consister à minimiser l'écart entre les qualités de service des différents clients.

Références

Baldacci, R., Mingozzi, A., 2009. A unified exact method for solving different classes of vehicle routing problems. Math. Program. 120, 347‑380. https://doi.org/10.1007/s10107-008-0218-9

Chassaing, M., Duhamel, C., Lacomme, P., 2016. 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. https://doi.org/10.1016/j.engappai.2015.10.002

Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 37, 579‑594. https://doi.org/10.1016/S0191-2615(02)00045-0

Firat, M., Woeginger, G.J., 2011. Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. Operations Research Letters 39, 32‑35. https://doi.org/10.1016/j.orl.2010.11.004