UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

École Doctorale MIPTIS

Laboratoire d'Informatique (EA 6300),
Equipe Ordonnancement et Conduite (ERL CNRS 6305)

THÈSE présenté par :

Ahmed ATAHRAN

soutenue le : 03 Décembre 2012

pour obtenir le grade de : Docteur de l'université François - Rabelais

Discipline/ Spécialité : Informatique

Étude et résolution d'un problème de Transport à la Demande multicritère

THÈSE DIRIGÉE PAR :

T'KINDT Vincent	Professeur des Universités, Université François - Rabelais Tours

RAPPORTEURS :

DHAENENS Clarisse	Professeur des Universités, Université Lille I
SEMET Frédéric	Professeur des Universités, Ecole Centrale de Lille

JURY :

DHAENENS Clarisse	Professeur des Universités, Université Lille I
LENTÉ Christophe	Maître de conférence, Université François - Rabelais Tours
PÉTON Olivier	Maître assistant, Ecole des Mines de Nantes
SEMET Frédéric	Professeur des Universités, Ecole centrale de Lille
T'KINDT Vincent	Professeur des Universités, Université François - Rabelais Tours
WOLFER CALVO Roberto	Professeur des Universités, Université Paris 13

MEMBRE INVITÉ :

BARTHÉLÉMY Jérôme	Responsable technique, ARTICQUE, Fondettes

Remerciements

Résumé

Les travaux présentés dans cette thèse visent à proposer des méthodes permettant de résoudre un problème de Transport à la Demande multicritère. Contrairement au problème de Transport à la Demande classique, connu dans la littérature sous le nom de *Dial-a-Ride Problem* (DARP), le contexte est ici multicritère puisque plusieurs critères doivent être pris en compte lors de la planification des tournées. On distingue deux versions de DARP : une version statique dans laquelle les données du problème sont connues à l'avance et une version dynamique dans laquelle les requêtes peuvent arriver au fil du temps.

Le premier travail réalisé dans cette thèse est l'étude d'un problème de DARP statique multicritère. Trois critères qui peuvent être conflictuels ont été définis : le premier consiste à minimiser le coût de transport, le deuxième critère consiste à minimiser l'insatisfaction des passagers et enfin le troisième critère consiste à minimiser la quantité de CO_2 émise par l'ensemble des véhicules. Nous avons développé une méthode évolutionnaire NSGA-II pour chercher un ensemble approximatif d'optimas de Pareto. Cette méthode intègre naturellement une méthode de diversification qui permet de faire un échantillonnage du front de Pareto.

Le second travail réalisé est l'étude d'un problème d'*Optimal Timing* dans une tournée. Ce problème consiste à calculer les dates de début de service optimales des points d'arrêts d'une tournée afin de minimiser l'insatisfaction des passagers.

Le dernier travail de cette thèse a porté sur l'étude d'un problème de Transport à la Demande dynamique dans lequel de nouvelles requêtes à traiter arrivent en cours de journée. Deux méthodes ont été proposées pour résoudre ce problème. La première est une heuristique d'insertion rapide qui consiste à insérer les nouvelles requêtes dans les tournées existantes. La seconde est une méthode arborescente tronquée qui contrairement à l'heuristique d'insertion, s'autorise à modifier largement les tournées construites antérieurement.

Mots clés : Problème de Transport à la Demande, statique, dynamique, horodatage, optimisation multicritère, algorithmes évolutionnaires multi-objectifs.

Table des matières

Introduction **17**

1 Introduction au Transport à la Demande **19**
1.1 Contexte de l'étude 19
1.2 Présentation de l'entreprise ARTICQUE 20
1.3 Qu'appelle-t-on *Transport à la Demande* ? 21
1.3.1 Le TAD aux États-Unis 22
1.3.2 Le TAD et la législation française 23
1.3.3 Objectifs et enjeux du TAD 24
1.4 Quelle configuration de TAD pour quel territoire ? 26
1.5 Quelques modes de conception du TAD 28
1.5.1 Mode en point-à-point 28
1.5.2 Mode en point de déviation 31
1.5.3 Mode en route de déviation 32
1.6 Les liens avec le DARP 33

2 État de l'art et modélisation du problème **37**
2.1 Introduction aux problèmes de tournées de véhicules 37
2.1.1 L'aspect statique dans les problèmes de tournées de véhicules 40
2.1.2 L'aspect dynamique dans les problèmes de tournées de véhicules . . 41
2.2 Revue de la littérature 41
2.2.1 Le DARP statique 42
2.2.2 Le DARP dynamique 49
2.2.3 Conclusions 53
2.3 Notions de base en optimisation multicritère 53
2.3.1 Méthodes de résolution des problèmes d'optimisation multicritères . 56
2.4 Présentation et formulation du problème 62
2.4.1 Définition et formulation du problème tricritère de DARP statique . 63
2.4.2 la version dynamique du transport à la demande 70

2.5 Conclusions 73

3 Le Problème de Timing dans une tournée 75
3.1 Définition 75
3.2 État de l'art 77
3.2.1 Le problème de minimisation de retards et d'avances pondérés dans un problème à une machine 78
3.2.2 Le problème $P\infty|prec, f_i\ convex|\sum_i f_i$ 79
3.3 Méthode de résolution du problème d'Optimal Timing dans le DARP 83
3.3.1 La fonction de coût 83
3.3.2 Algorithme de résolution 84
3.4 Conclusion 88

4 Méthode de résolution du problème de DARP statique tricritère 91
4.1 Introduction 91
4.2 Echantillonnage par l'algorithme NSGA-II 93
4.2.1 L'algorithme NSGA-II : principes généraux 93
4.2.2 Encodage d'une solution 93
4.2.3 Initialisation de la population 96
4.2.4 L'évaluation 105
4.2.5 Les opérateurs de reproduction 107
4.2.6 Stratégie de la préservation de la diversité 115
4.2.7 Sélection des solutions 115
4.2.8 Condition d'arrêt 117
4.3 Introduction aux expérimentations 118
4.4 Les indicateurs de qualité pour l'évaluation des méthodes multiobjectifs . . 119
4.4.1 Calcul de l'indicateur Q_1 119
4.4.2 Calcul de l'indicateur Q_2 120
4.4.3 Calcul de l'indicateur Q_3 120
4.5 Comparaison des performances des trois heuristiques 121
4.6 Calibrage des paramètres de l'algorithme NSGA-II 122
4.6.1 Les probabilités de croisement et de mutation 123
4.6.2 La taille de la population et le nombre d'itérations 124
4.6.3 Les résultats obtenus sur des instances aléatoires 127
4.6.4 Comparaison avec les résultats de [Cordeau et Laporte, 2003] 128
4.7 Conclusion 133

5 Le problème de Transport à la Demande dynamique 135

5.1 Introduction . . . 135
5.2 Définition et modélisation du problème de Transport à la Demande dynamique 136
5.2.1 Les données du problème . . . 137
5.2.2 Les contraintes . . . 137
5.2.3 Les critères . . . 138
5.3 Heuristique par insertion HI . . . 138
5.3.1 Description générale . . . 138
5.3.2 Indice de difficulté d'une requête I_{Dif} . . . 139
5.3.3 Degré d'attraction d'une requête par une tournée . . . 140
5.4 Méthode arborescente tronquée . . . 146
5.4.1 Description générale . . . 146
5.4.2 Etat initial . . . 148
5.4.3 Le schéma de branchement . . . 149
5.4.4 Le filtre . . . 150
5.4.5 L'évaluation . . . 150
5.4.6 La sélection . . . 151
5.4.7 La récupération . . . 152
5.5 Tests expérimentaux . . . 153
5.5.1 Les données . . . 153
5.5.2 Les résultats expérimentaux de l'heuristique HI . . . 154
5.5.3 Comparaison de l'heuristique d'insertion HI avec NSGA-II . . . 157
5.5.4 Les résultats expérimentaux de la méthode arborescente . . . 158
5.6 Conclusions . . . 159

Conclusion **163**

Annexes **169**

A Annexes **169**
A.1 Annexes . . . 169
A.2 les résultats des expérimentations de l'heuristique par insertion . . . 170
A.3 les résultats des expérimentations de l'heuristique par insertion . . . 171

Glossaire **187**

Liste des tableaux

2.1 Position du DARP par rapport aux autres problèmes de tournées ([Garaix, 2007]) 39
2.2 Comparaison des deux algorithmes évolutionnaires multiobjectifs NSGA-II et SPEA-II 62

3.1 Evolutions possibles des pentes λ_i 84

4.1 Les données liées aux véhicules 91
4.2 La taille de l'ensemble des optima de Pareto obtenus 92
4.3 Différentes categories de véhicules 118
4.4 Les résultats obtenus en combinant H_1, H_2 et H_3 122
4.5 Probabilities de croisement et de mutation 123
4.6 Les résultats obtenus avec les différentes tailles de population 124
4.7 Convergence de NSGA-II : le nombre d'itérations 125
4.8 Convergence de NSGA-II : la taille de population 126
4.9 Convergence de NSGA-II : le nombre d'itérations 127
4.10 Les paramètres de NSGA-II retenus 127
4.11 Les résultats des instances aléatoires 128
4.12 Résultats obtenus par la méthode NSGA-II 131
4.13 Résultats obtenus par la méthode de recherche Tabou ([Cordeau et Laporte, 2003]) 132
4.14 Résultats obtenus par l'algorithme génétique ([Jorgensen *et al.*, 2007]) . . . 132
4.15 Résultats obtenus par la méthode de recherche à voisinage variable ([Parragh *et al.*, 2010]) 133

5.1 Les paramètres de l'heuristique HI 155
5.2 Les résultats de l'insertion à t=0 156
5.3 Le classement moyen de la tournée choisie pour accueillir une requête selon le degré d'attraction 157
5.4 Comparaison des algorithmes HI et NSGA-II 161
5.5 Les résultats de la RBS pour t=0 162
5.6 Nombre moyen de solutions générées par la RBS 162

A.1 Les résultats de la comparaison des trois heuristiques H1, H2 et H3 sur une instance de 40 requêtes et 5 véhicules 169
A.2 Les résultats de la comparaison des trois heuristiques H1, H2 et H3 sur une instance de 80 requêtes et 5 véhicules 169
A.3 Les résultats de la comparaison des trois heuristiques H1, H2 et H3 sur une instance de 100 requêtes et 5 véhicules 169
A.4 Les résultats de l'insertion pour t=200 170
A.5 Les valeurs des trois critères (Min, Moy, Max) (insertion à t=200) 170
A.6 Les résultats de l'insertion pour t=400 171
A.7 Les valeurs des trois critères (Min, Moy, Max) (insertion à t=400) 171
A.8 Les résultats de l'insertion pour t=600 172
A.9 Les valeurs des trois critères (Min, Moy, Max) (insertion à t=600) 172
A.10 Les résultats de l'insertion pour t=600 173
A.11 Les valeurs des trois critères (Min, Moy, Max) (insertion à t=800) 173
A.12 Les résultats de RBS pour t=0 174
A.13 Les valeurs des trois critères (Min, Moy, Max) obtenues par RBS (insertion à t=0) 174
A.14 Les résultats de RBS pour t=200 175
A.15 Les valeurs des trois critères (Min, Moy, Max) obtenues par RBS (insertion à t=200) 175
A.16 Les résultats de RBS pour t=400 176
A.17 Les valeurs des trois critères (Min, Moy, Max) obtenues par RBS (insertion à t=400) 176
A.18 Les résultats de RBS pour t=600 177
A.19 Les valeurs des trois critères (Min, Moy, Max) obtenues par RBS (insertion à t=600) 177
A.20 Les résultats de RBS pour t=800 178
A.21 Les valeurs des trois critères (Min, Moy, Max) obtenues par RBS (insertion à t=800) 178

Table des figures

1.1 Positionnement du TAD 26
1.2 Modélisation spatiale du TAD 27
1.3 L'orientation des flux de déplacement 28
1.4 Les communes desservies par le service Fil Blanc dans l'agglomération de Tours 30
1.5 Mode point déviation 32
1.6 Mode route déviation 33
1.7 Positionnement du TAD par rapport au DARP 34

2.1 Classification des problèmes de PDP ([Parragh, 2009]) 39
2.2 Calssification des problèmes de PDP ([Berbeglia *et al.*, 2007]) 40
2.3 Illustration de la notion d'optimum de Pareto 54
2.4 Vecteur Ideal et vecteur de Nadir 55
2.5 Classification des méthodes d'optimisation multicritère 56
2.6 Quelques métaheuristiques pour l'optimisation multicritère 57
2.7 Approche $\epsilon-$contrainte de détermination d'un optimum de Pareto 59
2.8 La sélection parallèle dans l'algorithme VEGA 61
2.9 Exemple de requêtes de transport 65
2.10 Véhicule en avance au sommet d'arrivée v_{i+n} 68
2.11 Véhicule arrivé à l'heure au sommet de départ v_i 68
2.12 Véhicule en retard 69
2.13 Le temps d'attente à bord du véhicule 69
2.14 Les fenêtre de temps 71
2.15 Exemple d'une solution 72
2.16 3 requêtes statiques déjà planifiées et une requête dynamique 73

3.1 Un ensemble d'étapes observables lors de l'application de l'algorithme proposé dans [Garey *et al.*, 1988] 79
3.2 Algorithme d'optimal timing proposé par [Garey *et al.*, 1988] 79

3.3 Coût d'une activité j en avance 80
3.4 Coût d'une activité i en retard 81
3.5 Exemple de la fonction de coût associée à un bloc B constitué par deux travaux 82
3.6 Allures générales des fonctions de coût associées aux sommets v_i 85
3.7 La fonction du coût f_i associée à un sommet d'embarquement v_i 86
3.8 La fonction du coût f_j associée à un sommet de débarquement v_j 86
3.9 Algorithme d'Optimal Timing 89
3.10 la mise à jour des informations liées au fusionnement de deux blocs 90

4.1 Evolution du nombre d'optima de Pareto stricts 92
4.2 Itération de l'algorithme NSGA-II 94
4.3 Pseudo-code de l'algorithme NSGA-II 94
4.4 Un exemple de codage d'une solution de 5 tournées et 10 requêtes 95
4.5 Procédure de correction d'une tournée vis-à-vis de la contrainte de capacité du véhicule 97
4.6 Procédure d'insertion d'une liste de requêtes dans une tournée 97
4.7 Exemple de calcul des angles polaires 99
4.8 Construction des tournées de véhicules 100
4.9 Les étapes principales de l'heuristique H2 100
4.10 Deux tournées représentées dans un espace avec une dimension temporelle . 101
4.11 Principales étapes de l'algorithme de classification CAH 102
4.12 Exemple de combinaison des tournées alternatives générées par la méthode 2-Opt pour générer de nouvelles solutions 104
4.13 Les solutions générées à la fin du processus de combinaison des tournées . . 104
4.14 Exemple de l'application de l'opérateur 2-Opt sur une tournée de 6 sommets 105
4.15 Exemple de tournées générées avec la méthode 2-Opt représentées par leurs évaluations selon la distance parcourue et le critère de l'insatisfaction des passagers 105
4.16 Les étapes principales de l'heuristique H3 106
4.17 Algorithme de calcul des solutions non-dominées dans une population de solutions ([Kung *et al.*, 1975]) 107
4.18 Une procédure pour trier une population de solutions sous forme de sous-ensembles non-dominées 107
4.19 Exemple de deux chromosomes C et C' 108
4.20 Sélectionner deux tournées, par exemple : Vh_1 et Vh_3 109
4.21 Choisir deux points de croisement dans chaque tournée 109
4.22 Croisement de deux tournées 110
4.23 Les deux chromosomes fils 110

4.24 Suppression des doublons 111
4.25 Insertion des sommets manquants 111
4.26 Procédure d'insertion d'un sommet de destination v_d dans une tournée Vh_k 112
4.27 Procédure d'insertion d'un sommet de départ v_d dans une tournée Vh_k . . 112
4.28 Elimination des requêtes redondantes dans les chromosomes C_f et C'_f 113
4.29 Compléter chaque chromosome par les requêtes manquantes 113
4.30 Procédure d'insertion d'un sommet de départ v_j dans une tournée Vh_k . . . 114
4.31 Insertion d'un sommet de destination v_{j+n} dans une tournée Vh_k 114
4.32 Exemple de l'opérateur de mutation appliqué sur le chromosome Cf' 115
4.33 Illustaration de distance de Crowding 116
4.34 Calcul de la distance de *Crowding* 116
4.35 Exemple de processus de sélection basé sur la roulette 117
4.36 Exemple de calcul des indicateurs de qualité Q_1 et Q_2 120
4.37 Exemple de calcul des indicateurs de qualité Q_3 121

5.1 Scénario de planification d'une tournée en temps réel 136
5.2 Insertion de deux sommet dans une tournée 144
5.3 Violation de la contrainte de capacité lors de l'insertion d'une requête . . . 145
5.4 Insertion d'une requête sans violation de contrainte de capacité du véhicule 146
5.5 Heuristique d'insertion pour résoudre le problème du Transport à la Demande dynamique 147
5.6 Le branchement selon le type de sommet 149
5.7 Affectation d'un sommet statique 150
5.8 Exemple de séparation des noeuds de la RBS 153
5.9 Evolution du temps d'exécution en fonction de la taille du problème 155
5.10 Evolution du temps d'exécution de la RBS en fonction de la taille du problème 159

Introduction

Pour les collectivités locales le transport et les déplacements de personnes sont devenus un enjeu majeur dans le cadre d'une démarche de développement durable. Si les transports en commun cadencés répondent à une partie importante des demandes de déplacements, ils restent incapables de répondre à des besoins de déplacements hétérogènes dans le temps et dans l'espace. Pour cela une diversification des modes de transport est souhaitable afin de remédier aux carences des transports en commun habituels, d'assurer un service de transport de qualité sur l'ensemble du territoire et de limiter l'utilisation des véhicules personnels. Le transport à la demande (TAD) se présente comme une solution complémentaire au transport cadencé. En France, la réglementation a beaucoup évolué et elle a permis la création de nombreux services de TAD sur le territoire français. La mise en place d'un service TAD coïncide toujours avec une volonté de réaliser des économies tout en répondant aux besoins de déplacements des citoyens. L'aspect environnemental lui aussi est souvent mis en avant car le TAD est une façon de réduire l'impact du transport routier, qui génère 23% de la pollution. Dans le cadre de cette thèse nous nous intéressons à l'étude et la résolution d'un problème de Transport à la Demande multicritère.

Dans le chapitre 1 nous présentons quelques définitions du Transport à la Demande du point de vue du législateur aux Etats-Unis et en France : un rappel sur l'évolution des lois est présenté afin de dresser le cadre juridique de ce mode de transport, ce qui nous permettra de relever les points communs du TAD entre ces deux pays ainsi que les éléments de différence. Ensuite, nous détaillons ces définitions en mettant en évidence les différentes caractéristiques du TAD. Nous présentons aussi quelques conceptions de TAD qui peuvent être mises en place selon le territoire et la population à desservir. Nous finissons ce chapitre en établissant le lien avec le « Dial-a-Ride Problem » (DARP) étudié dans le domaine de la Recherche Opérationnelle.

Dans le chapitre 2, nous présentons un état de l'art des travaux récents sur le problème de DARP, en mettant d'avantage l'accent sur les problèmes multicritères ainsi que sur les versions dynamiques du DARP. Dans ce même chapitre nous présentons la définition du problème multicritère du transport à la demande que nous avons abordé dans cette thèse. Notre étude a été menée en deux étapes, ce qui nous amène à définir et distinguer deux problèmes : un problème de DARP statique multicritère et un problème de DARP dynamique. Pour chacun de ces problèmes nous présentons les données, les contraintes et les critères à optimiser. Dans l'évaluation d'un TAD, il y a trois points de vues à prendre en compte : celui du transporteur qui veut minimiser ses coûts, celui des usagers qui veulent un service de bonne qualité et celui la collectivité qui veut minimiser l'impact environnemental. Ces trois critères restent en général antagonistes quand une flotte hétérogène de véhicules

est utilisée : toute la difficulté consiste alors à trouver une solution de "juste milieu".

Le problème de DARP que nous abordons dans cette thèse peut être divisé en trois sous-problèmes, qui sont néanmoins interconnectés : un problème d'affectation des passagers aux véhicules, un problème de séquencement de divers points d'arrêts pour créer des tournées et un problème d'horodatage (encore appelé Timing). Le chapitre 3 est dédié à l'étude du sous-problème d'horodatage. Dans ce chapitre nous présentons une synthèse de l'état de l'art sur les problèmes de Timing, notamment dans le domaine de l'ordonnancement. Nous définissons ensuite le problème de Timing que nous avons considéré dans le cadre de cette thèse ainsi que l'algorithme qui a été utilisé pour le résoudre. Cet algorithme est appelé d'une manière intensive dans les méthodes de résolution qui ont été mises en place.

Le problème de DARP statique défini dans le chapitre 2 est multicritère : il faut trouver une solution de bon compromis. Pour trouver cette solution, nous avons choisi d'approximer le front de Pareto correspondant et de l'échantillonner. Le choix de la solution finale étant laissée à un décideur. Pour calculer ce front de Pareto, nous avons adapté la méthode NSGA-II au problème de Transport à la Demande qui nous intéresse. Le chapitre 4 détaille la mise en œuvre de cet algorithme génétique ainsi que ses divers mécanismes. Nous détaillons également dans ce chapitre trois heuristiques de construction de tournées utilisées pour créer la population initiale dans NSGA-II. Nous expliquons ensuite le processus suivi pour calibrer l'algorithme NSGA-II. A la fin de ce chapitre nous présentons les résultats obtenus sur des instances aléatoires et nous tentons de nous situer par rapport aux travaux existant dans le domaine en nous comparant avec des résultats issus de la littérature sur des problèmes proches mais non identiques.

Dans le chapitre 5 nous abordons le problème de DARP dynamique, dans lequel les requêtes arrivent au fil du temps. Deux méthodes ont été proposées. La première est une heuristique d'insertion qui consiste à insérer les nouvelles requêtes dans les tournées existantes. Cette heuristique est basée sur l'utilisation d'indices mesurant la difficulté posée par une requête et l'affinité entre une requête et une tournée existante. La deuxième approche est une méthode arborescente tronquée (*Recovering Beam Search*), qui contrairement à l'heuristique d'insertion, s'autorise à modifier largement les tournées construites antérieurement, en réaffectant des requêtes non encore servies à d'autres véhicules ou en modifiant le séquencement des points d'arrêts. Le chapitre se termine par la présentation de résultats expérimentaux sur des instances aléatoires.

Chapitre 1

Introduction au Transport à la Demande

Le transport à la demande (TAD) est globalement méconnu par le grand public et il est diversement apprécié suivant sa mise en oeuvre. Ce chapitre se propose de faire un rappel rapide sur les définitions juridiques et fonctionnelles du transport à la demande. Nous dégageons de la littérature quelques conceptions et structurations qui peuvent être utilisées pour mettre en place un service de TAD. Nous présentons aussi le positionnement de ce mode de transport par rapport aux autres moyens de transport public en termes d'usage collectif et de flexibilité du service fourni aux passagers. Nous présentons aussi au début de ce chapitre le contexte de cette thèse qui s'effectue dans le cadre d'un contrat CIFRE avec l'entreprise ARTICQUE.

1.1 Contexte de l'étude

Le travail présenté dans cette thèse a été réalisé dans le cadre d'une convention CIFRE (Conventions Industrielles de Formation par la REcherche) entre l'entreprise ARTICQUE, située à Fondettes, et l'équipe Ordonnancement et Conduite (ERL-CNRS 6305) du Laboratoire d'Informatique de l'Université François Rabelais de Tours (EA 6300). Le sujet de cette thèse s'articule autour du problème de transport à la demande qui consiste à planifier des tournées de véhicules pour servir un ensemble de personnes qui ont fait une demande de transport à l'avance. Nous présenterons dans la section 1.3 la définition et les différents modes de conception de ce type de transport. Ce problème attire de plus en plus l'attention des chercheurs dans des disciplines différentes, notamment dans le domaine de la Recherche Opérationnelle, dans lequel le problème de transport à la demande est souvent abordé sous le nom de « Dial-a-Ride Problem » (DARP). Généralement, le problème de DARP consiste à servir un ensemble de requêtes de transport de personnes en utilisant une flotte de véhicules limitée et habituellement située à un dépôt unique. Chaque requête est caractérisée par un point de départ et d'arrivée, des fenêtres de temps au départ ou/et à l'arrivée, un nombre de personnes à servir, etc. L'objectif est de planifier un ensemble de tournées pour servir l'ensemble de ces requêtes en respectant les contraintes du problème : les fenêtres de

temps, la capacité des véhicules, etc.

Malgré l'intérêt accordé à ce problème depuis la fin des années 1990 par les chercheurs, il reste loin d'être suffisamment étudié. Nous verrons à la fin de ce chapitre les similarités et les différences entre le TAD et le DARP. Le problème de TAD que nous abordons dans cette thèse est un problème général, qui prend en considération trois aspects régulièrement soulevés par les différents intervenants lors de la mise en place d'un service de TAD. Le premier aspect reflète principalement le point de vue du transporteur et consiste à minimiser les coûts de transport. Le deuxième aspect quant à lui reflète l'opinion des usagers qui veulent un service de bonne qualité. Le dernier aspect, porté par les élus et les collectivités locales, représente l'intérêt général et consiste à réduire l'impact environnemental en minimisant par exemple la quantité de CO_2 émise par l'ensemble des véhicules. L'objectif de cette étude est de fournir un ensemble de solutions de meilleurs compromis au décideur, qui selon ses préférences choisira une solution convenable.

Contrairement au problème DARP classique, le contexte ici est multiobjectif puisque plusieurs critères, comme le coût, la qualité de service et l'impact environnemental, doivent être pris en compte dans le calcul des solutions. Deux problèmes ont été abordés dans cette thèse : Le problème DARP statique multicritère dans lequel toutes les informations liées aux demandes des passagers et aux véhicules sont connues avant de procéder à la planification des tournées. Ce problème consiste à trouver un ensemble de solutions de meilleurs compromis. Une méthode génétique avec approche Pareto est présentée et adaptée à notre problème, que nous avons amélioré en utilisant quelques heuristiques d'initialisation afin d'accélérer sa convergence vers un front de bonne qualité. Afin de maximiser la qualité de service rendue aux passagers, un algorithme « d'optimal timing » a été utilisé pour calculer les horaires de passage des véhicules au sein de chaque tournée.

Le deuxième problème abordé dans cette thèse consiste à résoudre le problème de transport à la demande multicritère dans lequel les passagers peuvent faire des réservations en « temps réel », c'est-à-dire au moment où les véhicules sont en train de servir les autres requêtes. Ce problème peut donc être vu comme un problème de DARP statique. Deux méthodes ont été proposées : la première est une heuristique de construction rapide basée sur le principe d'attraction spatiale, tandis que la seconde est une méthode arborescente tronquée, appelée « Recovering Beam Search ». L'ensemble de ce travail a permis de développer un outil de gestion et d'optimisation du problème de transport à la demande.

1.2 Présentation de l'entreprise ARTICQUE

La société ARTICQUE située à Fondettes (France) fut fondée en 1989 par Georges-Antoine Strauch. L'activité principale de l'entreprise est essentiellement axée sur la combinaison de la cartographie et des statistiques, permettant de visualiser mais aussi de nourrir et de piloter des bases de données stratégiques. La cartographie statistique n'est plus seulement un outil de visualisation mais aussi un outil d'analyse et d'aide à la décision. Il ne s'agit plus seulement de représenter mais aussi de participer au processus de découverte des messages sous les données chiffrées ou qualitatives. La cartographie doit remettre en

cause le raisonnement, et les aller-retour entre les chiffres et les cartes.

ARTICQUE propose une large gamme de solutions qui combinent la cartographie et des statistiques et désormais un éditeur de référence dans le domaine. Le principal logiciel développé par l'entreprise est *Cartes&Données* distribué depuis 1994. La version 6 (CD6) est sortie en 2011 et existe en trois langues : français, anglais et chinois. Ce logiciel est riche en applications qui permettent de traiter les données et les visualiser sur la carte. Ces applications sont présentent sous forme de modules, chaque module permettant de faire un ensemble de traitements cartographiques, statistiques ou les deux au même temps. Le logiciel est doté d'un organigramme qui permet de mener des études cartographiques en utilisant plusieurs méthodes. Cet organigramme est conçu d'une manière à rendre le travail simple est claire pour l'utilisateur. La modélisation d'une étude peut être réalisée en utilisant en entrée des données et des cartes géographiques placées en amont des modules d'analyses de données et de traitement de cartes. En sortie on peut exporter les résultats sous forme de cartes (visualisation) et de tableaux de résultats générés dans des fichiers(.TXT, .PDF,.XLS,...). Une fois la modélisation d'une étude effectuée, elle peut être enregistrée sous forme d'un projet réutilisable.

Le groupe ARTICQUE fondé en 2011, regroupe actuellement un ensemble d'entreprises dans le domaine de la cartographie numérique et du développement d'applications informatiques : INTERCARTO, Geobs, ATMO-Expert, ARTICQUE SOFTWARE. Dernièrement le groupe a acheté l'entreprise BlayFoldex spécialisée dans la fabrication et la distribution de cartes papiers.

1.3 Qu'appelle-t-on *Transport à la Demande* ?

Malgré son apparition au début des années 1970, le transport à la demande (TAD) reste toujours mal connu, et plusieurs définitions ont été utilisées pour introduire ce mode de transport. Face à la multitude des définitions qui sont données à ce mode de transport, il a été nécessaire, d'une part, de présenter quelques définitions et conceptions de ce mode de transport en France et aux État-Unis et, d'autre part d'en présenter les enjeux et les objectifs. Commençons par définir clairement ce que nous entendons par Transport à la Demande. En particulier, nous insistons sur les définitions dans deux pays : États-Unis et France. Les premières définitions du TAD ont été données par le législateur. Par la suite, ces définitions officielles ont été complétées afin de mieux préciser les objectifs et le fonctionnement de ce mode de transport.
Castex [Castex, 2007] fournit un tour d'horizon sur les définitions des problèmes de transport à la demande, ainsi que la chronologie de l'apparition et de l'évolution du TAD, en rapportant les différentes caractéristiques et idées reçues sur le TAD. Castex présente aussi un état de l'art sur le TAD dans plusieurs pays du monde, notamment en Europe et en Amérique du nord. Les lecteurs intéressés y trouveront de nombreuses informations relatives au Transport à la Demande et la présentation de plusieurs points de vues, notamment du point de vue géographique.

Nous ne présentons ici que quelques éléments nécessaires à la compréhension de ce mode de transport, et nous nous limitons à la présentation du TAD dans les deux pays que sont les États-Unis et la France. Nous avons choisi ces deux pays parce que le TAD est né aux

États-Unis mais que nous nous intéressons dans cette étude aux TAD en France.

1.3.1 Le TAD aux États-Unis

Le TAD aux États-Unis a connu un développement significatif sous plusieurs appellations, à savoir *Paratransit, Dial a Bus, Dial a Ride, etc.* Le TAD en tant que transport public est apparu aux États-Unis avec la loi de 1966 : « *the Amendment to the Urban Mass Transportation Act* » ([Smerk, 1991]). Cette loi comprend trois tendances, dont une qui prévoit de lancer un programme de recherche sur de nouveaux modes de transport urbain. C'est lors du congrès de 1968 intitulé « *Tomorrow's Transportation : New Systems for the Urban Futur* » que les intervenants attirent l'attention sur de nouveaux systèmes de transport public, dont le transport à la demande. Deux ans plus tard, en juillet 1970, onze régions sont choisies pour expérimenter ces formes innovantes de transport, dont une pour le TAD « demand-responsive buses » destinée à desservir un espace de faible densité en population. Mais c'est seulement à partir des années 1990 que ce service va réellement se développer avec la loi, « *The American with Disabilities Act* » (ADA) qui favorise le développement de services de transport en faveur des personnes à mobilité réduite (PMR) et des personnes âgées.

La première définition a été donnée par l'*American Public Transportation Association* (APTA) sous la forme : « *non-fixed route service utilizing vans or buses with passengers boarding and alighting at pre-arranged times at any location within the system's service area. Also called "dial-a-ride"* ».
Une deuxième définition donnée par la *Federal Transit Administration* (FTA) [Kessler, 2004], un peu plus précise, vient compléter la première, et fréquemment reprise comme référence : «*Passenger cars, vans or small buses operating in response to calls from passengers or their agents to the transit operator, who then dispatches a vehicle to pick up the passengers and transport them to their destinations. A demand response operation is characterized by the following : (a) The vehicles do not operate over a fixed route or on a fixed schedule except, perhaps, on a temporary basis to satisfy a special need ; and (b) typically, the vehicle may be dispatched to pick up several passengers at different pick-up points before taking them to their respective destinations and may even be interrupted en route to these destinations to pick up other passengers*».

Ces deux définitions officielles permettent de se faire une première idée de la vision américaine du transport à la demande. Elles s'accordent sur le fait que les trajets sont libres et sont fixés selon les demandes des passagers. Toutefois, la question de flexibilité dans ces définitions reste un peu vague, ce qui laisse une marge de manoeuvre aux opérateurs de transport lors de la mise en place du service de transport à la demande. Il est vrai que toutes les définitions s'accordent sur la caractéristique de la réservation préalable, mais elles ne donnent pas plus de précisions si la réservation doit être fait la veille, une heure avant le départ ou s'il est possible de faire une réservation en temps réel. Donc le TAD est défini dans son cadre juridique seulement avec des caractéristiques principales et la façon de le mettre en place est laissée aux organisateurs du transport pour qu'ils choisissent la conception la plus appropriée.

1.3.2 Le TAD et la législation française

En France, le TAD a connu une progression remarquable durant ces 30 dernières années. Malgré le caractère flexible des services de TAD, des textes de lois importants ont été mis en place afin d'encadrer le fonctionnement de ce mode de transport qui touche actuellement tous les types d'espaces (urbains, périurbains et ruraux) et qui est utilisé pour servir l'ensemble du public, comme dans les transports publics cadencés, ou servir une catégorie spécifique de passagers, comme des élèves, des personnes âgées ou des personnes à mobilité réduite. Le TAD est offert à la place (chaque personne transportée doit avoir un siège dans le véhicule) et les véhicules doivent être capable de transporter au moins quatre personnes, y compris le chauffeur (le service des taxis, tel qu'il est connu, c'est-à-dire s'il n'accepte pas de regrouper plusieurs personnes pour les transporter dans un seul trajet, ne fait pas partie du transport à la demande).

Le TAD, en tant que mode de transport public, a fait son apparition en France dans l'article 29 de la *Loi d'orientation des Transports Intérieurs* (LOTI) du 30 décembre 1982. Sa définition est précisée par un décret d'application daté du 16 août 1985 : un TAD « *est un service collectif offert à la place, déterminé en partie en fonction de la demande des usagers et dont les règles générales de tarification sont établies à l'avance et qui sont exécutés par des véhicules dont la capacité minimale est fixée par décret* » (article 26).

Dans ce même décret, l'article 27 permet aux autorités organisatrices de réaliser des services à la demande pour des catégories spécifiques d'usagers (personnes à mobilité réduite, demandeurs d'emploi,...).

C'est le décret du 31 Décembre 1985 relatif aux services publics à la demande de transports routiers de personnes qui fixe la capacité minimale à quatre places, y compris celle du conducteur. La circulaire de 14 février 1986 précise les modalités générales d'application du décret du 16 Août 1985 et vient conclure cette première phase d'avancées réglementaires en faveur du TAD. Cette dernière indique que le TAD permet « *une meilleure adaptation aux caractéristiques locales tant géographiques que démographiques et offre la possibilité pour les autorités organisatrices et les exploitants de transport de satisfaire des besoins qui risqueraient de n'être pas pris en compte* » ([GART, 1997]).
Et afin d'élargir l'utilisation du TAD sur tous les territoires, la loi de *Solidarité et Renouvellement Urbain* (SRU) du 13 décembre 2000 a permis l'organisation du service de transport à la demande par l'ensemble des autorités organisatrices de transport grâce à l'article 104 qui modifie le II de l'article 7 de la LOTI. Ainsi l'ensemble des territoires, qu'ils soient urbains, périurbains ou ruraux, peuvent disposer d'un service de TAD sous réserve que l'autorité compétente possède la compétence de transport ([M.BLONDEAU, 2004]).

Une dernière évolution a eu lieu avec la loi de 23 février 2005 relative au développement des territoires ruraux. L'article 54 précise des éléments de substitution qui peuvent être mis en place en cas de carence dans l'offre de transport, sans toutefois préciser la notion de carence.

Ces lois ont fixé le cadre juridique du TAD tout en lui laissant une marge de manoeuvre, ce qui a donné plus de souplesse aux collectivités locales pour l'expérimentation de ce mode de transport sur leurs territoires.

Les études statistiques et les inventaires qui ont été menés durant les 20 dernières années sur le Transport à la Demande sont incomplets, soit à cause de l'absence de données, soit parce qu'ils ont été effectués seulement sur quelques expériences à un niveau local. Souvent ces études sont organisées par des organismes d'Etat ou des collectivités territoriales, comme par exemple : Le Centre d'Etudes sur les Réseaux, les Transports, l'Urbanisme et les constructions publiques (CERTU), le Groupement des Autorités Responsables de Transport (GART), l'Union des Transports Publics et ferroviaires (UTP), etc. En 1992, le CERTU a réalisé une étude auprès des conseils généraux et a dénombré 40 départements organisant du TAD sur 250 zones. En 1997, un nouvel inventaire est réalisé par le GART pour tenter de couvrir l'ensemble des TAD français. Le rapport qu'ils ont publié fait état de 41 conseils généraux et de 37 Autorités Organisatrices de Transport Urbain (AOTU) qui organisent du TAD ; cette étude a constitué la référence jusqu'au début des années 2000. A partir de cette date plusieurs études sur les TAD seront menées, qui visent à montrer un panel d'expériences représentatives. En 2005 un rapport a été publié par l'UTP sur l'utilisation de TAD dans les milieux urbains, 71 services étaient concernés par cette étude.

Que ce soit en France ou aux États-Unis, toutes les définitions comportent des similitudes mais aussi des divergences au niveau de la façon de présenter le service ou de le concevoir. Toutes les définitions s'accordent sur le principe de base que la réservation doit être faite au préalable, mais les conceptions, les utilisations et les modes de fonctionnement des TAD restent différents d'un pays à l'autre ou d'un auteur à l'autre. Ce mode de transport offre aux usagers la possibilité de choisir leur horaires de service ainsi que les points de desserte. Les tournées sont calculées en se basant sur les choix des passagers en terme de temps et d'espace, c'est-à-dire que l'offre s'adapte à la demande des usagers et pas l'inverse comme c'est le cas dans le transport cadencé. Le TAD peut être mise en place sous plusieurs formes que nous présentons dans la section 1.5. La flexibilité de ce mode de transport collectif donne plus de marge de manœuvre aux organisateurs de ce service d'offrir une desserte qu'ils considèrent adéquate pour la population et la zone géographique concernée. Donc on peut dire que le Transport à la Demande est une notion générale qui regroupe tous les services publics de transport dont tout ou partie ne s'effectue qu'à la demande ce qui le distingue des transports réguliers qui s'effectuent en fonction de programmes prédéfinis.

1.3.3 Objectifs et enjeux du TAD

Après les définitions présentées dans la section précédente, nous détaillons quelques objectifs et enjeux de ce mode de transport mis en exergue par les différents acteurs. Souvent les autorités organisatrices des transports, les élus et les usagers ne partagent pas le même point de vue sur ce mode de transport. L'un de ces points de vue affirme que le TAD peut être qualifié de transport collectif public novateur qui peut être mis en place aussi bien dans des zones rurales qu'urbaines ou périurbaines, et que sa flexibilité lui donne la capacité de s'adapter aux demandes de déplacements des usagers dans les différentes zones desservies.

Dans le même sens, le TAD est vu aussi comme l'une des stratégies de gestion de la mobilité dans des zones à faible densité, pendant des périodes de faibles demandes de transport ou comme la possibilité d'offrir un moyen de transport aux personnes qui ne peuvent pas utiliser le réseau de transport régulier, telles les personnes âgées ou à mobilité réduite. Donc on peut dire que le TAD est une solution qui peut être utilisée également pour augmenter l'offre en matière de déplacements en zones rurales.

L'une des caractéristiques principales du TAD est la réservation préalable qui le différencie en premier lieu des lignes régulières, cette caractéristique est vue différemment par les différents intervenants. D'une part, comme la demande est connue à l'avance et le transporteur peut s'organiser en adaptant l'itinéraire et le véhicule en fonction de ses commandes, ce qui permet également d'éviter que les véhicules roulent à vide. Par conséquent, elle permet de rationaliser les coûts. D'autre part, la réservation est aussi un point faible car elle peut être perçue comme une contrainte. En effet, l'usager est obligé de connaître ses déplacements à l'avance ce qui laisse peu de place aux imprévus, et pour être plus attractifs, les nouveaux TAD doivent proposer des délais de réservation plus courts (jusqu'au dernier moment). Pour cela il faut se doter des nouvelles technologies de communication et d'information et d'un algorithme qui permet de résoudre le problème dans un temps raisonnable. Actuellement la majorité des TAD présentent un service avec une réservation la veille ce qui contribue à diffuser une vision négative du TAD. Un service de transport à la demande dynamique avec un système de réservations le plus proche possible du temps réel devient donc de plus en plus nécessaire pour améliorer le service et attirer plus de passagers.

La flexibilité du TAD dépend de plusieurs paramètres comme notamment l'adéquation du service aux exigences des passagers. Dans ce contexte, ce transport flexible se situe d'une part entre les transports publics réguliers et d'autre part les services de transports personnels et les taxis. Il est vrai que les taxis offrent un service plus adéquat et individualisé aux besoins de transport car ils permettent d'avoir une réponse rapide et la possibilité de se déplacer de n'importe quelle origine à n'importe quelle destination. Par contre, le coût de ce service est très élevé et il a un impact environnemental important : il est à ce niveau plus intéressant de transporter simultanément plusieurs passagers en utilisant, si possible, des véhicules de type monospaces ou minibus. D'où l'intérêt de l'utilisation d'un service de TAD.

La Figure 1.3.3 présente le positionnement du TAD par rapport à d'autres modes de transport en terme de flexibilité et d'usage collectif.

Pour résumer on peut dire que le TAD se veut comme étant un système très efficace, réactif à la demande des usagers et permettant de répondre à l'évolution des besoins de mobilité de la population.

Le TAD est un problème qui attire de plus en plus l'attention de chercheurs de différentes disciplines telles que l'informatique, la géographie, la sociologie, ... Á l'inverse du transport cadencé dans lequel la demande s'adapte à l'offre, le TAD vise à adapter l'offre à la demande, et l'optimisation du trajet se fait en fonction des personnes transportées. Il est vrai que dans les années 1970 et 1980, quelques expériences de TAD avaient du mal à fonctionner, faute de moyens techniques adaptés et en raison des modes de conception utilisés. Aujourd'hui, cette conception est remise en cause grâce aux nouvelles technologies de l'in-

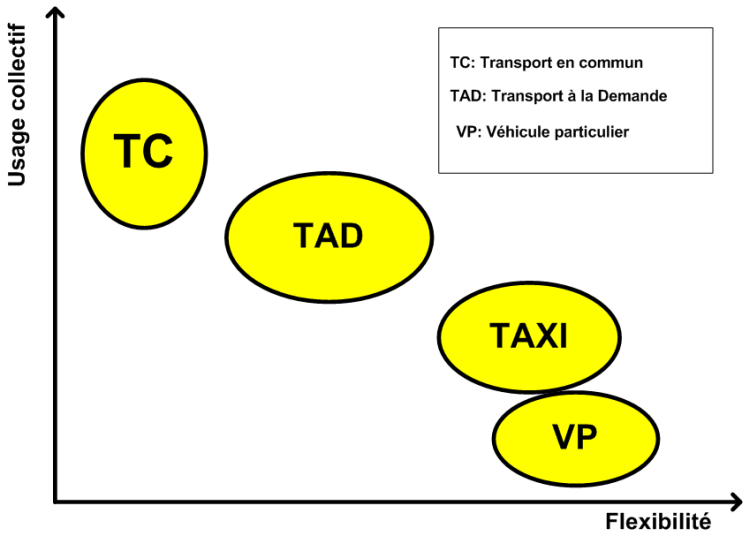


FIGURE 1.1 – Positionnement du TAD

formation et de la communication qui permettent de compenser les handicaps d'autrefois. Les chercheurs tendent à promouvoir de nouveaux modes de conception car le potentiel du TAD est loin d'être totalement exploité. Dans la suite nous présentons quelques conceptions du TAD selon le territoire desservi et la forme des flux de déplacements des passagers.

1.4 Quelle configuration de TAD pour quel territoire ?

La particularité du TAD est d'être souple et flexible. Il n'existe pas qu'une unique conception pour son fonctionnement. Les organisateurs de ce mode de transport gardent une marge de manoeuvre non négligeable, pour choisir ou adapter le service aux besoins spécifiques d'une clientèle ou d'un territoire particulier. Pour cela avant de mettre en place un service de transport à la demande sur une zone géographique donnée, une étude approfondie du territoire est nécessaire pour connaître la dispersion de la population, les directions des flux et la présence des autres modes de transport. Cette étude permettra au décideur de choisir une conception appropriée du mode de desserte.

Castex ([Castex, 2007]) propose une modélisation du territoire sur lequel un service de TAD peut être mis en place. Cette modélisation est nécessaire lors de l'étude de faisabilité qui est souvent réalisée en amont de la création d'un service de transport public. Cette modélisation graphique est initialement destinée à expliquer les organisations et les dynamiques spatiales, qui vont permettre de déterminer les différentes configurations que peuvent prendre les TAD sur une zone géographique donnée. L'objectif de cette modélisation géographique est de faciliter la lecture du territoire et de réduire le nombre de cas de figures pour n'observer que la forme des déplacement.

En comparaison avec des autres modes de transport en commun, le TAD n'a pas une inscription spatiale marquée et sa visibilité dans l'espace est faible voire nulle dans le cas d'un service en porte à porte (l'embarquement et le débarquement des passagers se fait

à domicile), pour plus de détails le lecteur peut se référer à la thèse de Élodie CASTEX ([Castex, 2007]). Les transports en communs par contre sont observables au travers des infrastructures (les voies, les arrêts et les gares) ainsi que de la fréquence de passage des bus. Mais dans le TAD souvent les arrêts, les voies et les fréquences ne sont pas définis. Alors pour représenter le TAD, la seule chose sur laquelle on peut se baser est le flux des déplacements, éventuellement par des arrêts ou des lignes à tracés plus ou moins fixes. Dans une moindre mesure, ils peuvent également être matérialisés par les lieux de destination que sont les villes ou les gares. Les TAD en porte à porte qui n'ont aucune inscription physique, sont néanmoins délimités dans l'espace par l'aire dans laquelle ils circulent, cela veut dire que le service de transport à la demande n'est offert que dans cette zone géographique (contours administratifs d'une commune ou agglomération des communes, département,...) et les véhicules ne peuvent se déplacer qu'au sein d'une zone aux contours bien précis. La Figure 1.4 montre les différentes organisations que peut présenter le TAD au sein d'un territoire.

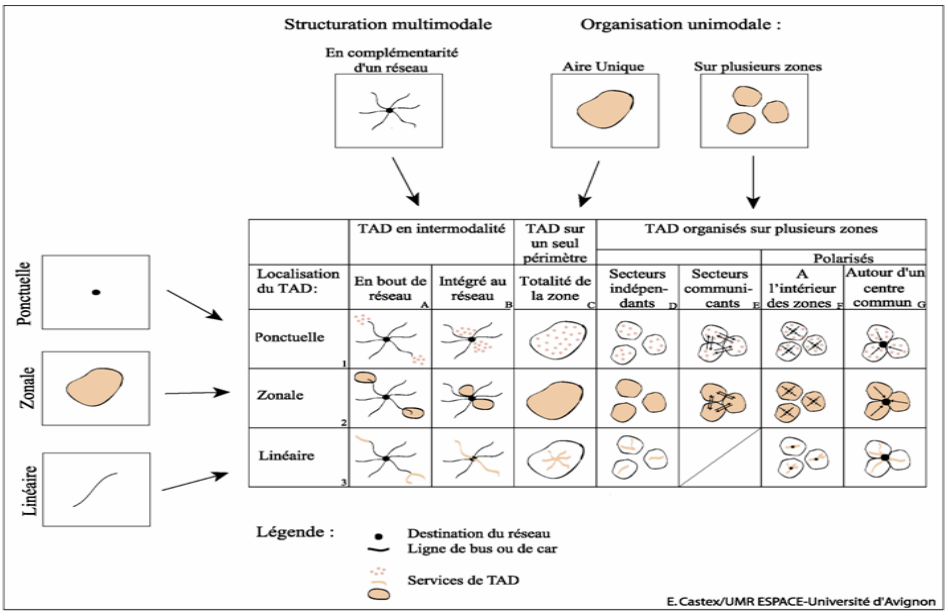


FIGURE 1.2 – Modélisation spatiale du TAD

Pour représenter le TAD les trois formes élémentaires suivantes ont été utilisées : les points (arrêts, lieux de destination), les lignes (lignes virtuelles, réseau) ou les zones (quartier, commune,...). Les lignes orientées (flèches) permettent de rendre compte des trajets des véhicules transportant des passagers. L'ensemble des déplacements réalisés en TAD génère des flux. Ces flux prennent souvent une direction privilégiée, représenté par les flèches. Ainsi, à partir des trois formes élémentaires de l'espace, on peut modéliser selon la direction des flux, l'ensemble des configurations des TAD comme dans la Figure 1.3.

Une fois que la base de fonctionnement du TAD (ponctuelle, zonale ou linéaire) a été déterminée ainsi que sa place au sein du territoire, il reste à définir les modalités de déplacement. Alors la question qui se pose est comment organiser spatialement un TAD dans un territoire donné ? Plusieurs conceptions peuvent être utilisées selon les caractéristiques du

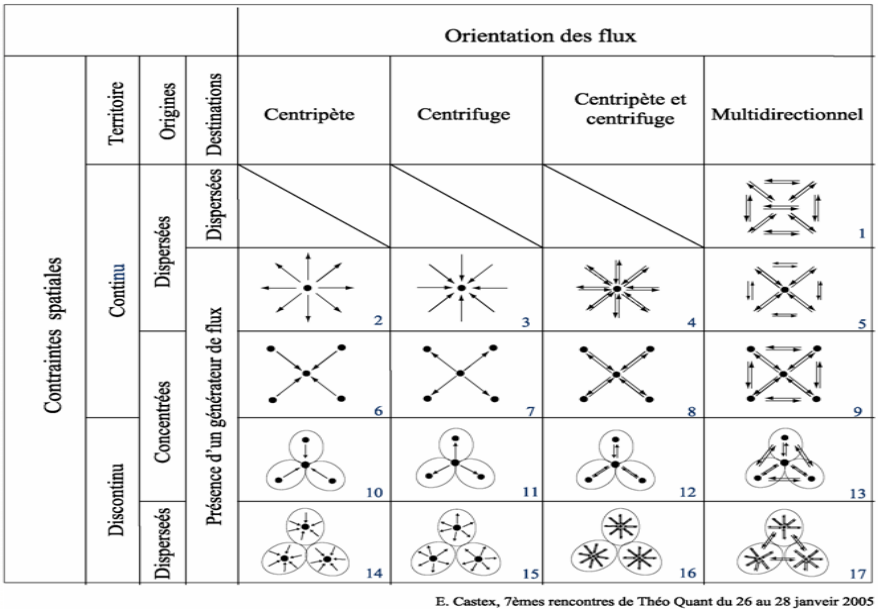


FIGURE 1.3 – L'orientation des flux de déplacement

territoire et de la population à servir. Dans la section suivante nous présentons quelques configurations de TAD, les avantages et les inconvénients de chaque conception.

1.5 Quelques modes de conception du TAD

Le mode de conception d'un TAD dépend essentiellement de la façon par laquelle les passagers seront desservis et si des itinéraires et des points d'arrêts sont prédéfinis ou non. Les points de départ et d'arrivée peuvent être prédéfini ou non, et les horaires de desserte peuvent être imposés ou choisis par les passagers. Ainsi, plusieurs modes de desserte peuvent être considérés selon le territoire et la population concernée. Ci-dessous, nous présentons quelques modes de desserte qui sont utilisés pour mettre en place un service de TAD.

1.5.1 Mode en point-à-point

Ce mode s'appuie sur une base zonale ou ponctuelle : zonale signifie qu'on travaille dans une zone géographique limitée à l'intérieur de laquelle tout point peut être desservi, ponctuelle signifie qu'on travaille avec une liste d'arrêts prédéfinis.

Lorsque les flux sont multi-directionnels dans une zone prédéfinie, le service proposé est celui d'un porte à porte. Dans ce type de TAD les passagers ont la liberté de choisir leurs points d'embarquement et de débarquement dans la zone géographique desservie et les horaires ne sont pas prédéfinis : c'est la forme la plus souple et la plus flexible. Ce système consiste à prendre en charge, en un endroit donné un ou plusieurs passager(s) pour ensuite les déposer à leurs destinations. Un exemple connu de service point à point à la demande mais non partagé est le taxi, mais ce type de transport ne relève pas du TAD.

Tandis que, si dans une zone géographique seulement une liste de points (les points de départ ou d'arrivée) est prédéfinie par le transporteur, les passagers n'ont pas la liberté de choisir leurs points d'embarquement et/ou de débarquement, mais ils sont obligés de joindre les points d'arrêt qui sont prédéfinis. Les arrêts ont été disposés de manière à couvrir tout le territoire et de sorte que tous les lieux soient accessibles. Une forte densité des arrêts fait que le service se rapproche d'un système en porte à porte.

Les systèmes de point-à-point peuvent être mises en place de différentes manières :

- Plusieurs vers plusieurs (many-to-many),
- Plusieurs vers un seul (many-to-one),
- Plusieurs vers quelques'uns (many-to-few),
- Quelques uns vers quelques'uns (few-to-few).

Plusieurs vers plusieurs (many-to-many).
Les usagers sont pris en charge à n'importe quel point de la zone géographique desservie et peuvent être déposés en tout point de cette zone. Ce mode de fonctionnement est très adapté au transport des personnes à mobilité réduite et des personnes âgées, vue sa forte flexibilité. Ce mode de service est très proche du mode de fonctionnement du taxi. Mais le problème de tournées sous-jacent à ce mode de fonctionnement est plus compliqué et nécessite des algorithmes très efficaces afin de réduire les coûts d'exploitation tout en fournissant une meilleure qualité de service aux clients.

Le service Fil Blanc du transport à la demande mise en place dans l'agglomération de Tours (en France) est basé sur une telle approche. Ce service est dédié aux personnes à mobilité réduite et les personnes qui ne peuvent pas emprunter le réseau urbain Fil Bleu (transport cadencé). Il couvre plus de 25 communes (voir la Figure 1.4) et fonctionne toute l'année et tous les jours de la semaine de 7h00 du matin à 1h00 du soir.

Actuellement, la majorité de ces systèmes fonctionne en mode statique (réservation à l'avance) mais aujourd'hui un fonctionnement en mode dynamique est devenu de plus en plus nécessaire, vu les besoins de déplacement qui ne sont pas toujours prévisibles à l'avance. Les développements technologiques rendent maintenant possible la mise en place d'un système de transport à la demande dynamique et plusieurs travaux de recherche ont été consacrés à l'étude de ce type de problème, pour plus de détail le lecteur intéressé trouvera plus de détail dans l'état de l'art publié dans ([Berbeglia *et al.*, 2010b]).

Plusieurs vers un seul (many-to-one).

On parle aussi de « un seul vers plusieurs (one-to-many) ». Dans ce type de système un des deux points de parcours est fixe et l'autre est variable au sein d'une zone géographique prédéfinie. Ce système est intéressant dans le cas de la desserte d'une zone industrielle à l'extérieur de la ville ou pour transporter les passagers vers ou depuis les aéroports, organiser des transport de rabattement vers les gares, les stations de métro ou les arrêts de bus des lignes régulières, etc. Dans une zone non desservie ou faiblement desservie par les services de transport réguliers, les entreprises organisent pour leurs employés des services de

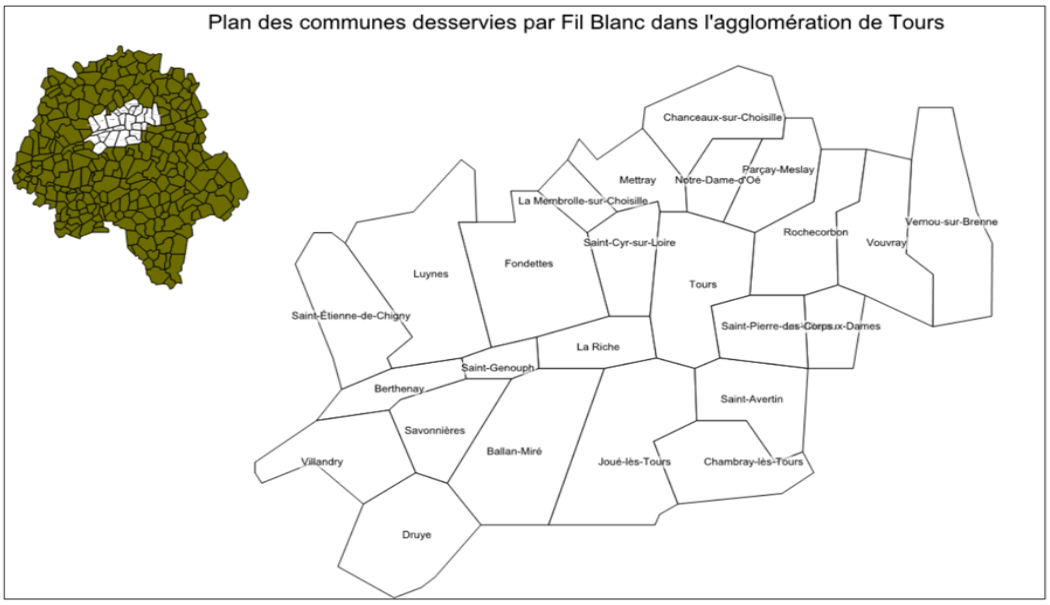


FIGURE 1.4 – Les communes desservies par le service Fil Blanc dans l'agglomération de Tours

transport pour transporter les employés de leur domicile vers le lieu de travail ou l'inverse. Dans ce cas, il est intéressant de mettre en place un service de transport à la demande qui peut répondre aux demandes de déplacement dans ces zones non desservies.
La ville de Besançon a mis en place un service de transport à la demande appelé « Evolis-gare » basé sur un système de « many-to-many ». Ce service a été conçu pour acheminer à la gare de Besançon les voyageurs des TGV du matin, il dessert un ensemble d'arrêts localisés au plus prêt de la population. Le rabattement des usagers se fait sur chaque arrêt Evolis-gare, et globalement vers la gare, qui constitue le début (ou la fin) de la ligne.

Plusieurs vers quelques uns (many-to-few).
C'est le même mode que "Quelques uns vers plusieurs (a few to many)". C'est une extension du mode many-to-one dans lequel les passagers peuvent choisir n'importe quel point de départ dans la zone géographique desservie, mais les points de destinations sont limités et fixés par l'organisateur du TAD. Il s'agit d'une prise en charge à domicile pour des destinations fixes et en général les horaires ne sont pas imposés par le transporteur. Ce type de service est très répandu dans les zones rurales peu dense pour desservir quelques lieux publics comme des hôpitaux, commerces, stations de transports en commun réguliers, etc.

La ville de Tours offre plusieurs services qui se basent sur ce mode de conception : le service « Fil Vert de Nuit » est un service de transport desservant les communes de Montlouis sur Loire, Véretz et Larçay, et qui permet une dépose des usagers à Tours à la gare routière ou au quartier des 2 lions les vendredi et samedi soirs, Le service « Fil Vert à Domicile pour aller à Loches » permet aux habitants de la communauté de communes de Montrésor d'être pris en charge directement à leur domicile vers des points d'arrêts à

Loches.

La ville de Saint-Dié-des Vosges a mis en place un service de transport à la demande basé sur la conception de plusieurs-vers-quelques uns. Le service proposé est dédié à tous le public à condition de faire une réservation au plus tard la veille avant le départ. Les passagers sont servis de leur domicile vers quelques destinations prédéfinies par le transporteur et vice versa.

Quelques uns-vers-quelques uns (few-to-few)
Ce système est souvent mis en place avec des horaires fixes, ce qui le rapproche d'une conception classique de ligne, souvent appelé « ligne virtuelle avec déclenchement ». Cette organisation relativement rigide appartient au coeur du métier des opérateurs de transport, donc à ce qu'ils savent faire et permet de limiter les coûts d'exploitations. Dans ce type de système de transport les points de départ et de destination sont prédéfinis à l'avance par le transporteur : les personnes ne sont pas prises en charge à leur domicile. Ce mode de transport ressemble au transport régulier mais tout ou partie des points d'arrêt n'est desservi que s'il existe une réservation. La réduction du nombre de points d'arrêt diminue la flexibilité du système pour les passagers, mais elle permet une meilleure efficacité du système d'une manière globale et une réduction importante des coûts de transport pour l'organisateur du TAD. Ce type de transport est très adapté dans les milieux ruraux et périurbains qui sont peu denses et il est surtout destiné à un public sans handicap. Ce mode de desserte est largement utilisé par les collectivités locales pour mettre en place un service de transport à la demande. D'une part, parce qu'il est moins cher et d'autre part parce qu'il est facile à gérer et ne nécessite aucune optimisation à faire ni au niveau des tournées ni au niveau des horaires de service des passagers. Certains TAD sont dédiés à ce rôle de lignes virtuelles comme le service TaxiTub mis en place par la ville de Saint-Brieuc comme une substitution au réseau commun à certaines heures ou dans certaines zones difficilement accessibles. Ce service utilise des lignes virtuelles avec itinéraires, arrêts et horaires fixes et il est composé de 57 lignes. Il existe aussi des services spécialement organisés pour les trajets domicile-travail, de service Allobus a Roissy est aussi un service de lignes virtuelles qui permet de compléter les autres services pendant des heures de pointe en offrant des possibilités de déplacement vers les centres urbains en matinée ou l'après-midi. Un autre service de transport à la demande dynamique a été expérimenté sur un ensemble de communes de Montbéliard. Le Modulobus Noêl a permis de se rendre au marché de Noel (2006) à partir de 529 arrêts de bus de la communauté d'agglomération de Montbéliard. La nouveauté est que les personnes peuvent réserver le dimanche, par téléphone ou internet, à tout moment de la journée pour un départ dans l'heure : il n'est pas nécessaire de réserver la veille ni même plusieurs heures à l'avance.

1.5.2 Mode en point de déviation

On trouve particulièrement ce mode de fonctionnement du TAD aux États-Unis. Il consiste à dérouter, à la demande de l'usager, des trajets réguliers de transports en commun. Les premières expérimentations de ce service aux Etas-unis ont consisté à étendre des lignes régulières pour desservir les personnes à mobilité réduite, les personnes âgées ou en général les personnes qui ne peuvent pas rejoindre un arrêt régulier du service depuis leur domicile.

Dans le projet FAMS (Flexible Agency for Collective Demand Responsive Mobility Services) [Ambrosino *et al.*, 2004] ce système de transport est mentionné sous le nom de « deviations on a scheduled service to predefined routes in corridor » tandis que [Burkhardt *et al.*, 1995] l'appellent «point-deviation».

Ce type de service permet de combiner TAD et transport régulier ce qui évite de mettre en place deux modes de transport en commun en parallèle. Ce mode de transport est mis en place entre deux extrémités prédéfinies et quelques arrêts intermédiaires prédéfinis. Généralement l'itinéraire de chaque ligne est semi-fixe, c'est-à-dire que le véhicule est obligé de passer par les arrêts prédéfinis, mais le chemin suivi entre deux arrêts successifs est flexible et s'adapte aux demandes des utilisateurs. Les arrêts obligatoires sur ce type de lignes sont généralement très espacés pour donner une plus grande souplesse au système. Ce type de système peut se voir comme un cas particulier du modèle plusieurs-vers-quelques-uns. [Malucelli *et al.*, 1999] ont étudié un système de transport appelé « Demand Adaptive System », ce système consiste à planifier des lignes de service flexible de transport des personnes composées de plusieurs arrêts prédéfinis, la flexibilité du système consiste à activer des arrêts optionnels pendant des intervalles de temps prédéfinis. La Figure 1.5 montre la structure de ce mode de desserte.

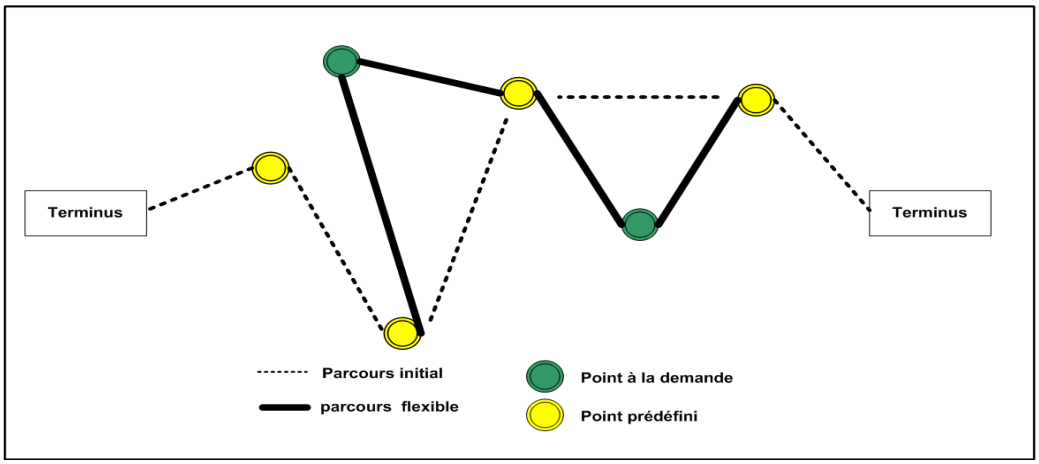


FIGURE 1.5 – Mode point déviation

1.5.3 Mode en route de déviation

Le rapport américain de 1995 ([Burkhardt *et al.*, 1995]) évoque également le cas d'un système nommé «route déviation » . Ce service est proche du mode point de déviation, à la seule différence que le véhicule reprend l'itinéraire au point de déviation. L'objectif ici est d'offrir un service de TAD a trajet semi-fixe : quand le véhicule visite un sommet non prédéfini, il est contraint de rejoindre l'itinéraire dont il a dévié à l'endroit même où il l'a quitté ou il rejoint la ligne prédéfini le plus tôt possible, à la différence du mode point de déviation dans lequel le véhicule rejoint le sommet successeur dans la tournée. Le nombre d'arrêts entre les extrémités de chaque ligne est plus important. L'objectif est d'offrir un

service de transport régulier combiné avec un service de transport à la demande. Le choix de ce mode est toujours basé sur une étude de territoire et sur la nature des déplacements effectués par l'ensemble des passagers dans la zone considérée. Plusieurs travaux de recherche ont été effectuées sur ce mode de fonctionnement du TAD, par exemple le travail de [Quadrifoglio et Dessouky, 2004] dans lequel les auteurs proposent une heuristique d'insertion pour planifier un service de route de déviation appelé « Mobility Allowance Shuttle Transit ». Un ensemble de tests de simulation ont été effectués pour évaluer la performance et l'efficacité de l'algorithme proposé. [Smith *et al.*, 2003] ont défini deux paramètres qui permettent de planifier le service de TAD basé sur la route en déviation : le premier paramètre permet de calculer la zone de service, entre chaque deux points d'arrêt successifs prédéfinis, dans laquelle les demandes de déviation peuvent être servies. Tandis que le deuxième paramètre s'intéresse au calcul de la durée allouée pour servir l'ensemble des demandes de déviation entre chaque deux sommets successifs prédéfinis. Ces deux paramètres ont été utilisés pour simplifier le processus de planification. L'effet de ces deux paramètre à été évalué à travers l'utilisation d'un modèle d'optimisation binaire multiobjectifs. La Figure 1.6 montre un exemple de ce mode de conception.

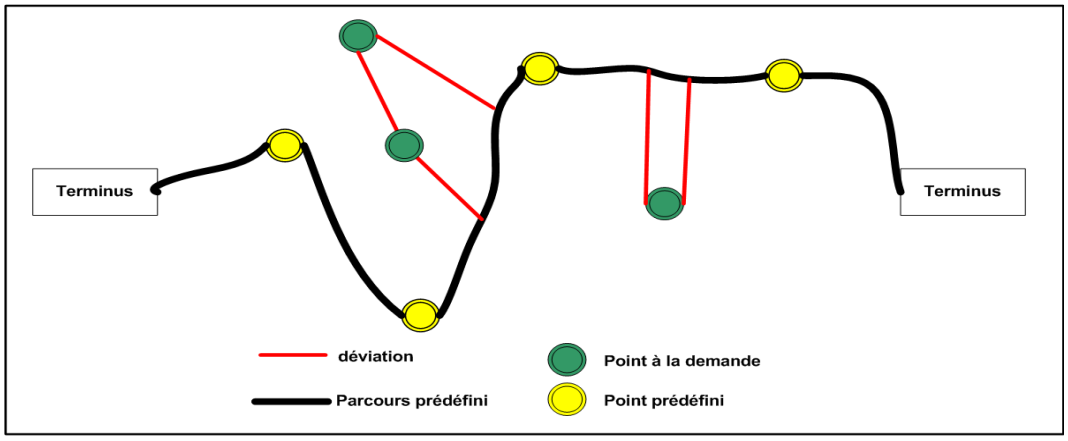


FIGURE 1.6 – Mode route déviation

1.6 Les liens avec le DARP

Le problème général de planification de tournées des véhicules est connu sous le nom de Vehicle Routing Problem (VRP) et représente un problème d'optimisation combinatoire qui a fait l'objet de nombreux travaux de recherche dans la littérature.

Dans sa version de base, le problème de VRP modélise un problème de transport qui consiste à livrer (ou collecter) des produits auprès d'un ensemble de clients à l'aide d'une flotte de véhicules. La résolution consiste à déterminer un ensemble de tournées qui minimise au mieux des objectifs comme la distance totale parcourue, le nombre de véhicules utilisés, la somme des retards des clients. Quand il s'agit du problème de VRP avec des collectes et livraison on parle de PDP (Pickup & Delivery Problem ou Problème de Collecte & Livraison). Un DARP (Dial-a-Ride Problem) est une extension du PDP dans lequel

le transport de marchandises est remplacé par le transport des personnes, d'où certaines contraintes supplémentaires dues au transport des personnes. Par exemple, l'obligation de respecter les délais d'embarquement et de débarquement et des contraintes liées à la qualité de service rendue aux passagers, notamment en minimisant le temps passé à bord du véhicule par chaque passager. Dans sa version de base, le problème de DARP, qui est souvent étudié dans un contexte « many-to-many », consiste à servir un ensemble de requêtes de transport connues à l'avance en utilisant une flotte de véhicules située à un dépôt unique. Chaque requête est caractérisée par un point de départ et d'arrivée, une fenêtre de temps au départ et/ou à l'arrivée et le nombre de personnes à servir.

Le problème de calcul de tournées de véhicules sous-jacent aux problèmes de transport à la demande est connu dans le domaine de la Recherche Opérationnelle sous la dénomination générique de *« Dial-A-Ride Problem »*. Mais le nom DARP englobe aussi d'autres modes de transport de personnes à savoir les taxis et les ambulances qui ne font pas partie du transport à la demande tel qu'il est défini par la législation française. Dans la littérature l'appellation DARP est utilisée aussi pour référencer un problème de transport de patients dans un complexe hospitalier, un problème de tournée du personnel de soins pour l'hospitalisation à domicile, etc. Donc on peut dire que les problèmes de TAD représente un sous ensemble des problèmes de DARP (voir Figure 1.6).

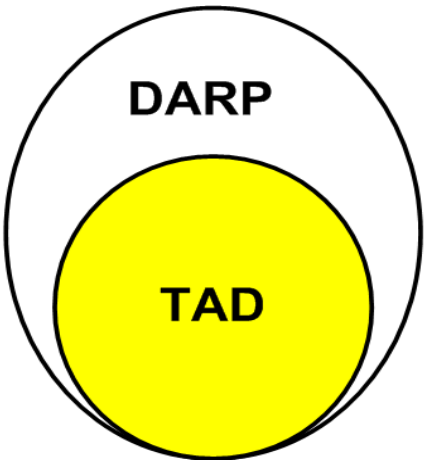


FIGURE 1.7 – Positionnement du TAD par rapport au DARP

Souvent le DARP est abordé comme étant un problème de tournées avec un seul dépôt et rarement sous forme de lignes avec deux extrémités (deux terminus). Les autres modes de conceptions restent marginalement étudiés dans la littérature même si ils sont largement utilisés en pratique, comme le mode en « point de déviation » et « route de déviation » où le TAD est combiné avec d'autres modes de transport (transports multimodaux).

Il faut noter que le service des lignes virtuelles ne peut pas être classé parmi les problèmes de DARP, du moment où le trajet et/ou les horaires de service sont prédéfinis, donc aucun calcul de tournées à effectuer. Le choix d'un mode de desserte dans un transport à la demande a une influence majeure sur la flexibilité du service et le coût de transport. Dans notre étude nous avons choisi de travailler sur une conception « many-to-many » qui reflète le vrai service de transport à la demande avec plus de flexibilité, mais aussi plus de com-

plexité au niveau de la gestion et l'optimisation du service. En général, le DARP incorpore un ensemble de critères, souvent en conflit, qui doivent être optimisés simultanément, d'où une étude multicritère avec approche de Pareto peut s'avérer intéressante dans ce contexte.

Dans le chapitre suivant nous définissons le problème sur lequel nous avons travaillé, nous présentons les données du problème, ses contraintes ainsi que les trois objectifs à optimiser. Et afin de se positionner par rapport à ce qui a été fait dans la littérature, un état de l'art sur les problèmes de DARP statiques et dynamiques sera présenté. Nous introduisons brièvement quelques concepts en relation avec l'optimisation combinatoire ainsi que quelques méthodes de résolution des problèmes multicritères.

Chapitre 2

État de l'art et modélisation du problème

Nous présentons dans ce chapitre un état de l'art sur les problèmes de DARP. Une classification transversale aux problèmes de DARP est basée sur la distinction entre les problèmes statiques, pour lesquels toutes les données sont connues avant de procéder à la planification des tournées, et les problèmes dynamiques où de nouvelles données peuvent apparaitre en cours ou après la planification. Afin de se positionner par rapport aux études qui ont abordé le DARP multicritère, il est intéressant de présenter un état de l'art en classifiant les problèmes selon qu'ils sont monocritères, multicritères avec une agrégation de critères et enfin les problèmes multicritères avec plusieurs critères à optimiser simultanément. Nous mettons l'accent d'avantage sur les problèmes de DARP multicritères et les problèmes dynamiques ainsi que les méthodes de résolution qui ont été utilisées. Avant d'aborder la littérature sur les problèmes de DARP, nous commençons par introduire le problème de PDP (*Pickup and Delivery Problems*) en relation avec le DARP, ensuite, nous définissons l'aspect statique et dynamique ainsi que la différence entre les deux. Par la suite, une revue de la littérature est présentée pour les problèmes de DARP statiques et dynamiques.

2.1 Introduction aux problèmes de tournées de véhicules

Les problèmes de calcul de tournées font depuis plusieurs années l'objet de nombreuses études en Recherche Opérationnelle et certains de ces problèmes d'optimisation combinatoire continuent toujours à être des sujets d'actualité pour de nombreux chercheurs. Dans la littérature, on distingue deux grandes catégories de problèmes de tournées de véhicules : la première catégorie représente l'ensemble des problèmes dans lesquels les tournées se font sur les arcs. Dans cette catégorie de problèmes le but est de servir un ensemble d'arcs (de tronçon) en utilisant une flotte de véhicules, les arcs peuvent être orientés dans un sens ou dans les deux sens. C'est le cas des problèmes de collecte des déchets ménagers, de sablage de routes en cas de verglas ou de déneigement. Pour la deuxième catégorie, ce sont des problèmes dans lesquels les tournées se font sur les noeuds. Dans ce cas le but est de planifier des tournées pour servir un ensemble de points (villes, clients, magasins,...) en utilisant une

flotte de véhicules, plusieurs arcs peuvent être empruntés pour aller d'un point à l'autre. Cette catégorie englobe un ensemble large de problèmes dans plusieurs domaines, comme le problème de distribution de marchandises, de l'hospitalisation à domicile, du transport à la demande, etc.

Les problèmes de tournées sur noeuds se divisent eux même en deux catégories : la première catégorie concerne les problèmes de tournées avec collecte et livraison (embarquement-débarquement ou chargement-déchargement). Ainsi chaque requête ou demande du client est caractérisée par un couple de points : un point de collecte et un point de livraison. Si la collecte et la livraison doivent être effectuées par le même véhicule alors une contrainte appelée « contrainte de couplage » doit être prise en compte lors de la résolution du problème. La deuxième catégorie représente les problèmes de tournées avec visite d'un ensemble de clients où chaque requête est caractérisée par un seul point à visiter (pas de contrainte de couplage).

Le problème de TAD que nous abordons dans cette thèse fait partie de la catégorie des problèmes avec collecte et livraison PDP (*Pickup and Delivery Problems*). Le problème général de « collecte et livraison » (*General Pickup and Delivery Problem*) constitue une classe importante de problèmes de tournées de véhicules. Ce problème a été étudié depuis plus de 30 ans et on le rencontre dans de nombreux domaines à savoir la logistique, la robotique, la gestion de services de mobilité, etc.

Le problème de calcul de tournées de véhicules sous-jacent aux problèmes de TAD est connu sous le nom de *Dial-a-Ride Problem* (DARP). C'est le nom académique le plus utilisé par les chercheurs pour parler du transport à la demande. Le DARP partage plusieurs caractéristiques avec le problème de PDP mais avec la particularité que la qualité de service que nécessitent les passagers devient de plus en plus importante, ce qui rend le problème de DARP de plus en plus compliqué par rapport a un problème de PDP.

Le DARP est structurellement plus proche du PDPTW (*Pickup and Delivery Problem with Time windows*), qui est lui même une extension du VRPTW (*Vehicle Routing Problem with Time Windows*), par conséquent de nombreuses méthodes utilisées dans la littérature pour résoudre le DARP sont en réalité des adaptations des méthodes qui ont été utilisées pour résoudre des problèmes PDPTW ([Berbeglia *et al.*, 2007]). Vu la ressemblance reliant ces deux problèmes, il est intéressant d'avoir des connaissances sur les travaux de recherche qui se font sur les problèmes de PDP et ses variantes qui sont largement étudiées dans la littérature. Des états de l'art plus détaillés peuvent être trouvés dans ([Berbeglia *et al.*, 2007]), ([Parragh *et al.*, 2008]) et ([Parragh et K.F. Doerner, 2008]). La Figure 2.1 montre une classification générale des problèmes de PDP.

La table 2.1 donne un positionnement du DARP par rapport aux autres problèmes de tournées de véhicules selon différentes caractéristiques.

Dans la suite nous présentons rapidement une classification des problèmes de PDP et ces variantes, qui a été proposé par [Berbeglia *et al.*, 2007]. Cette classification se base sur trois caractéristiques pour distinguer les différents problèmes : la structure, les visites, les véhicules.

La première caractéristique, qui est la structure des points de départ et d'arrivée, indique le nombre d'origines et de destinations des biens ou des personnes à transporter. Trois grandes catégories sont envisagées :

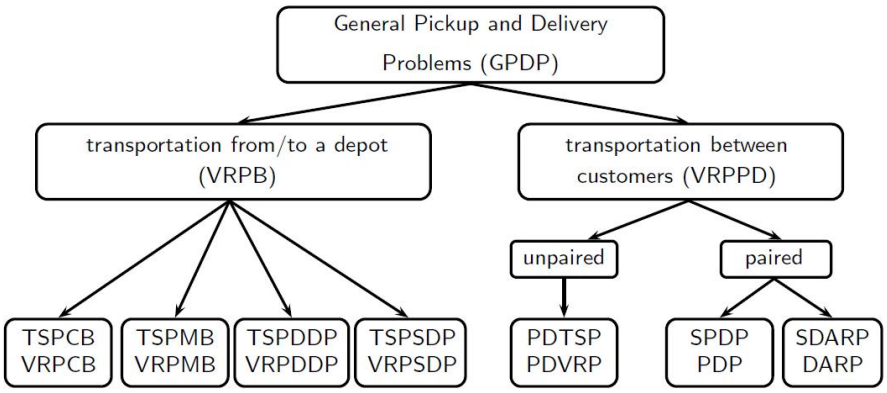


FIGURE 2.1 – Classification des problèmes de PDP ([Parragh, 2009])

Problème	Dépôt	Véhicules	Capacité	Fenêtre de temps	Couplage	Qualité de service	Autres
TSP							
MTSP	*	*					
CVRP	*	*	*				
VRPTW	*	*	*	*			
PDP	*	*	*		*		
PDPTW	*	*	*	*	*		
DARP	*	*	*	*	*	*	*

TABLE 2.1 – Position du DARP par rapport aux autres problèmes de tournées ([Garaix, 2007])

- Many-to-Many (M-M) : chaque sommet peut être utilisé comme un point de départ ou une destination des marchandises (ou des personnes) pour chaque commande (ou requête).
- One-to-many-to-One (1-M-1) : ce cas concerne principalement les biens comme dans le cas de la distribution de bouteilles pleines avec collecte de bouteilles vides. Tous les biens à distribuer sur un ensemble de clients partent d'un unique dépôt. Puis, les biens à collecter chez un ensemble de clients doivent revenir également au même dépôt.
- One-to-One (1-1) : chaque requête est caractérisée par un unique point de départ et d'arrivée. C'est le cas dans le transport de personnes.

La deuxième caractéristique (les visites) est la façon dont la collecte et la livraison sont effectuées à chaque sommet : soit chaque sommet est collecté et livré en même temps et en une seule fois, soit les deux opérations de collecte et livraison sont réalisées séparément, dans ce cas un sommet représente une livraison ou une collecte. Dans certains cas quelques sommets sont considérés comme des point de transfert ou de correspondance ([Masson *et al.*, 2011]).

Enfin, la dernière caractéristique précise le nombre de véhicules à utiliser dans le problème, on distingue entre problèmes avec un seul véhicule, 1-PDP, et problèmes avec plusieurs véhicules homogènes ou hétérogènes, M-PDP. La Figure 2.2 présente un résumé de

la classification proposée dans ([Berbeglia *et al.*, 2007]).

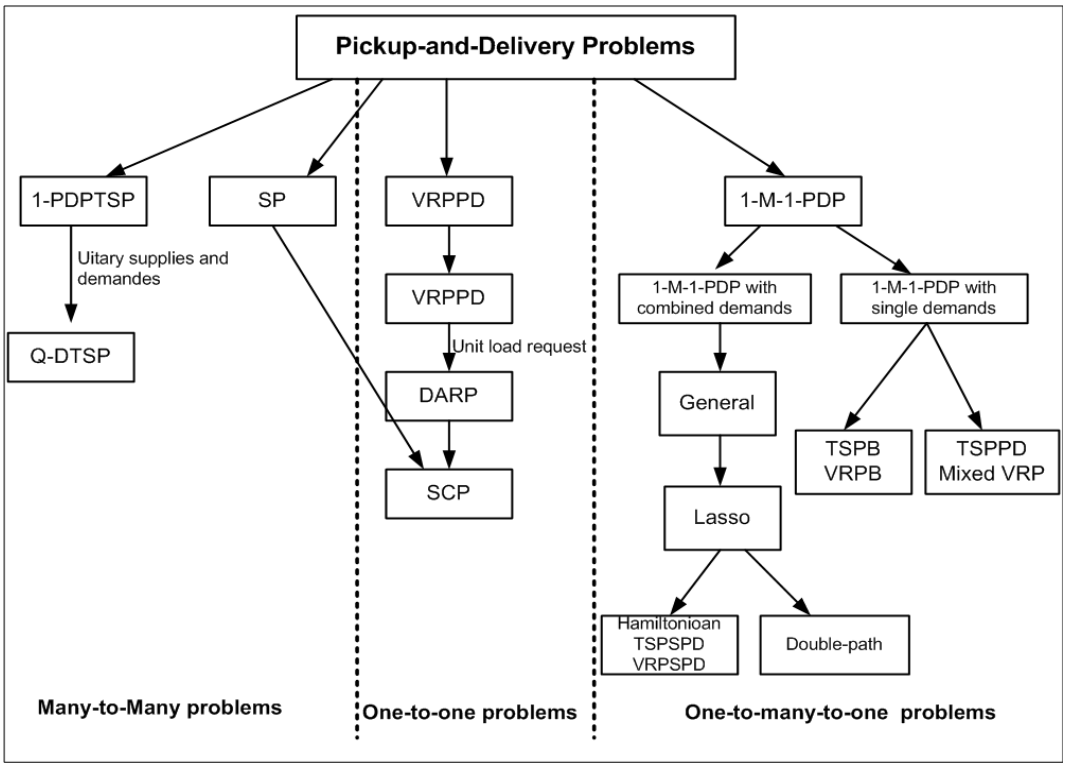


FIGURE 2.2 – Calssification des problèmes de PDP ([Berbeglia *et al.*, 2007])

On peut également distinguer les problèmes de tournées de véhicules selon deux aspects majeurs suivant : l'aspect statique-dynamique et l'aspect déterministe-stochastique. Dans la suite de cette section nous nous attardons sur la distinction des problèmes statiques versus problèmes dynamiques.

2.1.1 L'aspect statique dans les problèmes de tournées de véhicules

Psaraftis ([Psaraftis, 1995]) est l'un des premiers auteurs à avoir défini l'aspect statique dans les problèmes de tournées de véhicules. Il considère que « un problème de tournée est statique si la solution est un ensemble de tournées planifiées à l'avance qui ne sont pas ré-optimisées et qui sont calculées à partir de données d'entrées qui n'évoluent pas en temps réel ». A partir de cette définition on peut dire qu'un problème de tournées de véhicules est statique si toutes les informations nécessaires à la construction des tournées sont connues avant de procéder au processus de planification.

De même ces informations ne doivent pas changer une fois les tournées construites, et il en va de même pour les informations liées aux tournées (affectation des requêtes aux véhicules, affectation des chauffeurs,...) une fois la planification finie.

Un problème de tournée de véhicule statique peut être déterministe ou stochastique. Dans le cas déterministe, toutes les données sont connues entièrement à l'avance. Par contre

dans le cas stochastique, il peut y avoir des incertitudes sur quelques données qui peuvent être modélisées par des variables aléatoires. Par exemple, l'incertitude peut porter sur le temps de service chez le client ou le temps du trajet d'un tronçon routier. Si les données sont incertaines, on ne peut pas garantir la satisfaction des contraintes avec 100 % de réussite. Mais, il est possible que les contraintes soient satisfaites avec une probabilité donnée ou un taux de réussite donné.

2.1.2 L'aspect dynamique dans les problèmes de tournées de véhicules

Un problème de tournées de véhicules est dit dynamique si l'entrée du problème est modifiée de façon non déterministe en fonction du temps. Dans la pratique, cela implique qu'il n'existe pas de solution complète a priori et que la résolution doit être menée en parallèle de l'exécution : à chaque nouvel événement, la solution doit être mise à jour. Dans le cas d'un problème DARP, ce dynamisme consiste en général en l'apparition de nouvelles demandes pendant le déroulement des tournées.

Un problème dynamique peut également être déterministe ou stochastique. Dans le cas déterministe, les données sont connues avec certitude et seuls quelques éléments d'information dépendent du temps ([Larsen et Madsen, 2000]) mais ils sont pas modélisés par des variables aléatoires. Dans le cas stochastique ([Bent et Van Hentenryck, 2003]), une partie des informations du problème n'est pas connue et ces données incertaines sont représentées par des variables aléatoires. Par exemple, l'arrivée des demandes des clients peut suivre une distribution de type loi de Poisson.

L'état de l'art proposé dans ([Bianchi, 2000]) définit clairement la différence entre les problèmes dynamiques qui nécessitent une résolution concurrente à l'exécution et les problèmes dont l'évolution dans le temps est déterministe, ainsi que certains problèmes stochastiques qui peuvent être résolus a priori avec les méthodes probabilistes. Dans le problème de DARP, la prise en compte de certains événements rend le problème dynamique, comme par exemple : l'apparition d'une nouvelle requête, l'annulation d'une requête, l'accident de véhicules, des embouteillages sur les routes ce qui engendre des retards pour des clients pour lesquels des engagements ont été déjà pris, d'où la nécessité de mettre à jour les tournées courantes.

Psaraftis ([Psaraftis, 1995]) a listé un ensemble de caractéristiques principales d'un problème de tournées de véhicules dynamique qui permettent de le différencier d'un problème statique. Ces caractéristiques ont aussi été présentées en détail dans la thèse de Larsen ([Larsen et Madsen, 2000]).

2.2 Revue de la littérature

Le problème de DARP constitue une classe importante des problèmes d'optimisation combinatoire. Ce problème consiste à planifier un ensemble de tournées de véhicules qui vont servir un ensemble de requêtes de transport de personnes tout en respectant l'ensemble des contraintes du problème. Dans la littérature le DARP est souvent résolu comme un problème mono-objectif avec un seul critère à optimiser ou une fonction objectif qui résulte de l'agrégation de plusieurs critères. Notons que dans la littérature les problèmes de DARP

sont parfois abordés sous d'autres nominations comme par exemple :*Demand Responsive Transport*, *Dial-a-Ride Transit*, etc.

Dans cette section nous présentons une revue de la littérature liée aux problèmes de DARP statiques et dynamiques. Nous utilisons une classification selon le type de la fonction objectif utilisée dans chaque travail. Plus précisément, que ce soit pour le DARP statique ou dynamique nous commençons par présenter une revue de la littérature des problèmes mono-objectif. Ensuite, nous présentons les études mono-objectif mais avec une agrégation de plusieurs critères. Enfin, nous présentons les problèmes de DARP abordés dans la littérature avec une approche multiobjectif dans lesquels plusieurs critères en concurrence doivent être optimisés simultanément.

2.2.1 Le DARP statique

2.2.1.1 Panorama des problèmes avec un seul critère

Dans cette section, nous nous intéressons à un état de l'art des problèmes de DARP statique étudiés récemment et qui ne mettent en jeu qu'un seul critère à optimiser. [Borndörfer *et al.*, 1997] ont étudié le cas d'un problème de DARP dédié au transport des personnes handicapées à Berlin. Des contraintes liées au travail des chauffeurs sont prises en comptes, à savoir leur durée maximale de travail, les pauses obligatoires et l'utilisation d'une flotte de véhicules hétérogènes. Le problème est mono-objectif et consiste à minimiser le coût de transport. Une méthode en deux phases « Cluster First Route second » a été utilisée pour résoudre le problème. L'algorithme proposé commence par former des groupes de requêtes, sachant qu'un groupe de requêtes est défini comme étant une sous-tournée maximale dans laquelle le véhicule associé n'est jamais vide. Le début et la fin de chacune des sous-tournées correspondent à l'embarquement du premier passager et au débarquement du dernier passager. Dans la première phase, un ensemble de groupements (appelé *Orders*) sont formés, puis un problème de partitionnement est résolu avec une méthode de Branch-and-Bound pour choisir un sous-ensemble de *clusters* intéressants tout en minimisant la distance totale parcourue. Chaque *cluster* est constitué par un ensemble de groupements. Dans la deuxième phase, les groupes sélectionnés sont ordonnancés pour construire des tournées réalisables en respectant l'ensemble des contraintes du problème. Cette méthode a été appliquée à des instances comprenant entre 859 et 1721 requêtes.

Un problème de DARP au niveau de la ville de Bologne en Italie a été étudié dans ([Toth et Vigo, 1997]). Les passagers spécifient leurs points de départ et d'arrivée avec des fenêtres de temps au départ et à l'arrivée. Le temps passé à bord par passager est limité et il est défini par une fonction linéaire du temps du trajet direct. Le service est assuré par une flotte de minibus et de taxis. L'objectif est de minimiser le coût de transport. Le problème a été résolu par une méthode à deux phases : une phase d'affectation des requêtes aux véhicules effectuée en utilisant une procédure d'insertion parallèle et une phase de post optimisation qui consiste à effectuer des recherches locales en utilisant des échanges intra et inter tournées. Des instances de 276 à 312 requêtes ont été résolues.

Une méthode de recherche Tabou a été proposée dans ([Cordeau *et al.*, 1997]) pour résoudre un problème de DARP statique avec une flotte homogène de véhicules. Les auteurs considèrent le cas du problème dans lequel les passagers spécifient une fenêtre de temps

souhaitée au départ pour les requêtes de retour et une fenêtre de temps souhaitée à l'arrivée pour les requêtes d'aller. La desserte est effectuée par une flotte de véhicules homogènes située dans un dépôt unique. Les tournées construites ne doivent pas dépasser une durée maximale prédéfinie et le temps passé par un passager à bord d'un véhicule est limité. Lors de la recherche des solutions un mécanisme de relaxation est utilisé en cas de violation de certaines contraintes : capacité des véhicules, durée maximale d'une tournée, les fenêtres de temps et la durée maximale qu'un passager peut passer à bord du véhicule. Les violations de ces contraintes sont pénalisées dans la fonction d'évaluation des solutions et les coefficients de relaxation sont ajustés d'une manière automatique et dynamique. Ce mécanisme de relaxation facilite l'exploration de l'espace des solutions : il permet ainsi à la méthode d'évaluer des solutions non réalisables et essayer de construire des solutions réalisables à partir de celles-ci. Une stratégie de diversification est utilisée pour guider le processus de recherche des solutions vers les régions moins explorées dans l'espace des solutions. Des échanges intra-tournées sont effectués à chaque k itérations ou à chaque fois qu'une meilleure solution est trouvée pour optimiser le trajet de chaque tournée. Les auteurs cherchent à minimiser un seul critère qui est le coût total de transport. La méthode a été testée sur 20 instances qui contiennent entre 24 et 144 requêtes générées aléatoirement. La méthode a été testée aussi sur six instances réelles issues de la compagnie *Danish Transporter*.

Cordeau ([Cordeau, 2006]) a ensuite développé une formulation linéaire mixte pour modéliser un problème de DARP statique. Cette première approche de résolution exacte a permis de résoudre des instances avec 32 requêtes et 4 véhicules en utilisant une méthode de Branch-and-Cut. [Ropke et Cordeau, 2009] proposent une modélisation améliorée par rapport à la précédente. Plusieurs nouvelles familles d'inégalités valides au modèle ont été ajoutées pour rendre la formulation du problème plus robuste et ainsi améliorer sa résolution par la même méthode de Branch-and-Cut. Les auteurs ont comparé différentes formulations de DARP et PDPTW. Le modèle a permis de résoudre des instances de 96 requêtes et 8 véhicules, soit des problèmes de tailles deux voire trois fois plus grandes que par le modèle initialement utilisé dans ([Cordeau, 2006]). Notons que les deux problèmes sont mono-objectif et ils consistent à minimiser le coût total de transport.

[Parragh *et al.*, 2010] abordent également le problème de minimisation du coût de transport dans un problème de DARP statique. Les auteurs proposent un algorithme basé sur un voisinage variable en utilisant trois familles de voisinages. Les tests ont été effectués sur 20 instances contenant entre 24 et 144 requêtes et le nombre de véhicules varie entre 3 et 13. Les résultats obtenus ont été comparés avec ceux obtenus par la recherche Tabou proposée dans [Cordeau et Laporte, 2003] et seulement une légère amélioration a été obtenue. Afin de se comparer avec les résultats obtenus dans ([Jorgensen *et al.*, 2007]). Parragh et al. ont remplacé la fonction objectif avec la même fonction utilisée dans ([Jorgensen *et al.*, 2007]) et ils ont utilisé la même méthode de recherche locale. Les résultats obtenus sont largement meilleurs sur les différents critères testés.

[Parragh, 2011] a publié aussi une étude d'un problème de DARP destiné au transport de patients en utilisant des véhicules hétérogènes (2 types) de différents capacités et en considérant des usagers diversifiés. Parragh distingue entre quatre types de patients : des patients qui nécessitent seulement une place assise, un fauteuil roulant, une civière ou un accompagnant. Pour chaque requête une fenêtre de temps est associée soit au sommet

de départ soit au sommet d'arrivée. L'objectif est de minimiser un seul critère qui est le coût total de transport en respectant ces intervalles de temps et les durées maximales des tournées. Les périodes de repas et de repos pour les chauffeurs sont prises en compte lors de la résolution du problème. Le modèle mathématique proposé dans ([Cordeau, 2006]) et qui a été amélioré dans ([Ropke et Cordeau, 2009]) a été modifié pour prendre en considération l'hétérogénéité des passagers et des véhicules. La méthode de Branch-and-Cut est utilisée pour résoudre des instances de moins de 40 requêtes. Parragh a adapté ensuite l'heuristique proposée dans ([Parragh *et al.*, 2010]) pour résoudre ce nouveau problème. Cette métaheuristique a permis de trouver des résultats de bonne qualité.

2.2.1.2 Panorama des problèmes multicritères avec une fonction d'agrégation

En 1980, [Psaraftis, 1980] a examiné le cas d'un problème de DARP dans lequel les clients ne spécifient pas de fenêtres de temps, mais souhaitent seulement être desservis aussi tôt que possible. Ce problème a été résolu de manière exacte par une méthode de programmation dynamique. L'objectif est de minimiser une combinaison linéaire du temps total de service et de l'insatisfaction des clients. Le niveau d'insatisfaction est calculé comme étant la somme pondérée du temps d'attente avant l'embarquement et du temps passé à bord du véhicule avant d'arriver à la destination. Ensuite, le même algorithme a été adapté pour résoudre le même problème mais avec des fenêtres de temps au départ et à l'arrivée ([Psaraftis, 1983]). L'algorithme proposé ne permet de résoudre d'une façon optimale que des instances de petite taille ($\leq$10) et la complexité de la méthode est en $O(n^2 3^n)$ où n est le nombre de requêtes.

[Sexton et Bodin, 1985] ont proposé une heuristique pour résoudre un problème similaire à celui étudié dans ([Psaraftis, 1983]), dans lequel les passagers spécifient une date souhaitée de débarquement pour laquelle le retard n'est pas souhaité. L'insatisfaction des passagers est mesurée par la somme de deux termes : le premier terme est le temps passé à bord du véhicule (la durée du trajet effectuée moins la durée du trajet direct). Le deuxième terme calcule pour chaque requête la différence entre l'heure à laquelle le débarquement est effectué moins l'heure souhaitée. La méthode proposée est basée sur la décomposition de Benders pour le DARP avec un seul véhicule, le problème esclave étant le problème d'ordonnancement des sommets avec minimisation de l'insatisfaction des clients.

Le problème de DARP avec un seul véhicule a été aussi reformulé en PLNE dans ([Desrosiers *et al.*, 1986]) en incluant les fenêtres de temps et la capacité du véhicule. Le problème est résolu en utilisant la programmation dynamique avec un schéma d'étiquetage double. L'algorithme a été appliqué pour résoudre des instances de taille inférieure à 40 passagers.

Depuis la fin des années 1980, la majorité des travaux de recherche ont été consacrés au DARP statique multi-véhicules, multicritères avec une fonction d'agrégation. L'heuristique d'insertion proposée dans ([Jaw *et al.*, 1986]) est l'une des premières méthodes qui ont été proposées pour résoudre un problème de DARP statique multi-véhicules . Cette heuristique a été reprise et adaptée dans plusieurs autres travaux sur le DARP (voir [Madsen *et al.*, 1995]). Les auteurs abordent un problème dans lequel les fenêtres de temps sont soient associées aux points de départ pour les requêtes Aller ou associées aux points de destination dans le cas des requêtes Retour. Une contrainte de durée maximale de trajet

est imposée pour chaque requête et cette durée est définie proportionnellement au temps de trajet direct. Les véhicules ne sont pas autorisés à attendre quand les passagers sont à bord. Une fonction objective flexible qui équilibre la minimisation du coût de la prestation de service avec les préférences des clients en matière de la qualité de service. L'heuristique sélectionne les requêtes dans l'ordre au plus tôt des dates d'embarquement, et graduellement les requêtes sont insérées dans la tournée. L'algorithme a été testé sur des instances réelles avec 2617 requêtes et 28 véhicules. Une technique assez proche de celle proposée dans ([Jaw *et al.*, 1986]) est utilisée dans ([Madsen *et al.*, 1995]) pour résoudre un DARP statique avec fenêtre de temps, multi-véhicules et multiobjectif. Le même algorithme est adapté pour résoudre le problème dans un contexte dynamique. Quatre critères à minimiser ont été pris en compte lors de la résolution du problème : la durée totale des tournées, le nombre de véhicules, le temps d'attente, la déviation par rapport à l'heure de service promise et le coût total de transport. Il est claire que ces quatre critères sont partiellement conflictuels. L'approche multiobjectif adoptée par les auteurs pour résoudre ce problème est celle de la somme pondérée des quatre fonctions objectif. Les auteurs ont expliqué en détail la manière avec laquelle les quatre pondérations sont calculées et leur signification. La méthode a été testée sur des instances réelles de la compagnie danoise « *Copenhagen Fire-Fighting Service* (CFFS) », qui offre le service de transport à la demande dans la ville de Copenhagen pour les personnes handicapées ou âgées. Bien que la méthode proposée permette de résoudre des instances de grandes tailles en quelques secondes, les auteurs ne présentent aucune comparaison sur la qualité des résultats obtenus avec d'autres méthodes de la littérature. Étant donnée la rapidité de la méthode proposée, les auteurs ont réutilisé le même algorithme pour résoudre le même problème dans un contexte dynamique après la mise à jour du problème lors de l'apparition de nouvelles requêtes.

La dimension environnementale est prise en compte pour la première fois lors de la résolution du problème de DARP par ([Dessouky *et al.*, 2003]). L'un des objectifs des auteurs est de prendre en compte les trois critères conflictuels suivant : minimisation du coût de transport, maximisation de la qualité de service et minimisation de l'impact environnemental. Ces trois critères sont agrégés dans une seule fonction objectif. Les auteurs se sont basés sur la méthode proposée dans ([Jaw *et al.*, 1986]) pour résoudre leur problème. Des expérimentations intensives ont été effectuées sur des instances de taille inférieure à 10000 requêtes et 210 véhicules. Les résultats obtenus montrent clairement qu'avec une légère augmentation du coût de transport et des retards non importants au niveau des horaires de service des passagers, on peut réaliser une amélioration significative en terme d'impact environnemental.

[Jorgensen *et al.*, 2007] ont développé une heuristique en deux phases pour résoudre un problème de DARP statique. Dans la première phase, un algorithme génétique est utilisé pour regrouper les clients dans les différentes tournées. Par la suite, une heuristique de recherche locale est utilisée pour déterminer l'ordre de visite des sommets (les points de départ et d'arrivée des requêtes). Les auteurs cherchent à minimiser la somme pondérée de sept termes différents dont la majorité représentent la qualité de service rendue aux passagers. L'algorithme a été testé et comparé sur les instances de Cordeau et Laporte ([Cordeau et Laporte, 2003]). Les résultats obtenus montrent qu'avec l'algorithme génétique les solutions sont meilleures au niveau du temps d'attente et du temps passé à bord des véhicules par les passagers, mais elles sont moins bonnes au niveau du coût de transport

et de la durée des tournées, ceci peut s'expliquer par le fait que la qualité de service n'est pas prise en compte dans la fonction objectif du problème traité par Cordeau et Laporte.

[Cubillos *et al.*, 2009] proposent l'utilisation d'un algorithme génétique pour résoudre un problème de DARP statique. Le problème étudié considère que la flotte des véhicules est homogène et que les passagers peuvent spécifier une fenêtre de temps au départ et à l'arrivée. La fonction d'évaluation des solutions est une somme pondérée des critères suivants : la durée d'une tournée, le nombre de véhicules, le temps passé à bord et le temps d'attente des véhicules. Les instances proposées dans ([Cordeau et Laporte, 2003]) ont été utilisées pour tester la méthode. Les résultats obtenus sont meilleurs par rapport à ceux obtenus dans ([Jorgensen *et al.*, 2007]) sur l'ensemble des critères considérés. Tandis que, par rapport aux résultats obtenus par la recherche Tabou ([Cordeau et Laporte, 2003]), ces résultats sont moins satisfaisants au niveau de la durée des tournées. Ils sont néanmoins meilleurs au niveau du temps passé à bord des véhicules par l'ensemble des passagers et de la durée d'exécution de l'algorithme.

[Mauri et Lorena, 2006] proposent une modélisation d'un problème particulier de DARP statique multiobjectif. Le problème étudié considère plusieurs dépôts, une flotte de véhicules hétérogènes et des fenêtres de temps associées aux points de départ et d'arrivée des requêtes. L'objectif est de minimiser la somme pondérée de plusieurs critères (combinaison linéaire de 10 critères). Une méthode de recherche locale, de type Simulated Annealing, est utilisée pour résoudre le problème. La méthode a été testée sur 20 instances de Cordeau et Laporte ([Cordeau et Laporte, 2003]) et les résultats obtenus sont comparés avec ceux obtenus par la recherche Tabou proposée par les mêmes auteurs ([Cordeau et Laporte, 2003]). Les solutions obtenues ont été comparées au niveau des critères suivants : la distance totale, la durée des tournées, le temps d'attente et le temps passé à bord par l'ensemble des passagers. Les résultats montrent que la méthode de « Simulated Annealing » ne permet pas de trouver des solutions de distance totale meilleure que la méthode de recherche Tabou, tandis qu'elle arrive à faire mieux au niveau des autres critères et avec un temps d'exécution beaucoup plus petit.

[Diana et Dessouky, 2004] proposent une méthode d'insertion basée sur un "regret parallèle" pour résoudre un problème de DARP statique de grande échelle, ils minimisent une combinaison linéaire de trois termes qui représentent trois points de vues différents : minimiser la distance totale parcourue par l'ensemble des véhicules, minimiser la violation de la contrainte de temps passé à bord par l'ensemble des passagers et minimiser la durée totale d'attente des véhicules. La méthode proposée à une complexité en $O(n^3)$ et elle a été testée sur des instances réelles issues de *Los Angeles County*, contenant entre 500 et 1000 requêtes. Les résultats obtenus montrent que la méthode permet d'obtenir de bons résultats notamment pour les instances de grande taille.

[Xiang *et al.*, 2006] ont résolu un problème de DARP statique avec des contraintes complexes dans lequel les véhicules sont hétérogènes, la durée maximale de travail et les pauses obligatoires doivent être respectées et chaque chauffeur ne peut conduire qu'un certain type de véhicules. Les requêtes sont hétérogènes, c'est-à-dire qu'il y a des requêtes qui ne peuvent être servies que par certains type de véhicules. Les contraintes classiques du DARP ont été prises en compte comme la durée maximale que peuvent passer les passagers à bord des véhicules, le respect des fenêtres de temps aux points de départ des requêtes et la capa-

cité des véhicules. La fonction objectif minimisée est une somme pondérée des coûts fixes des véhicules, des coûts d'utilisation des véhicules, des coûts liés aux chauffeurs, du temps d'attente et du temps de service. Une heuristique basée sur une recherche locale est utilisée pour résoudre ce problème. Les auteurs ont introduit un mécanisme de diversification lors de la recherche des solutions en utilisant une fonction objectif secondaire pour minimiser le temps d'attente des véhicules. La méthode a été testée sur des instances de taille variant entre 50 et 2000 requêtes et le temps d'exécution est environ de 10 heures pour une instance de 2000 requêtes. Enfin, [Wolfer Calvo et Colorni, 2007] proposent une heuristique rapide pour résoudre un DARP statique. Cette heuristique se base sur un graphe auxiliaire dans lequel chaque noeud représente une paire de noeuds d'une requête et le graphe est valué avec un coût qui représente la proximité entre deux passagers (requêtes) en termes de distance géographique et temporelle. Le graphe auxiliaire est utilisé pour résoudre le problème d'affectation, qui sera suivi d'une phase d'ordonnancement des sommets pour construire des tournées réalisables. La méthode tente d'optimiser deux critères d'une manière hiérarchique : le premier critère consiste à maximiser le nombre de passagers desservis et le deuxième critère correspond au niveau de la qualité de service rendue à l'ensemble des passagers desservis. Ceci se traduit par la minimisation de la somme des temps d'attente et du dépassement du temps maximum passé à bord des véhicules par les usagers. La méthode a été testée sur des instances contenant entre 10 et 180 requêtes provenant du réseau routier de la ville de Milan en Italie.

2.2.1.3 Panorama des problèmes multicritères

Dans la majorité des articles que nous avons cité précédemment, le problème de DARP est considéré par les auteurs comme un problème multicritère, mais souvent ces critères sont agrégés dans une seule fonction objectif pour résoudre en fin de compte un problème monocritère (par exemple [Dessouky *et al.*, 2003], [Jorgensen *et al.*, 2007], [Madsen *et al.*, 1995] parmi d'autres). Les méthodes utilisées pour résoudre ce genre de problèmes ne permettent que de générer une seule solution. Par ailleurs, plusieurs méthodes d'optimisation multicritère peuvent être utilisées pour générer un ensemble de solutions non-dominées, ce qui permet aux décideurs de mieux comprendre les arbitrages à faire entre les différents critères.

A notre connaissance, les travaux dans lesquels le DARP statique est traité comme un problème multicritère (avec, par exemple, une approche où on cherche explicitement des optima de Pareto, cf Section 2.3) sont limités et peu de chercheurs se sont intéressés à la résolution de cette catégorie de problèmes. La revue de la littérature nous a permis d'identifier quatre travaux qui ont été publiés récemment, dont deux sont bicritères.

[Baugh *et al.*, 1998] sont parmi les premiers auteurs qui se sont intéressés à l'étude d'un problème de DARP multicritère. Ils ont développé une heuristique de *Simulated Annealing* pour résoudre un problème de DARP statique avec deux critères. Ils s'intéressent à la recherche des solutions de meilleures compromis entre deux critères : le premier critère est la minimisation du coût de transport, tandis que le second critère consiste à minimiser les violations des fenêtres de temps ce qui reflète la qualité de service rendue aux passagers. Ces deux critères sont indépendants un de l'autre (antagonistes) d'où l'intérêt d'utiliser une approche multiobjectif pour fournir un ensemble de solutions au décideur afin qu'il

puisse faire un arbitrage entre les deux critères. Les auteurs ont testé leur approche sur des instances réelles comportant jusqu'à 300 requêtes. Aucun indicateur de qualité n'a été utilisé pour évaluer la qualité des solutions qui ont été générées.

[Parragh *et al.*, 2009] ont développé une heuristique en deux phases pour résoudre un DARP statique bicritère. Une méthode basée sur une recherche à voisinage variable (*Variable Neighborhood Search*) couplé avec une méthode de *Path-Relinking* est utilisée pour identifier les solutions non-dominées en minimisant simultanément le coût de transport et le temps moyen passé à bord des véhicules par l'ensemble des passagers. La méthode a été testée sur des instances de moins de 96 requêtes et trois indicateurs ont été utilisés pour évaluer la qualité des solutions générées par la méthode proposée : un indicateur d'hypervolume, un indicateur *Epsilon* et un indicateur de préférences du décideur.

[Chevrier *et al.*, 2010] ont étudié un problème de DARP statique multiobjectif. Les trois critères que les auteurs cherchent à optimiser sont : la minimisation du nombre de véhicules requis, la minimisation de la durée des tournées et la minimisation des retards. Pour résoudre ce problème trois méthodes évolutionnaires multiobjectif ont été utilisées : NSGA-II, IBEA et SPEA2. Ces méthodes ont été testées et comparées sur des instances générées aléatoirement et d'autres instances réelles générées en utilisant les coordonnées GPS et les systèmes d'informations géographiques (SIG) pour calculer la matrice des distances. Ces instances ont été utilisées en utiliant le framework ParadisEO-MOEO (*http ://paradiseo.gforge.inria.fr/index.php*). Les résultats obtenus montrent que NSGA-II est souvent meilleur (ou au moins équivalent) à IBEA et SPEA2, notamment sur des instances aléatoires. Tandis que les résultats obtenus sur des instances réelles montrent que IBEA est meilleur que NSGA-II et SPEA2. L'indicateur d'$Hypervolume I_H$ et l'indicateur $\epsilon - Indicator$ sont utilisés pour comparer les front de Pareto générés par les différentes méthodes.

Enfin, dans le cadre de sa thèse de doctorat, Paquette ([Paquette *et al.*, 2011]) a étudié un problème de DARP multicritère, dans lequel l'accent est d'avantage mis sur la définition de la qualité de service dans un transport à la demande. L'algorithme de recherche Tabou proposé à la base pour résoudre un problème de DARP statique monocritère par Cordeau et Laporte ([Cordeau et Laporte, 2003]) a été adapté pour résoudre le problème dans un contexte multicritère. En plus du coût de transport, trois autres critères de qualité de services (QoS) ont été pris en compte : le premier critère de QoS « Quality1 » est la somme quadratique des temps d'attente aux sommets de destination des requêtes Aller, le deuxième critère de QoS « Quality2 » est la somme quadratique des temps d'attente aux sommets de départ des requêtes Retour et le dernier critère de QoS « Quality3 » est la somme quadratique du rapport entre le temps réel passé à bord par chaque passager pour arriver a sa destination et le temps du trajet direct. L'aspect conflictuel entre ces trois critères de qualité de service n'est pas clair. La méthode utilisée adapte dynamiquement les poids de la combinaison linéaire des quatre objectifs et les solutions non-dominées correspondantes sont stockées dans une mémoire de type archive. Des tests ont été fait sur des instances aléatoires et des instances issues d'un opérateur de transport à Montréal au Canada.

2.2.2 Le DARP dynamique

Si les problèmes de DARP statiques sont largement étudiés dans la littérature, les problèmes de DARP dynamiques n'ont pas encore reçu autant d'attention de la part des chercheurs. Récemment, [Berbeglia *et al.*, 2010a] ont publié une revue de la littérature sur les problèmes de collecte et livraison dynamiques qui contient une partie importante dédiée aux problèmes de DARP dynamiques. Dans cette section nous utilisons la même classification que nous avons utilisé dans la section 2.2.1 pour les problèmes de DARP statiques : ainsi les problèmes seront classés selon le type de la fonction objectif. La revue de la littérature sur les problèmes statiques montre une tendance dans l'utilisation des métaheuristiques qui nécessite un temps d'exécution assez grand. Pour cette raison, la majorité de ces méthodes n'a pas été réadaptée pour résoudre la version dynamique du même problème.

2.2.2.1 Panorama des problèmes avec un seul critère

Les premières études du problème de DARP dynamique ont consisté à traiter des problèmes avec un seul véhicule. [Psaraftis, 1980] a utilisé la programmation dynamique pour résoudre la version dynamique d'un problème de DARP avec un seul véhicule de capacité supérieure à 2. L'objectif consiste à minimiser le degré de satisfaction des passagers exprimé comme une fonction linéaire du temps d'attente des clients et du temps passé à bord du véhicule jusqu'à l'arrivée au lieu de destination. La taille maximale des instances testées est de 10 passagers.

En 1983, Psaraftis a modifié l'algorithme exact de programmation dynamique utilisé précédemment pour résoudre le même problème avec les fenêtres de temps au départ et à l'arrivée. L'objectif cette fois-ci est de minimiser le temps total nécessaire pour servir tous les passagers. Cet algorithme exact a une complexité en $O(n^2 3^n)$, où n est le nombre de requêtes ([Psaraftis, 1983]).

[Colorni et Righini, 2001] ont étudié un problème de DARP dynamique, proche du Transport à la Demande, de deux villes italiennes. Afin de maximiser le nombre de requêtes acceptées, les auteurs proposent de négocier avec les passagers lors de la réservation : les fenêtres de temps souhaitées, l'origine et la destination de desserte. Le seul objectif retenu est la qualité de service calculée comme étant le maximum des déviations des temps de transport par rapport aux temps direct. La méthode utilisée pour résoudre le problème fait partie des stratégies à deux phases. Dans la première phase, les auteurs utilisent une recherche locale pour résoudre le sous problème d'allocation des requêtes aux véhicules. Dans la seconde phase, les auteurs utilisent une méthode de séparation et évaluation pour déterminer l'ordre de visite des passagers dans chaque tournée. Etant donnée la complexité du problème, les auteurs proposent de réduire l'horizon de planification et d'exécuter la méthode sur des horizons roulants afin de réduire la taille du problème en s'intéressant à chaque fois aux requêtes urgentes. Cette méthode a été intégrée dans un logiciel (DARIA) qui a été testé sur des instances provenant des villes italiennes de Crema et Verbania.

[Horn, 2002] a développé un outil logiciel pour l'optimisation du Transport à la Demande dans le cas statique et dynamique. Dans le cas du problème DARP dynamique les nouvelles requêtes qui arrivent en temps réel sont insérées dans les tournées existantes en

minimisant le sur-coût engendré par l'insertion. Une recherche locale est exécutée périodiquement pour améliorer la solution tout en essayant d'anticiper les informations futures sur les requêtes qui peuvent apparaitre le long de la journée. La méthode utilise aussi une heuristique d'affectation des véhicules pour un déploiement efficace des véhicules et une meilleure qualité de service pour les utilisateurs. Le logiciel a été testé sur des instances comportant jusqu'à 4200 requêtes par jour et 220 taxis disponibles.

[Attanasio *et al.*, 2004] étudient un problème de DARP dynamique dans lequel les passagers spécifient une fenêtre de temps au point de départ s'il s'agit d'un Aller, ou une fenêtre d'arrivée s'il s'agit d'un Retour. Un algorithme parallèle de recherche Tabou, basé sur la méthode proposée dans ([Cordeau et Laporte, 2003]), est utilisé pour résoudre le problème. L'utilisation de l'algorithme Tabou permet de générer des solutions de haute qualité dans le cas du DARP statique, mais avec un temps d'exécution significatif, ce qui justifie l'intérêt du calcul parallèle. Dans cet article, un certains nombre d'implémentations parallèles de la recherche Tabou ont été présentées. Les contraintes opérationnelles du problème de DARP statique sont les mêmes pour le DARP dynamique, mais la fonction objectif dans la version dynamique contient une composante qui permet de maximiser le nombre de demandes acceptées en temps réel.

L'algorithme parallèle de recherche Tabou fonctionne de la manière suivante : la solution statique est construite en se basant sur les demandes connues au début de la planification des tournées en utilisant l'algorithme Tabou avec un seul processeur. Quand une nouvelle requête apparait par téléphone ou par Internet le client doit recevoir la réponse dans une durée qui ne dépasse pas l'ordre de quelques secondes, pour cela l'algorithme effectue une vérification en parallèle pour savoir s'il existe une solution réalisable, c'est-à-dire plusieurs processus sont lancés en parallèle : si au moins une solution réalisable est trouvée par un des processus, on arrête l'exécution et la requête est acceptée. Si au bout d'un moment aucune solution réalisable n'a été trouvée, la requête sera rejetée.

Dans le cas de l'acceptation de la requête, l'algorithme effectue une post optimisation des tournées, pour améliorer la solution actuelle. Plusieurs stratégies de parallélisme de l'algorithme Tabou ont été testées. Des instances de taille inférieure à 144 requêtes ont été résolues.

[Coslovich *et al.*, 2006] ont proposé une heuristique d'insertion en deux phases pour résoudre un problème particulier de DARP dynamique sans contrainte de capacité et avec la possibilité qu'un passager puisse faire une demande de transport à un arrêt de véhicule directement au chauffeur : ce dernier est capable de décider s'il peut accepter la requête ou non. Dans la première phase, de l'heuristique les auteurs utilisent une procédure simple et rapide d'insertion pour tenter de trouver une solution réalisable. Dans la deuxième phase la solution est améliorée par une recherche locale en utilisant l'opérateur de 2-OPT. La fonction objectif utilisée est la minimisation de la non satisfaction des passagers : ce critère est mesuré en fonction des déviations par rapport à la date de service souhaitée par chaque passager et le dépassement du temps maximum que peut passer chaque passager à bord du véhicule. L'algorithme a été testé sur plusieurs scénariis d'instances générées aléatoirement et comportant jusqu'à 50 requêtes.

Enfin, [**?**] proposent un algorithme hybride pour résoudre un problème de DARP dynamique. Cet algorithme s'inspire de la méthode de recherche Tabou développée dans

[Cordeau et Laporte, 2003], combinée avec un algorithme de programmation par contraintes. Quand une nouvelle requête apparaît, la recherche taboue tente de l'insérer dans l'une des tournées existantes. En parallèle, la programmation par contraintes tente aussi de trouver une solution réalisable au nouveau problème (requêtes statiques + la nouvelle requête) ou de confirmer que le problème n'est pas réalisable. Dans leur article, les auteurs proposent trois méthodes pour l'optimisation des horaires de desserte pour chaque passager afin d'augmenter le nombre de requêtes desservies.

Pour évaluer la performance la méthode qu'ils ont proposé, les auteurs ont mené une série de tests sur des instances de Cordeau et Laporte ([Cordeau et Laporte, 2003]) et les instances proposées dans ([Ropke et Cordeau, 2009]) : la taille de ces instances ne dépasse pas 144 requêtes.

2.2.2.2 Panorama des problèmes avec agrégation de plusieurs critères

[Madsen *et al.*, 1995] réutilisent une méthode initialement dédiée à la résolution d'un problème de DARP statique pour résoudre un problème dynamique. Le problème est résolu avec la méthode appelée "*Rolling Time Horizon* (RTH)" qui consiste à re-exécuté périodiquement le même algorithme utilisé pour résoudre le problème statique, en prenant en considération la, ou les nouvelles, requêtes qui apparaissent au fil du temps. L'heuristique développée à l'origine pour résoudre le problème statique est très rapide (de l'ordre de quelques secondes). Le travail effectué par Madsen et al. s'inscrit dans le cadre d'un projet qui a été initié par le CFFS (*Copenhagen Fire-Fighting Service*) pour répondre à environ 50000 demandes annuelles de transport de personnes âgées et handicapées. Dans ce problème, les passagers peuvent spécifier une fenêtre de temps de départ ou d'arrivée et la flotte de véhicules est hétérogène et disponible au dépôt. La solution développée permet à la société de transport de combiner plusieurs types d'objectifs et de les paramétrer, comme la minimisation du temps total de voyage, le nombre de véhicules utilisés, le temps d'attente des passagers, la déviation du trajet par rapport au trajet direct et minimiser le coût total du transport.

[Teodorovic et Radivojevic, 2000] utilisent la logique floue pour résoudre un problème de DARP qui consiste à affecter en temps réel chaque nouvelle demande de transport à une tournée et recalculer à nouveau le parcours de la tournée. Une flotte de véhicules homogènes est localisée à un dépôt unique au début de la journée. Plus précisément, au moment de l'apparition d'une nouvelle requête, plusieurs autres passagers ont été déjà acceptés et on doit les servir tout en respectant leurs contraintes : ces passagers sont déjà affectés à un ensemble de tournées de véhicules et une partie d'entre eux sont à bord des véhicules sur le chemin de leur destination, tandis qu'une autre partie des clients attend l'arrivée du véhicule qui va les servir. La logique est utilisée pour modéliser les temps de parcourt des véhicules. Le problème est résolu en deux étapes : la première consiste à déterminer l'affectation d'une nouvelle requête à un véhicule en minimisant la distance totale des tournées et le temps d'attente des véhicules. La deuxième étape consiste à mettre à jour la tournée du véhicule ayant reçu cette requête. Dans ce cas on cherche à minimiser la distance totale parcourue et la durée totale de déviation, c'est-à-dire les retards engendrés par l'affectation de nouvelles requêtes aux véhicules sur les heures de service des requêtes qui sont déjà planifiées

[Fu, 2002] étudie un problème de DARP dynamique dans lequel il ne considère pas l'apparition de nouvelles requêtes, mais seulement les événements liées au temps, comme les retards engendrés par les congestions sur les routes et les accidents. Les durées de voyage sont modélisées par des variables aléatoires, ce qui implique que la fonction objectif devient stochastique. Celle-ci est une somme pondérée de trois termes : le coût de transport, la déviation par rapport à l'heure de service souhaitée par le passager et la déviation par rapport à la durée maximale qu'un passager peut passer à bord du véhicule. Une heuristique d'insertion parallèle a été utilisée pour résoudre ce problème, et elle est inspirée de la méthode proposée dans ([Toth et Vigo, 1997]). Des tests expérimentaux ont été effectués sur des instances contenant 2800 requêtes et 119 véhicules.

[Zhihai *et al.*, 2007] s'intéressent à l'étude d'un problème de DARP dynamique dans un environnement stochastique qui ne se limite pas au seul événement de l'apparition d'une nouvelle requête. D'autres événements stochastiques sont pris en compte : les nouvelles requêtes apparaissent d'une manière aléatoire et les positions des lieux de départ et d'arrivée sont aléatoirement distribuées sur la zone géographique desservie, le temps de voyage entre deux points dépend de l'heure de départ, etc. La fonction objectif utilisée pour l'évaluation des solutions est une combinaison linéaire de 8 critères différents : le coût de transport, la longueur de la tournée, la durée de la tournée, le temps d'attente, le temps de service, la durée de sur-conduite des chauffeurs, l'excès de temps de travail des chauffeurs et les retards des passagers. La méthode utilisée pour résoudre le problème est une recherche locale avec une méthode de diversification qui utilise une fonction objectif secondaire pour guider la recherche. Des expérimentations intensives ont été menées pour évaluer la performance de la méthode proposée. La taille des instances testées est de 610 requêtes.

Dans le domaine hospitalier, [Hanne *et al.*, 2009] ont développé un outil de gestion et de planification « Opti-Trans » de transport de patients. Ce problème de transport de patients fait partie des problèmes de DARP avec des contraintes complexes spécifiques au domaine hospitalier. Pour résoudre ce problème les auteurs ont proposé une heuristique rapide. La fonction objectif à minimiser est une somme pondérée de plusieurs termes dont deux sont liés au coût de transport et deux autres sont liés à la qualité de service rendue aux patients. L'utilisation de ce logiciel a permis de réduire le coût de transport et d'augmenter la qualité de service rendue aux patients. Dans l'article, aucun résultat numérique n'est fourni, ni comparaison avec d'autres méthodes de la littérature. L'objectif de ce travail été axé plus sur l'outil de gestion et moins d'intérêt a été porté sur l'optimisation de la solution.

[Beaudry *et al.*, 2010] ont aussi étudié un problème de DARP dynamique particulier qui survient dans le transport des patients entre les divers bâtiments d'un campus hospitalier. Ce problème contient plusieurs contraintes compliquées spécifiques au domaine hospitalier et les auteurs ont utilisé une heuristique à deux phases pour le résoudre : dans la première phase une méthode simple d'insertion est utilisée pour trouver une solution réalisable. Dans la seconde phase, un algorithme de recherche Tabou est utilisé pour améliorer la solution initiale. La fonction objectif considérée dans ce problème est la minimisation de la somme pondérée de trois critères : la durée cumulée de voyage, la durée cumulée de retard et la durée totale d'avance. L'algorithme a été intensivement testé sur des instances réelles provenant d'un centre hospitalier allemand. Les résultats obtenus montrent que l'algorithme permet d'obtenir des solutions de meilleures qualités en minimisant le temps

d'attente des patients et en réduisant le nombre de véhicules utilisés.

2.2.2.3 Panorama des problèmes multicritères

Dans la revue de la littérature que nous avons effectué, nous n'avons trouvé aucun article qui traite le problème de DARP dynamique multicritère, avec une approche de dominance de Pareto ou autre. La majorité des travaux de la littérature se sont focalisés sur l'agrégation des critères pour ramener le problème à un problème mono-objectif. Ceci peut s'expliquer par la complexité du problème de DARP, ainsi que par la dimension dynamique qui nécessite des algorithmes rapides pour trouver des résultats en temps réel.

2.2.3 Conclusions

En se limitant seulement aux titres des articles publiés sur le problème de DARP multicritère, on peut dire que la littérature est abondante pour cette catégorie de problème. Mais comme nous venons de le montrer précédemment dans la revue de la littérature, la majorité des auteurs qui travaillent sur un problème multicritère utilisent une agrégation de critères, ce qui revient à résoudre un problème mono-objectif qui ne permet d'obtenir qu'une seule solution à la fois.

On ce qui concerne les problèmes de DARP multicritère, nous avons identifié quatre travaux de recherche, dont deux s'intéressent à l'étude d'un problème de DARP bicritère. Les deux critères qui reviennent souvent dans la littérature sont : la minimisation du coût de transport et la maximisation de la qualité de service (point de vue du passager). Pour le critère de la qualité de service, une multitude de définitions ont été utilisées, et la majorité des formulations de ce critère incorporent une composante liée au temps passé à bord d'un véhicule.

2.3 Notions de base en optimisation multicritère

Dans cette section nous présentons la définition de quelques notions et concepts de base liés au domaine de l'optimisation multicritère (ou multiobjectif). Ensuite, Nous présentons brièvement quelques approches qui sont couramment utilisées pour résoudre ces problèmes, Nous renvoyons à [Ehrgott et Gandibleux, 2000] pour un état de l'art en détail sur les problèmes d'optimisation multicritère.

Dans la pratique, on est souvent confronté à des problèmes qui impliquent l'optimisation simultanée de plusieurs critères qui sont en général antagonistes (l'amélioration d'un critère implique la dégradation d'un autre). Comme nous le verrons dans la section 2.4, le problème de Transport à la Demande peut être modélisé comme un problème d'optimisation multicritère.

Dans la suite nous nous intéressons à un problème de minimisation de K critères Z_i, $i \in \{1, ..., K\}$ et soit S l'ensemble des solutions. Lorsque l'on cherche à résoudre un problème d'optimisation multicritère, on obtient souvent une multitude de solutions, du fait que les objectifs sont contradictoires. Un concept intéressant, qui permet de caractériser les solutions obtenues est le *compromis* entre les divers objectifs. En effet, les solutions que l'on

obtient lorsque l'on a résolu le problème sont des solutions dites *de meilleur compromis*. Elles minimisent un certain nombre d'objectifs tout en dégradant les performances sur d'autres objectifs.

La notion d'optimalité que nous considérons dans cette thèse est l'optimalité au sens de Pareto qui est formellement introduites dans la suite de cette section. D'un point de vue aide à la décision, on suppose d'un point de vue général qu'un décideur va nécessairement retenir une solution parmi l'ensemble des optima de Pareto. L'étude des compromis a donné lieu à plusieurs définitions de l'optimalité au sens de Pareto. Les deux plus classiques sont données ci-dessous.

Définition 1 Une solution x est un optimum de Pareto faible s'il n'existe pas de solution y faisable telle que $\forall i \in \{1, ..., K\}, Z_i(y) < Z_i(x)$. Un optimum de Pareto faible est aussi parfois appelé une solution faiblement non-dominée.

Définition 2 Une solution x est un optimum de Pareto strict s'il n'existe pas de solution faisable y telle que $\forall i \in \{1, ..., K\}, Z_i(y) \leq Z_i(x)$ avec au moins une inégalité strict. Un optimum de Pareto strict est aussi parfois appelé une solution strictement non-dominée.

Il est important de noter que tout optimum de Pareto strict est également faible. De même, les solutions réellement intéressantes pour le décideur sont des optima de Pareto stricts, les optima de Pareto faible étant "partiellement dominée". Par abus de langage, un optimum de Pareto strict est souvent appelé *optimum de Pareto* tout simplement. Enfin, on appelle *front de Pareto* l'ensemble des optima de Pareto stricts. La Figure 2.3 illustre les notions d'optimum de Pareto strict et faible.

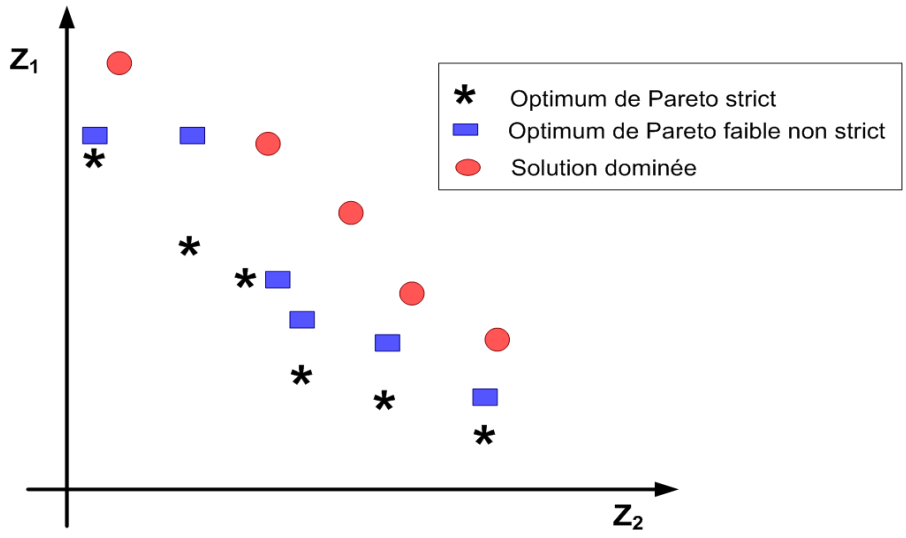


FIGURE 2.3 – Illustration de la notion d'optimum de Pareto

La représentation de l'ensemble des solutions non-dominées est une tâche difficile, surtout quand le nombre de critères à optimiser est supérieur à trois. Les décideurs cherchent toujours à trouver des représentations particulières, qui permettent de voir à quoi correspondent les différentes régions du front de Pareto dans l'espace de décision. Ci-dessous quelques méthodes qui sont couramment utilisées dans la littérature :

- méthode de projections (*Scatter-plot Matrix Method*),
- représentation par le chemin de valeurs (*Path Value Method*),
- visualisation par diagramme (*Bar Chart Method*),
- univers M-dimensionnel (*Star Coordinate System*).

Pour plus d'informations sur ces différentes méthodes, le lecteur est invité à se référer à la thèse de Roudenko ([Roudenko, 2004]).

Il faut noter qu'il n'est pas toujours facile de trouver le front de Pareto exact d'un problème d'optimisation multiobjectif. Ceci est dû à la complexité du problème étudié ou à l'augmentation du nombre d'optima dans le front de Pareto qui peut devenir *exponentiel*. Dans ce cas on se contente souvent d'une approximation de ce front.

On termine cette section par la définition des points particuliers dans l'espace des critères (2.4).

Le *vecteur idéal* z^* est le vecteur de l'espace des critères dont chaque composante z_m est la solution du problème de minimisation de la fonction objectif $Z_m(x)$ sous les contraintes du problème. Généralement, ce vecteur ne correspond pas à une solution de l'espace de décision mais il est quelques fois utile en tant que référence, par exemple, lors de l'évaluation des solutions ou lors la normalisation des valeurs des objectifs.

A la différence du vecteur idéal qui représente les bornes inférieures de chaque objectif dans l'espace faisable, le *vecteur de Nadir* z^{nad} correspond à leurs bornes supérieures sur le front de Pareto et non pas dans tout l'espace faisable (voir Figure 2.4). Le vecteur idéal et le vecteur de Nadir sont en particulier utilisés pour normaliser chaque objectif.

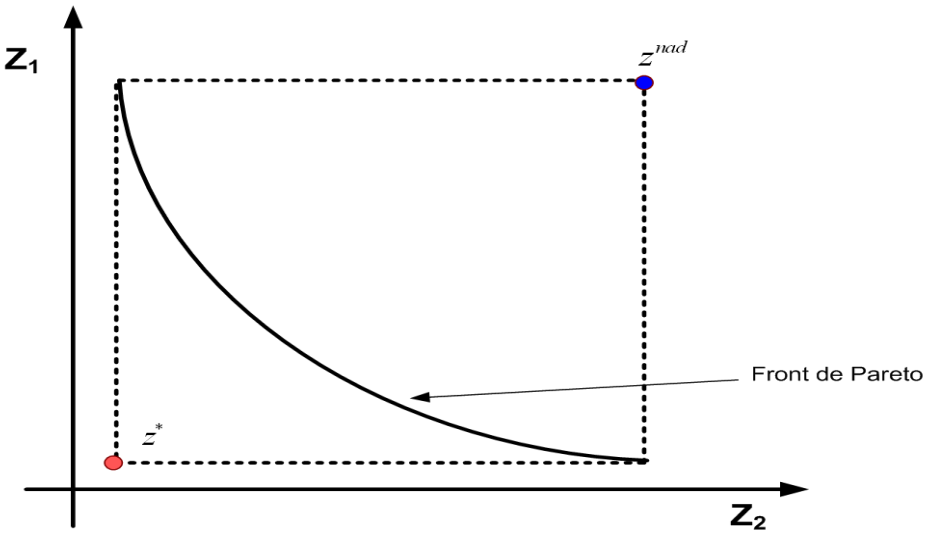


FIGURE 2.4 – Vecteur Ideal et vecteur de Nadir

2.3.1 Méthodes de résolution des problèmes d'optimisation multicritères

L'optimisation multicritère a fait l'objet du développement de nombreuses méthodes ayant pour but de déterminer un optimum de Pareto ou un front de Pareto exact ou approximatif. Afin de gérer les critères conflictuels, un décideur doit intervenir. Le moment d'intervention du décideur définit les trois grandes approches de résolution d'un problème multicritère (voir Figure 2.5 et 2.6) :

- a priori : le décideur modélise ses préférences avant la résolution et unique optimum de Pareto est calculé,
- interactive : le décideur interagit avec le processus de résolution pour obtenir l'optimum de Pareto qu'il préfère,
- a posteriori : le décideur choisit une solution parmi un ensemble d'optima de Pareto potentiels .

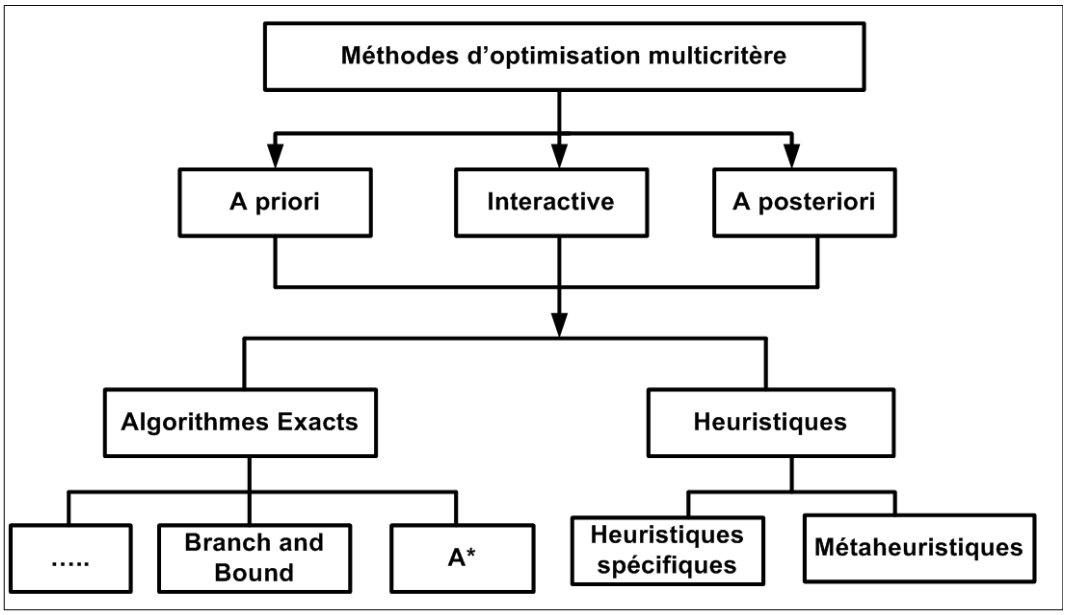


FIGURE 2.5 – Classification des méthodes d'optimisation multicritère

Un panorama de telles méthodes est disponible dans ([Talbi, 1999]) et ([T'kindt et Billaut, 2002])
.

2.3.1.1 Les méthodes *a priori*

Ces méthodes consistent à construire une seule fonction objectif qui englobe tous les critères afin de se ramener à un problème d'optimisation mono-objectif. Nous présentons ci-dessous quelques techniques :

- Utilisation de l'ordre lexicographique,
- Détermination par combinaison convexe des critères,
- Détermination à l'aide de métriques (Tchebycheff,...),
- Détermination par l'intégrale de Choquet,

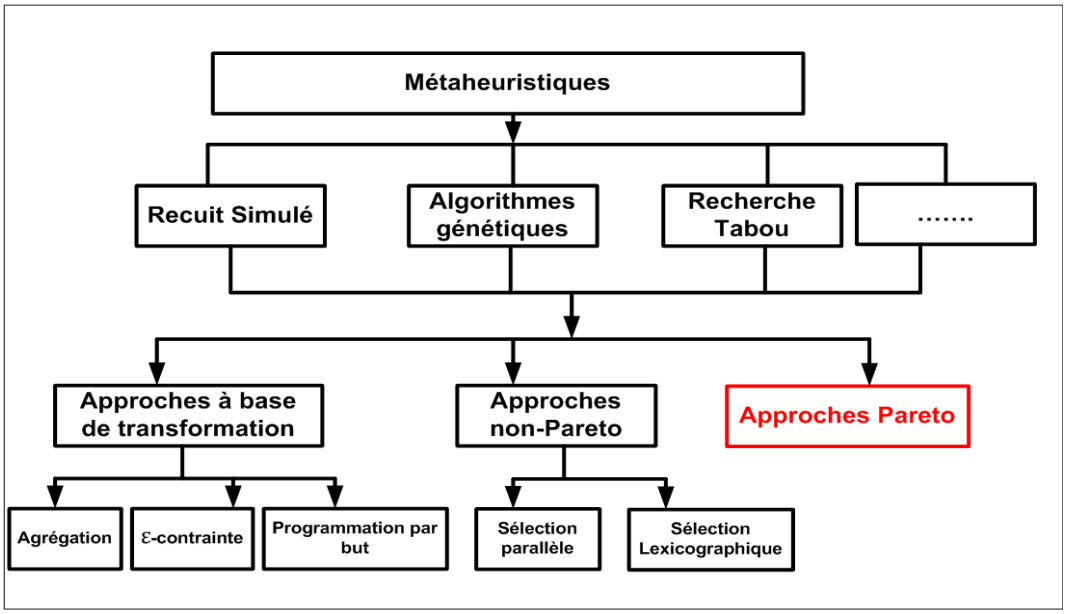


FIGURE 2.6 – Quelques métaheuristiques pour l'optimisation multicritère

– Détermination par l'approche ϵ-contrainte.

Le choix de la stratégie à adopter dépend essentiellement des volontés du décideur et des informations qu'il est capable de fournir concernant les différents critères. Les questions essentielles auxquelles il doit répondre sont : les compensations entre les critères sont-elles autorisées ? Dispose-t-on d'objectifs à atteindre pour certains critères ? A-t-on des bornes que ne doivent pas dépasser certains critères ?

– **Utilisation de l'ordre lexicographique.**

L'utilisation de l'ordre lexicographique est nécessaire dès lors qu'aucune compensation n'est possible ou désirée entre les critères. On a donc une fonction objectif que l'on note $Lex(Z_1, Z_2, ..., Z_K)$ et on suppose que le décideur a classé les critères dans l'ordre de préférence. Soit S l'ensemble des solutions faisables du problème. On définit les sous-ensembles de S suivants :

$$
\begin{aligned}
S^1 &= \{x^* \in S / Z_1(x^*) = min_{x \in S}(Z_1(x))\} \\
S^2 &= \{x^* \in S / Z_2(x^*) = min_{x \in S^1}(Z_2(x))\} \\
&... \\
S^k &= \left\{x^* \in S^{K-1} / Z_K(x^*) = min_{x \in S^{K-1}}(Z_K(x))\right\}
\end{aligned}
$$

Cette approche consiste alors à chercher l'ensemble S^1 des solutions minimisant Z_1, puis son sous-ensemble S^2 dont les solutions ont des valeurs de Z_2 qui sont minimales, et ainsi de suite pour les $K-2$ critères restant ([T'kindt et Billaut, 2002]).

– **Détermination par combinaison convexe des critères.**

Lorsqu'il y a des compensations possibles entre les critères, on peut utiliser les méthodes par combinaison convexe des critères. Ces méthodes, probablement les plus largement utilisées dans la pratique, consistent à ramener le problème multicritère au problème d'optimisation d'une combinaison linéaire des objectifs initiaux : à chaque critère Z_i est associé à un poids $\alpha_i > 0$ et on minimise $\sum_{i=1}^{k} \alpha_i Z_i$. Normalement, cette approche a du sens si chaque objectif peut être ramené à une même unité de mesure (par exemple convertir le retard en coût, la distance en coût).

– **Détermination à l'aide de métriques.**

D'autres métriques peuvent être utilisées pour agréger les critères, afin de résoudre un problème mono-objectif. La norme de Tchebychef est l'une des métriques les plus utilisées. Le décideur définit une solution idéale (ou de référence) : cette solution peut être calculée en optimisant individuellement chaque objectif (vecteur idéal). La solution optimale est alors la solution minimisant la distance de Tchebycheff à la solution idéale.

– **Détermination par l'intégrale de Choquet.**

Afin de prendre en considération les interactions entre les différents critères, l'intégrale de Choquet se présente comme un moyen de modéliser les préférences du décideur plus finement que par la simple linéarisation. Avec cette modélisation il est possible de définir des préférences entre deux ou plusieurs solutions, plutôt que simplement associer un poids à chaque solution comme c'est le cas dans la technique de la somme pondérée des critères. Une introduction générale à cette approche est disponible dans ([Grabisch, 2006]).

– **Détermination par l'approche ϵ-contrainte.**

Nous décrivons maintenant l'approche $\epsilon - \textit{contrainte}$. Il s'agit d'une approche pour laquelle on dispose de bornes supérieures que doivent respecter $K-1$ des K critères. On cherche alors la solution respectant ces bornes et minimisant le critère non borné. La fonction objectif se note $\epsilon(Z_1/Z_2, ..., Z_k)$ dans le cas où on recherche la solution minimisant Z_1 dans l'ensemble des solutions ayant des valeurs de Z_i telles que $Z_i \leq \epsilon_i$, $\forall i \in 2, ..., K$, La Figure 2.7 illustre l'approche $\epsilon - \textit{contrainte}$ dans le cas bicritère, lorsque la borne ϵ_2 sur Z_2 est fixée.

Eu utilisant une telle approche, nous avons les garanties suivantes :

- la solution optimale du problème $\epsilon - \textit{contrainte}$ est un optimum de Pareto faible,
- pour chaque optimum de Pareto strict, il existe un problème $\epsilon - \textit{contrainte}$ dont il est la solution optimale.

2.3.1.2 Les méthodes interactives

Les approches interactives consistent à déterminer une solution de compromis en fonction des préférences du décideur qui vont s'affiner pendant la résolution. Pour cela, deux phases sont répétées alternativement jusqu'à l'obtention d'un optima de Pareto satisfaisant le décideur :

- la phase de calcul d'une ou plusieurs solutions de compromis,
- la phase de dialogue avec le décideur où ce dernier fait évoluer ses préférences en fonction des solutions qui lui sont proposées.

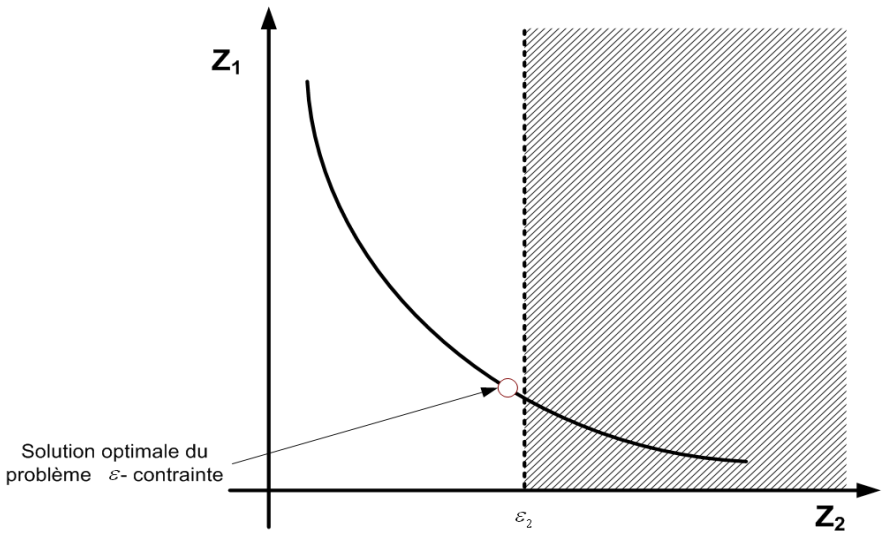


FIGURE 2.7 – Approche $\epsilon-$contrainte de détermination d'un optimum de Pareto

Il faut préciser que la phase de calcul dans ces méthodes repose également sur les mêmes méthodes que dans les méthodes *a priori*.

2.3.1.3 Les méthodes *a posteriori*

Ces approches visent à obtenir le front de Pareto le plus complet possible. L'ensemble des solutions est alors présenté au décideur qui choisira celle qu'il considère comme la plus adaptée. Cette approche à l'avantage de n'éliminer aucune solution qui aurait pu intéresser le décideur. La contrepartie est que la représentation de l'ensemble du front de Pareto est aisée avec deux objectifs et elle devient problématique au delà. De plus, le décideur souhaite généralement devoir uniquement choisir entre quelques solutions et non entre plusieurs milliers, d'autant plus que de nombreuses solutions peuvent être similaires.

Dans la suite nous présentons quelques méthodes basées sur cette approche, et plus particulièrement quelques méthodes évolutionnaires qui sont largement utilisées dans la littérature pour résoudre des problèmes d'optimisation multicritères.

Les algorithmes évolutionnaires sont reconnus comme des techniques particulièrement adaptées à la recherche du front de Pareto. Ces algorithmes travaillent sur une population de solutions candidates, à la différences de la plupart des autres méthodes qui manipulent en général une seule solution. Cette particularité des algorithmes évolutionnaires rend possible l'obtention d'un ensemble de Pareto approché, en un seul essai de l'algorithme et sans obliger l'utilisateur à définir des paramètres subjectifs supplémentaires, tels les poids relatifs des critères d'optimisation. Un autre avantage des algorithmes évolutionnaires, qui n'est pas spécifique aux problèmes multiobjectifs, est que ce type d'algorithmes peut être appliqué aussi bien aux problèmes discrets que continus ou mixtes.

Les méthodes évolutionnaires appartiennent à la classe des méthaheuristiques à base de population, qui consiste à améliorer une population de solutions d'une manière itérative en utilisant des opérateur de croisement, mutation et sélection.

Les algorithmes évolutionnaires se montrent bien adaptés au traitement de problèmes multiobjectif et leur conception est indépendant du problème à résoudre et se basent sur les principaux composants suivants :

1. la représentation des solutions,
2. la stratégie d'initialisation de la population,
3. l'évaluation,
4. les opérateurs de reproduction (croisement, mutation,....),
5. la stratégie d'affectation d'une valeur de fitness,
6. la stratégie de préservation de la diversité,
7. la stratégie de sélection,
8. la stratégie de remplacement,
9. la stratégie de gestion de l'archive (s'il y en a un),
10. la condition d'arrêt.

En se basant sur la conception générale d'un algorithme évolutionnaire, plusieurs algorithme évolutionnaires multiobjectifs ont été proposés durant les deux derniers décennies. Nous présentons dans la suite quelques unes de ces méthodes.

– ***Vector Evaluated Genetic Algorithm*(VEGA)**

En 1985, Schaffer a proposé le premier algorithme évolutionnaire multiobjectif pour trouver les solutions non-dominées d'un problème multiobjectif ([Schaffer, 1985]). Il l'a appelé *VEGA : Vector Evaluated Genetic Algorithm*. Dans cette méthode le schéma de l'évaluation est très simple : à chaque génération, la population est divisée en autant de sous-populations (de tailles égales) qu'il y a d'objectifs. Ensuite chaque sous-population est évaluée selon l'une des fonctions objectifs. L'opérateur de sélection est ensuite restreint à chaque sous-population (sélection parallèle) (voir Figure 2.8). Ensuite, l'algorithme VEGA recompose la population complète et applique les opérateurs de croisement et mutation.

Comme l'algorithme affecte aléatoirement les individus aux sous-populations à chaque génération, un même individu peut être évalué différemment suivant l'objectif qui lui est affecté, d'une génération à l'autre. L'analyse de l'algorithme VEGA a montré que son comportement est le même qu'un algorithme réalisant une agrégation linéaire ([Richardson *et al.*, 1989]).

VEGA est particulièrement adapté aux problèmes où on cherche surtout de bonnes solutions individuelles pour chacun des différents critères. D'une itération à l'autre chaque solution est sélectionnée selon l'un des critères : donc même si on peut espérer trouver de bons compromis à l'aide du croisement entre les bons individus, il est peu probable que de tels compromis puissent survivre assez longtemps aux cours de l'évolution parce que lors de la sélection, les solutions qui sont meilleures selon l'un des objectifs sont sélectionnées.

– ***Strenth Evolutionary Algorithm* (SPEA et SPEA-II)**

La méthode évolutionnaire élitiste pour l'optimisation multicritère *Strength Pareto Evolutionary Algorithm* (SPEA) a été proposée par ([Zitzler *et al.*, 1998]). Cette méthode maintient une population externe P', appelée aussi « archive », contenant les meilleures solutions de compromis rencontrées durant la recherche. A chaque itération, de nouveaux

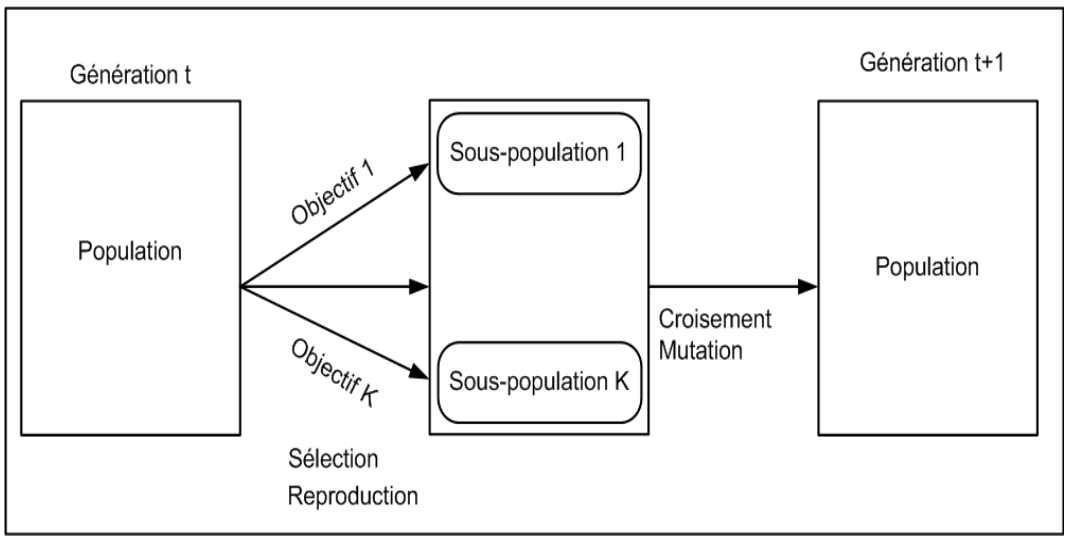


FIGURE 2.8 – La sélection parallèle dans l'algorithme VEGA

individus non-dominés sont comparés aux membres de P' au sens de la dominance de Pareto et seules les solutions non-dominées résultantes restent ensuite dans P'.

Cet algorithme démarre avec une population P_0 de taille N, créée aléatoirement, et avec une population externe $P_0^{'}$ vide. À chaque génération t, les solutions non-dominées de P_t sont copiées dans la population externe $P_t^{'}$ et ensuite toutes les solutions dominées, qui peuvent apparaitre du fait de l'ajout de nouveaux éléments à $P_t^{'}$ sont éliminées. Quand la taille de $P_t^{'}$ atteint la limite N', un critère supplémentaire pour accepter une solution dans l'archive rentre en jeu. Ce critère est basé sur le principe de Clustering afin de préserver la diversité entre les solutions de $P_t^{'}$.

Dans SPEA-II ([Zitzler *et al.*, 2001]) qui est une amélioration de SPEA, des modifications ont été ajoutées, notamment au niveau de la technique de contrôle de l'archive qui a été remplacée par une autre méthode de troncation qui préserve les individus qui sont éloignés les uns des autres. Dans SPEA-II, seules les solutions de l'archive participent au processus de reproduction.

– ***Nondominated Sorting Genetic Algorithm*** **(NSGA-II).**

[Deb *et al.*, 2002] ont proposé un algorithme élitiste qu'ils ont appelé NSGA-II, cet algorithme est une extension de la méthode NSGA. Dans NSGA-II, à une itération t la population des enfants Q_t est d'abord créée à partir de la population des parents P_t. Ensuite, elles sont réunies dans un ensemble $R_t = P_t \cup Q_t$, qui est trié selon le principe de la dominance de Pareto. La population suivante est remplie par les solutions des sous-ensemble non dominés de R_t l'un après l'autre en commençant, évidemment par le premier front. Vu que la taille de R_t est $2N$ tous ses sous-ensembles ne peuvent pas rentrer dans la nouvelle population dont la taille doit être égale à N.

Pour choisir les solutions qui vont survivre du front dont seulement une partie peut être placée dans la population suivante, une mesure de la densité des solutions dans l'espace de

critères est utilisée. Un des avantages de NSGA-II est qu'il n'exige aucun paramètre à fixer pour préserver la diversité dans une population. Nous revenons en détail sur cette méthode dans le chapitre 4 de ce manuscrit.

Des travaux de comparaison des algorithmes évolutionnaires ([Zitzler *et al.*, 2000]) ont montré que l'introduction de l'élitisme a mené à des performances nettement supérieures. Actuellement les deux algorithmes élitistes NSGA-II ([Deb *et al.*, 2002]) et SPEA-II ([Zitzler *et al.*, 2001]) sont considérés comme les plus populaires et les plus performants ([Coello *et al.*, 2007]). Il s'agit généralement des méthodes les plus utilisées de part leur capacité à bien fonctionner sur un grand nombre de problèmes. Mais le choix entre ces deux méthodes ne permet pas de faire un choix sans hésitation. Dans la pratique le choix entre NSGA-II et SPEA-II est plutôt influencé par des particularités de leur mise en oeuvre, et par le choix a priori de certains paramètres, comme le paramètre de la taille de l'archive qu'on retrouve dans SPEA-II et qu'on ne le retrouve pas dans NSGA-II. La table (2.2) montre les différences entre ces deux méthodes élitistes.

Composants	NSGA-II	SPEA-II
fitness	rang de dominance (dominance de Pareto)	compte et rang de dominance (dominance de Pareto)
diversité	distance de crowding	plus proche voisin
sélection	élitiste	élitiste
remplacement	remplacement élitiste	remplacement générationnel
gestion de l'archive	aucun	archive de taille fixe
condition d'arrêt	nombre de générations	nombre de générations

TABLE 2.2 – Comparaison des deux algorithmes évolutionnaires multiobjectifs NSGA-II et SPEA-II

Nous revenons en détail sur cette méthode dans le chapitre 4 dans lequel nous expliquons les différents mécanismes de cet algorithme ainsi que sa mise en oeuvre pour résoudre notre problème de DARP statique tricritère. Dans la section suivante nous présentons le problème de Transport à la Demande que nous abordons dans cette thèse.

2.4 Présentation et formulation du problème

La problématique de cette thèse est d'étudier un problème de transport à la demande qui prend en considération les préoccupations de l'ensemble des intervenants. Étant donné que le transport à la demande est un service public, souvent mis en place par des collectivités locales (au moins en France), la mise en place d'un tel service ne doit pas prendre en compte seulement l'aspect économique du service en ne minimisant que le coût de transport, mais aussi les attentes des passagers en terme de qualité de service. Une autre dimension, écologique, est devenue de plus en plus importante durant ces dix dernières années : cette dimension représente l'intérêt public et consiste à minimiser l'impact environnemental. Cet aspect est soulevé par les collectivités locales et les élus quand ils envisagent de fournir un service de transport à la demande sur leur territoire. Donc, pour répondre aux attentes de ces trois intervenants (le transporteur, les passagers et la collectivité locale), il faudra

prendre leur objectif, généralement conflictuels, en considération lors de l'organisation et la planification du service.

Dans la littérature du transport à la demande, l'aspect qualité de service est présenté selon des points de vues différents ([Paquette *et al.*, 2009]) : d'un point de vu général dans lequel la qualité de service regroupe les attentes de tous les intervenants (la qualité du système de transport à la demande), de point de vu planificateur (transporteur) qui considère que le coût de transport fait partie des mesures de la qualité de service, et du point de vu des passagers pour lesquels la qualité de service peut être mesurée en fonction du respect de leurs exigences en termes de temps, de mode de réservation utilisé (facile ou difficile), des attitudes des chauffeurs, etc.

Dans la suite de ce document le terme qualité de service fait référence toujours à la qualité du service perçue par les usagers. Elle est calculée en fonction de la satisfaction des passagers selon le respect de leurs horaires souhaités et leur temps d'attente à bord du véhicule.

Comme cela avait été mentionné au début de cette section, l'un des plus importants impacts attendus de ce mode de transport porte principalement sur l'aspect environnemental conjugué avec un aspect social. Le premier aspect vise à obtenir des gains significatifs au niveau de la consommation énergétique et des émissions de CO_2 afin de contribuer à la réduction de l'impact de l'activité humaine sur le réchauffement climatique. Le deuxième aspect s'intéresse aux impacts sociétaux : l'objectif est d'offrir un service de meilleur qualité qui répond aux besoins des passagers et améliore la qualité de leur mobilité.

Ces trois objectifs nous ont permis de définir un problème de transport à la demande tricritère, qui se décline en deux versions :
- la version statique du transport à la demande (DARP statique),
- la version dynamique du transport à la demande (DARP dynamique).

2.4.1 Définition et formulation du problème tricritère de DARP statique

Dans la suite nous commençons par la définition et la formulation du DARP statique tricritère dans lequel les requêtes à servir sont connues à l'avance. Chaque requête est caractérisée par un ensemble de personnes à transporter, un point de départ et d'arrivée et des fenêtres de temps dans lesquelles les passagers souhaitent être servis. Le problème que nous traitons dans cette thèse fait partie des problèmes d'optimisation multicritères, pour lequel nous cherchons un ensemble d'optima de Pareto.

Dans le DARP considéré, chaque personne (ou ensemble de personnes) émet une requête de transport définie par un point d'embarquement, un point de débarquement et un nombre de personnes à servir. Toutes les personnes d'une même requête ont le même point de départ et d'arrivée. Le but de ce problème est de planifier l'ensemble des requêtes R à l'aide d'une flotte de véhicules hétérogènes M. Pour toute requête, l'embarquement doit être effectué avant le débarquement et par le même véhicule. Dans le DARP considéré la « préemption » des requêtes n'est pas autorisée. Cette préemption se manifeste en deux cas de figures : premièrement, une requête ne peut pas être divisée en plusieurs requêtes pour être traitée par plusieurs véhicules, lorsqu'une requête est formulée par plusieurs

personnes. Deuxièmement, une requête ne peut pas être servie par plusieurs véhicules, c'est-à-dire qu'on n'autorise pas de correspondances entre véhicules.

Le DARP peut être décomposé en trois sous-problèmes : un problème d'affectation qui consiste à déterminer par quel véhicule est servie quelle requête (ce problème est connu sous le nom de *ressource allocation problem*), un problème de séquencement, qui consiste à déterminer dans quel ordre chaque véhicule visite les lieux qui lui sont attribués (ce problème est connu sous le nom de *routing problem*) et un problème d'horodatage qui consiste à déterminer l'heure de passage des véhicules en chaque lieu (ce problème est connu sous le nom de *timing problem*). Ces trois sous-problèmes peuvent être traités globalement par coopération d'algorithmes, ou indépendamment par une séquence d'algorithmes. Lors d'une décomposition *clustering first-routing second*, le problème de DARP correspond à résoudre plusieurs DARP avec un seul véhicule. Notons qu'un véhicule ne peut être affecté qu'à une seule tournée sur l'horizon de temps considéré ($[0, T]$).

Dans la suite nous définissons les données, les contraintes et les fonctions objectif du problème considéré dans cette thèse.

2.4.1.1 Les données liées aux requêtes des passagers

On définit l'ensemble des requêtes des passagers par $R = \{r_1, ..., r_n\}$ et chaque requête r_j est caractérisée par :

- v_j : le lieu de départ (l'origine),
- v_{j+n} : le lieu d'arrivée (la destination),
- $[e_j, l_j]$: la fenêtre de temps au sommet v_j , avec e_j l'heure au plus tôt, et l_j l'heure au plus tard à laquelle les passagers de la requête r_j souhaitent être servis, on définit de même un intervalle $[e_{j+n}, l_{j+n}]$ associée au sommet v_{j+n},
- q_j : le nombre de personnes de la requête r_j. On suppose que toutes les personnes doivent être embarquées à un seul et même point de départ et débarquées à un seul et même point d'arrivée,
- s_j : le temps de service associé au sommet v_j,
- s_{j+n} : le temps de service associé au sommet v_{j+n},
- d_{ij} : la distance qui sépare les deux sommets v_i et v_j,
- t_{ij} : le temps nécessaire pour aller de v_i à v_j par un trajet direct,
- α_j et β_j (α_{j+n} and β_{j+n}) deux paramètres pour pénaliser les retards et les avances aux points de départ et d'arrivée. Ces paramètres seront utilisés dans le calcul de la qualité de service rendue aux passagers. La fonction objectif relative au calcul de la qualité de service sera détaillée un peu plus loin dans cette section,
- ω_j : la pénalité unitaire associée au temps d'attente des passagers de la requête r_j à bord du véhicule. Ce paramètre sera également utilisé dans la fonction objectif de la qualité de service.

La Figure 2.9 montre un exemple de trois requêtes et les informations associées à chaque sommet.

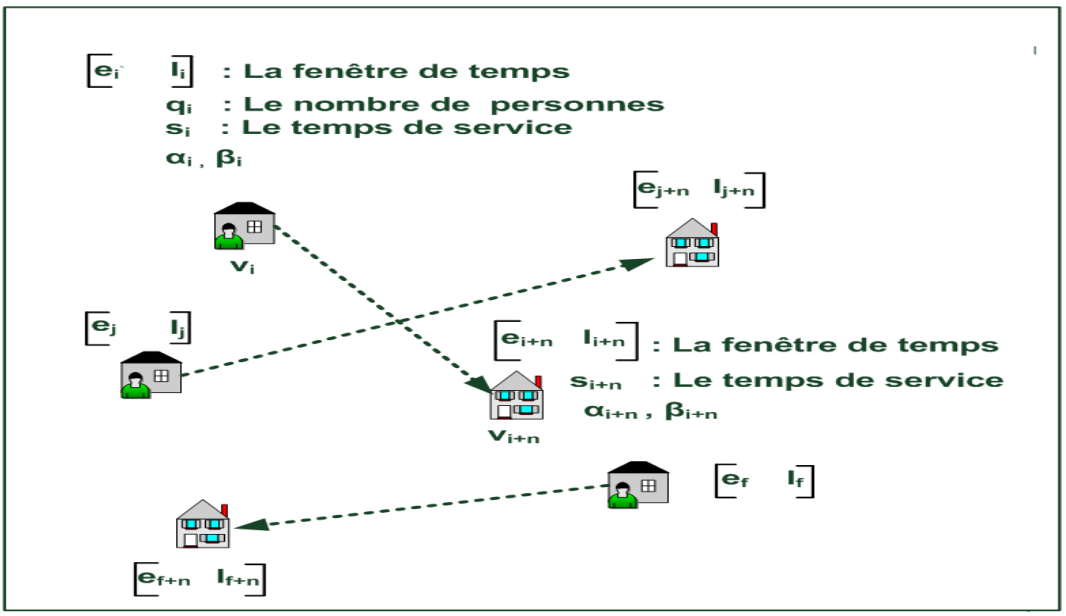


FIGURE 2.9 – Exemple de requêtes de transport

2.4.1.2 Les données liées aux véhicules

Le problème de DARP que nous étudions est un problème multi-véhicules dans lequel nous considérons une flotte de véhicules hétérogène, cette hétérogénéité est notamment observée au niveau du coût kilométrique de transport et de la quantité de CO_2 émise par kilomètre. Ces deux paramètres dépendent du type de véhicule utilisé : on peut avoir le choix entre un véhicule ancien dont le coût est moins élevé (parce qu'il a été amorti) mais qui pollue plus et un véhicule neuf qui coûte plus chère, mais avec une émission de CO_2 faible. Habituellement, dans le transport à la demande on fait appel aux taxis pour servir quelques requêtes. Dans ce cas, il peut y avoir un arbitrage à faire entre l'utilisation d'un grand bus qui pollue plus mais moins cher et la location de taxis qui polluent moins mais coûte très cher. Les données que nous considérons pour les véhicules sont les suivantes :

- v_0 : le dépôt commun de l'ensemble des véhicules,
- M : le nombre total de véhicules disponibles,
- Q_k : la capacité du véhicule k, avec $k \in \{1, ..., M\}$,
- b_k : le coût kilométrique de l'utilisation du véhicule k, avec $k \in \{1, ..., M\}$,
- h_k : émission de CO_2 du véhicule k en grammes par kilomètre, avec $k \in \{1, ..., M\}$.

2.4.1.3 Les données liées au réseau de transport

Les points de départ et d'arrivée des passagers peuvent être assimilés à des sommets dans un graphe $G = (V, A)$, avec V un ensemble de $2n + 1$ sommets indexés par i, et n le nombre de demandes (requêtes). V est constitué des trois sous-ensembles suivants : $V = P \cup D \cup \{v_0\}$ où v_0 représente le dépôt, $P = \{v_1, ..., v_n\}$ représente le sous-ensemble des lieux d'embarquement des passagers. De plus, $D = \{v_{n+1}, ..., v_{2n}\}$ le sous-ensemble des

lieux de débarquement des passagers et $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ est l'ensemble des arrêtes du graphe. On associe à chaque arrête qui relie le noeud v_i et v_j une distance d_{ij} et un temps de voyage t_{ij} et on associe aussi à chaque noeud $v_i \in V$ une durée de service non nulle s_i (embarquement, débarquement).

2.4.1.4 Les variables de décisions

Dans la suite nous définissons les variables de décision que nous avons utilisé pour la formulation du problème de DARP statique. Nous introduisons des variables A_i^k, $\forall k = 1, ..., M, \forall i = 0, ..., 2n$, qui correspondent à l'heure d'arrivée du véhicule k au noeud v_i. De la même manière, les variables D_i^k, $\forall k = 1, ..., M, \forall i = 0, ..., 2n$, définissent l'heure du départ du véhicule k au noeud v_i et les variables E_i^k $\forall k = 1, ..., M, \forall i = 0, ..., 2n$, valent le nombre de places occupées dans le véhicule k après avoir servi le noeud v_i. Nous définissons aussi des variables D_k, $\forall k = 1, ..., M$, qui représentent la distance totale parcourue par le véhicule k et S_k, $\forall k = 1, ..., M$, représentent l'ensemble des sommets v_i servis par le véhicule k.

2.4.1.5 Les fonctions objectif

Les trois critères à minimiser sont : le coût total de transport, l'insatisfaction des usagers et la quantité totale de CO_2 émise par l'ensemble des véhicules. Dans la suite nous utilisons les notations suivantes :

- Vh_k : la tournée de véhicule associée au véhicule k,
- $\overline{QoS_k}$: l'insatisfaction des usagers servis par le véhicule k,
- $\overline{QoS}$: l'insatisfaction totale des usagers (la qualité de service totale rendue à l'ensemble des passagers),
- TC : le coût total de transport,
- CO_2 : la quantité totale de dioxyde de carbone émise par l'ensemble des véhicules.

(1) Minimiser le coût total de transport.

Même si le transport à la demande est commandité par des collectivités locales, la prise en compte des coûts financiers lors de la planification des tournées est nécessaire, que ce soit par la minimisation des coûts d'exploitation du service ou par la maximisation des profits. Dans le cas de notre problème, ce critère consiste à minimiser le coût total de transport qui dépend de la distance parcourue et de la catégorie des véhicules utilisés. Il est défini par la formule suivante :

$$TC = \sum_{k=1}^{M} b_k D_k,$$

(2) Minimiser l'insatisfaction des passagers.

Généralement, la minimisation des coûts de transport ne conduit pas à réduire l'insatisfaction des passagers. Mais la prise en compte des besoins des utilisateurs en termes de confort et flexibilité peut avoir un impact économique positif à plus ou moins long terme

sur le transport à la demande : il vaut mieux éviter de rendre des usagers très mécontents car cela peut faire fortement chuter la demande et par la suite la rentabilité.

La fonction objectif de minimisation des insatisfactions des passagers est composée de deux termes : le premier terme concerne les violations des fenêtres de temps souhaitées par les usagers et le deuxième terme concerne le temps d'attente des passagers à bord des véhicules.

Pour chaque tournée Vh_k formée par des sommets de l'ensemble S_k, on définit le critère de l'insatisfaction des passagers comme suit :

$$\overline{QoS_k} = \sum_{j \in S_k} \left(max\left(0, \alpha_j(e_j - A_j^k)\right) + max\left(0, \beta_j\left(A_j^k - l_j\right)\right)\right) + \sum_{\substack{i,j \in S_k \\ i \to j}} w_i(A_j^k - (A_i^k + t_{i,j} + s_i)),$$

où $\alpha_j = 0$ si le sommet v_j correspond à un point d'embarquement, w_i est la somme des ω_i qui correspondent aux passagers qui sont à bord du véhicule k au sommet v_i et $i \to j$ signifie que le sommet v_i précède immédiatement le sommet v_j dans la tournée Vh_k.

Comme on peut le voir sur la formule de l'insatisfaction $\overline{QoS_k}$, le critère n'est pas seulement une somme d'évaluations pourtant sur chaque requête, mais il dépend aussi des autres requêtes servies par le même véhicule.

Le véhicule k associé à une tournée Vh_k, peut se trouver en trois états différents : au dépôt, en route ou arrivé à un sommet. Quand le véhicule arrive à un sommet v_i il peut aussi être dans l'un des trois états suivants :

- **En avance.** Dans ce cas, si le sommet v_i correspond à un point de départ d'une requête aucune pénalité n'est calculée, parce que c'est le véhicule qui va attendre jusqu'à e_i (la borne inférieure de la fenêtre du temps qui correspond au sommet v_i). Du point de vu des personnes à servir au sommet v_i leur qualité de service n'est pas dégradée même si le véhicule doit attendre. Si le sommet v_i correspond à un point d'arrivée d'une requête, alors les personnes concernées vont descendre même si le véhicule arrive en avance (voir Figure 2.10) et une pénalité d'avance est calculée : $\alpha_i(e_i - A_i^k)$.
- **A l'heure.** Le véhicule est arrivé à l'heure (voir Figure 2.11) si le sommet v_i (départ ou arrivée) est servi dans sa fenêtre de temps $[e_i, l_i]$.
- **En retard.** Le véhicule est en retard (voir Figure 2.12) si le sommet v_i est servi après sa borne supérieure l_i. Dans ce cas une pénalité de retard est calculée : $\beta_i(A_i^k - l_i)$.

Le second terme de la fonction objectif de l'insatisfaction des passagers a pour but de minimiser le temps passé à bord des véhicules par les passagers. Dans une tournée ce terme est égale à la somme à chaque sommet v_i de la différence entre l'heure d'arrivée au v_j (successeur du sommet v_i) multiplié par un poids w_i.

(3) Minimiser la quantité de CO_2 émise par l'ensemble des véhicules

Dans cette fonction objectif, le but est de minimiser la quantité de CO_2 émise par l'ensemble des véhicules. Le fait que les véhicules soient hétérogènes justifie la prise en

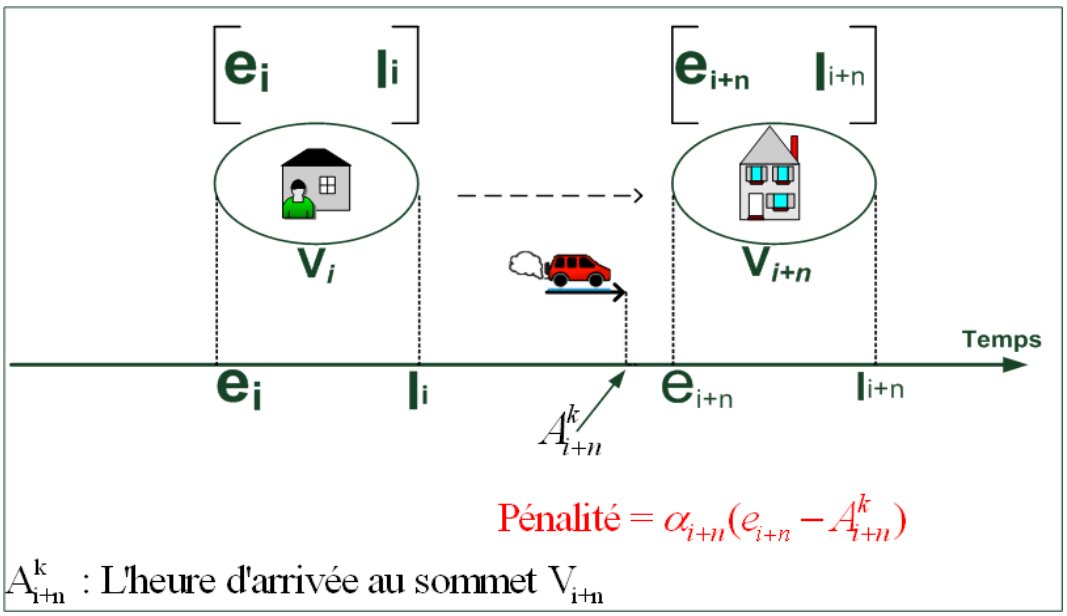


FIGURE 2.10 – Véhicule en avance au sommet d'arrivée v_{i+n}

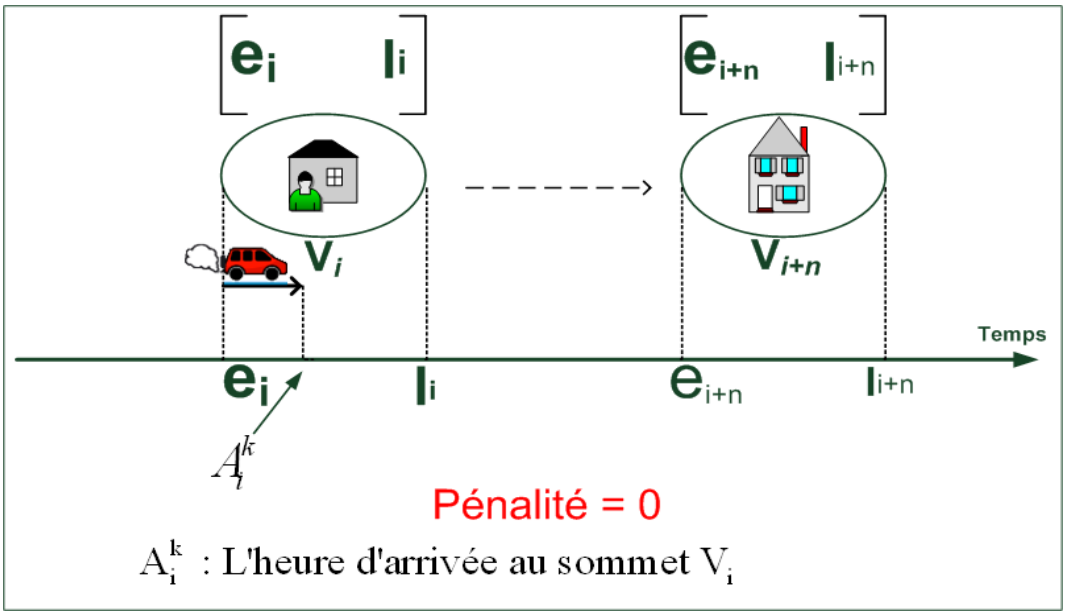


FIGURE 2.11 – Véhicule arrivé à l'heure au sommet de départ v_i

compte de ce critère dans notre problème. Notamment que le coût de transport n'est pas toujours corrélé positivement à l'émission de CO_2. [Dessouky *et al.*, 2003] ont expliqué, en utilisant des simulations sur des données réelles, l'indépendance entre le coût de transport et la quantité de CO_2 émise par les véhicules. Ils ont conclu que le fait d'augmenter légèrement le coût de transport et retarder légèrement le service pour quelques passagers peut réduire significativement la quantité de CO_2 émise dans la nature. Ce critère est défini par la

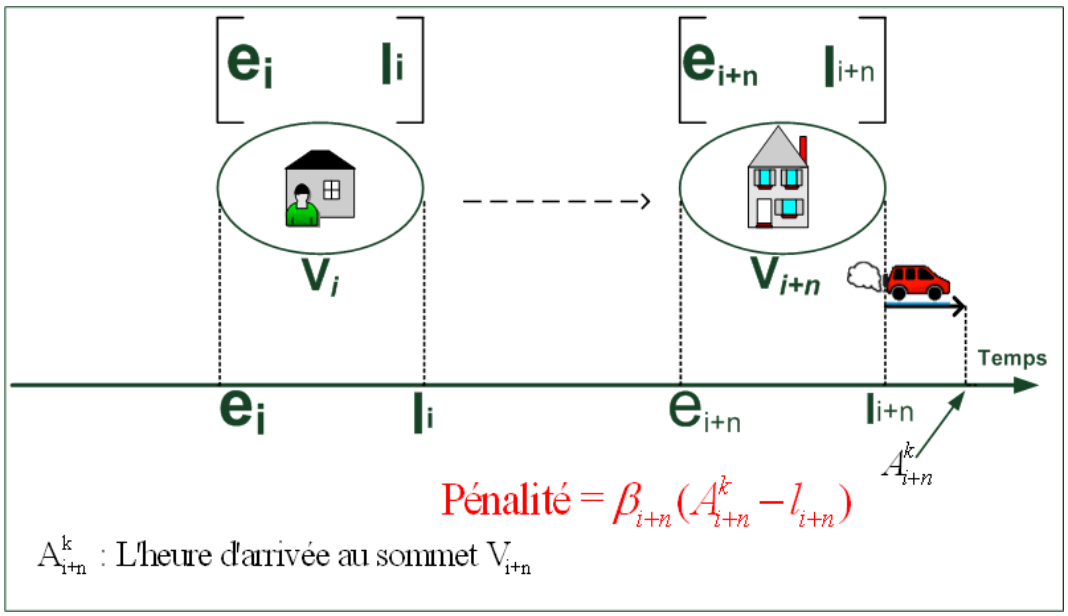


FIGURE 2.12 – Véhicule en retard

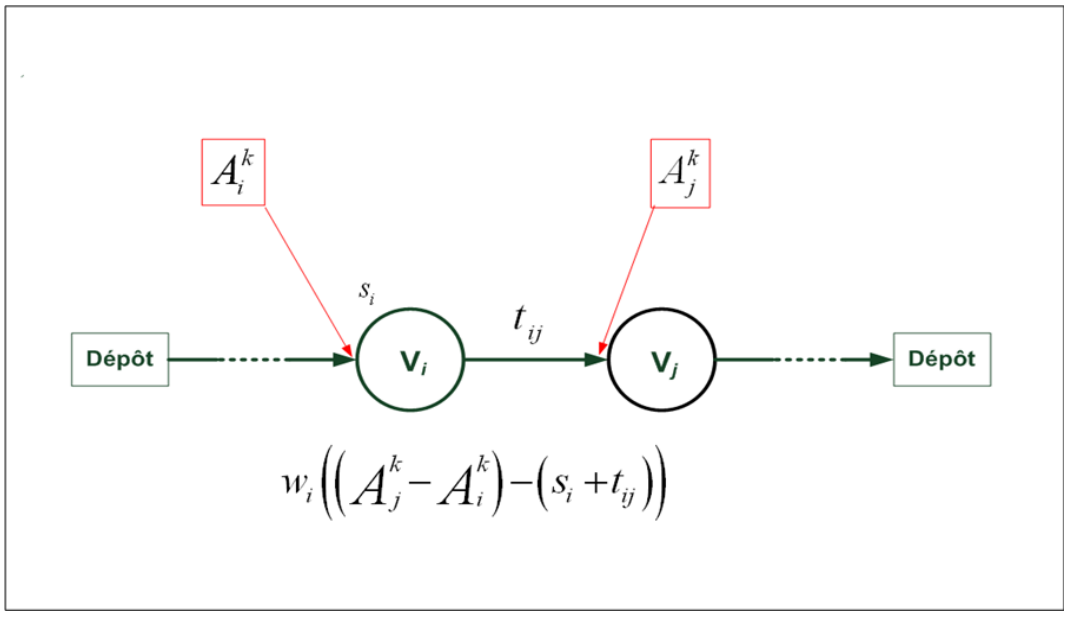


FIGURE 2.13 – Le temps d'attente à bord du véhicule

formule suivante :

$$CO_2 = \sum_{k=1}^{M} h_k D_k.$$

2.4.1.6 Les contraintes

Les contraintes principales qui définissent notre problème de DARP sont présentées ci-dessous.

Si la requête r_j est servie par le véhicule k, alors les contraintes suivantes doivent être respectées :

$$A_j^k \geq A_i^k + s_i + t_{ij} \text{ et } A_{j+n}^k \geq A_l^k + s_l + t_{l,j+n}$$

Où r_i (resp. r_l) la requête qui précède directement l'opération d'embarquement (resp. débarquement) de la requête r_j. Rappelons qu'il est interdit de servir en avance (avant e_j) le point d'embarquement de la requête r_j : si nécessaire le véhicule k doit attendre.

Bien évidemment le problème contient les contraintes classiques du problème général de collecte et livraison (PDP) qui sont :

- Couplage : pour chaque requête r_j, le point d'embarquement et de débarquement doivent être visités dans la même tournée,
- Précédence : pour chaque requête r_j, le point d'embarquement doit être servi avant le point de débarquement,
- Capacité : le nombre de personne à bord de chaque véhicule k ne doit pas dépasser sa capacité Q_k.

Étant donné que souvent le transport à la demande est mis en place sur une zone géographique limitée (une commune ou un ensemble de communes), la contrainte de la durée maximale d'une tournée n'est pas considérée dans notre problème. De même les contraintes liées aux chauffeurs ne sont pas prises en compte lors de la résolution du problème, du moment que le changement de chauffeurs peut être fait à n'importe quel endroit et pas obligatoirement au dépôt (les chauffeurs se déplacent), comme c'est la cas dans le transport cadencé (au moins dans la ville de Tours).

La Figure 2.4.1.6 montre cinq tournées qui sont programmées pour des requêtes. Les véhicules partent toujours du dépôt et reviennent toujours au même dépôt à la fin de leur tournée. Les points figurants sur le schéma représentent des arrêts effectués par les véhicules. Un arrêt peut signifier soit un point d'embarquement pour un passager ou l'ensemble des passagers d'une même requête, soit un point de débarquement. Normalement à chaque arrêt est associé une fenêtre de temps durant laquelle le passager souhaite être servi.

2.4.2 la version dynamique du transport à la demande

Un système de transport à la demande dynamique a pour caractéristique de permettre aux passagers de demander le service de transport en ligne (en "temps réel"), c'est-à-dire que l'on permet à des clients d'appeler dans la journée pour demander une prise en charge le jour même. Pour cela, le système mis en place doit être capable de répondre à deux questions lors de la soumission d'une requête :

- La requête est-elle acceptable ? C'est-à-dire, existe-t-il une solution à ce nouveau problème qui permettra de transporter la personne ? En pratique, nous cherchons une solution faisable en utilisant des méthodes heuristiques rapides.

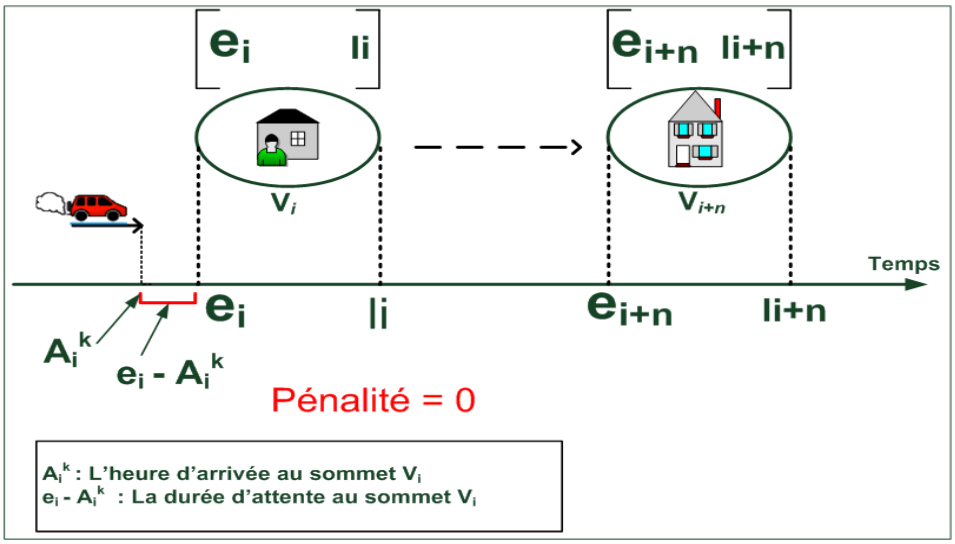


FIGURE 2.14 – Les fenêtre de temps

- Quel est le nouveau plan des tournées ? Il faut ensuite trouver la meilleure solution possible de réorganisation des tournées en prenant en compte cette nouvelle requête. Cette phase de post optimisation nécessite plus de temps d'exécution. Plusieurs scénarii peuvent être adoptés pour re-planifier les tournées : le premier scénario consiste à réoptimiser les tournées à l'apparition de chaque nouvelle requête. Tandis que dans le deuxième scénario il s'agit de définir une fréquence horaire (par exemple chaque 20 minutes) et lancer la phase de replanification si une ou plusieurs requêtes ont été acceptées entre temps.

Le problème consiste alors à insérer une nouvelle requête entrante (qui arrive en temps réel) à un instant t dans les tournées déjà existantes. Cela peut être fait de plusieurs manières différentes :

- La requête est insérée directement dans une tournée existante entre deux arrêts existants, plusieurs questions se posent alors : quelle tournée choisir ? Quelle position choisir ? Et selon quel critère ?,
- Une tournée est réorganisée pour y insérer la nouvelle requête, le choix de la tournée doit être fait sur quel critère ?,
- Mettre en service un nouveau véhicule (qui est au dépôt) pour prendre en charge cette requête. Sur quelle base cette décision est-elle prise ?
- Les tournées sont toutes réorganisées pour insérer la nouvelle requête.

Cependant, il s'agit ici d'un problème dynamique. Si l'on envisage de modifier les tournées existantes, il faut prendre en compte un facteur important : l'heure de soumission de la requête. En effet, à un instant t, on ne peut envisager de modifier une partie de tournée qui a déjà été effectuée. Une phase de mise à jour des tournées est nécessaire avant de procéder à la résolution du problème, par exemple pour les requêtes dont le point de départ est déjà servi par un véhicule k : son point de destination étant servi par le même véhicule

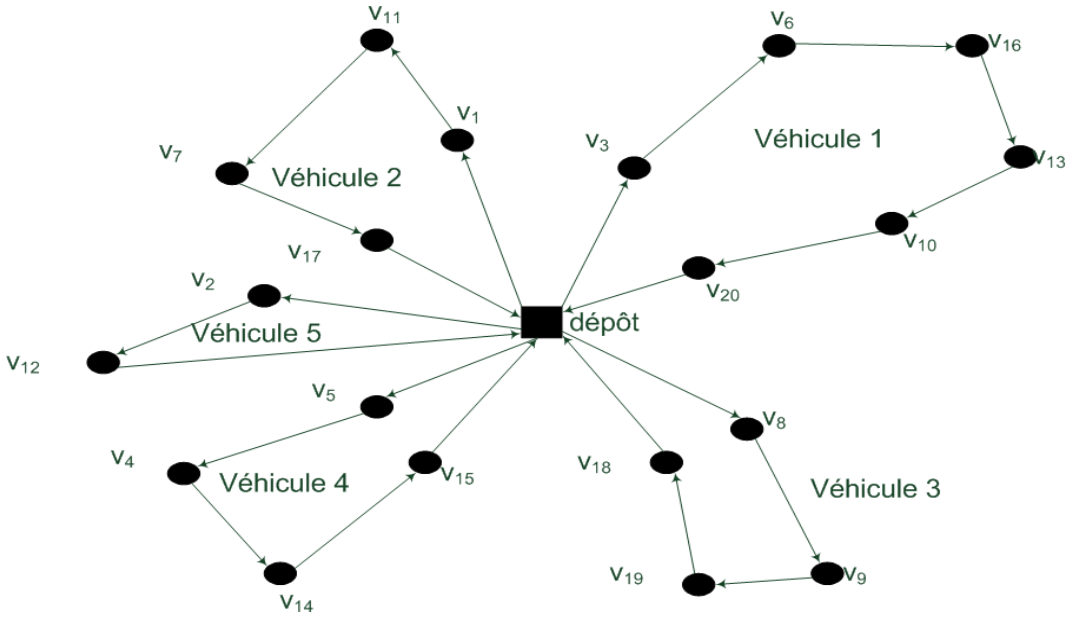


FIGURE 2.15 – Exemple d'une solution

k, on peut donc considérer que cette requête est pré-affectée définitivement au véhicule k. Les requêtes déjà servies peuvent être supprimées. Les requêtes pour lesquelles nous nous sommes déjà engagés sur une heure de prise en charge et une heure de dépose doivent aussi être respectées.

On peut donc diviser les sommets qui composent notre problème en quatre catégories :

- Le sommet appartient à une requête déjà effectuée (heure de départ et d'arrivée $< t$) : le sommet n'est plus pris en compte.
- Le sommet appartient à une requête en cours (heure de départ $< t$ et heure d'arrivée $> t$) : le sommet est pré-affecté à un véhicule et on ne peut pas modifier l'heure de passage qui lui est associé.
- Le sommet appartient à une requête programmée mais non démarrée (heure de départ et heure d'arrivée $> t$) : le sommet n'est pas affecté à un véhicule mais les fenêtres de passage doivent absolument être respectées.
- Le sommet appartient à une nouvelle requête : l'affectation a un véhicule est libre et les fenêtres de passage peuvent être éventuellement violées.

La Figure 2.16 montre l'exemple de 3 requêtes statiques déjà planifiées et une nouvelle requête dynamique à insérer.

Notons que durant la résolution des deux versions du transport à la demande, des critères secondaires ont été optimisées. Par exemple dans le DARP statique tricritère nous avons considéré le critère de minimisation de distance totale parcourue lors de l'insertion de nouvelles requêtes dans une tournée. Dans le cas du Transport à la Demande (la version dynamique) nous avons utilisé une métrique de Tchebechyv basée sur le critère de l'insatisfaction des passagers et le critère de la distance totale parcourue. Il est intéressant de signaler que lorsque on cherche à insérer une nouvelle requête dans une tournée donnée, seulement le critère de l'insatisfaction des passagers et de la distance totale parcourue sont

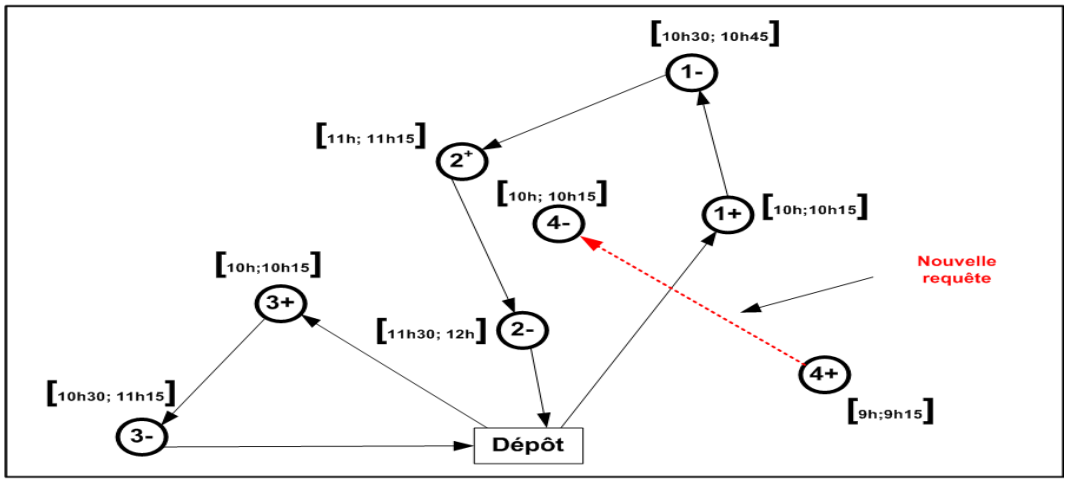


FIGURE 2.16 – 3 requêtes statiques déjà planifiées et une requête dynamique

prises en compte. Ceci est dû au fait que le critère du coût de transport et de l'émission de CO_2 sont exprimés en fonction de la distance parcourue multiplier par une constante. Donc la minimisation de ces deux critères revient à minimiser seulement le critère de la distance parcourue.

2.5 Conclusions

Ce chapitre est composé de deux parties : une première partie consacrée à l'état de l'art sur les problèmes de DARP et les méthodes d'optimisation multicritère et une deuxième partie consacrée à la présentation des deux problèmes que nous avons abordé dans cette thèse. Dans la première partie les travaux de recherche sur le DARP ont été classés selon deux versions : les DARP statiques et les DARP dynamiques. Les travaux de chacune de ces deux versions sont classés selon leur type de fonction objectif : On distingue trois catégories de problèmes. Des problèmes monocritères, multicritères avec une agrégation de critères et les multicritères avec l'optimisation simultanée de l'ensemble des critères. Il ressort de cet état de l'art que les problèmes de DARP multicritères avec optimisation simultanée de plusieurs critères n'ont pas reçu une attention importante de la part des chercheurs, ainsi nous avons pu ressortir que 4 travaux qui ont été publiés récemment dont deux en bicritères. La majorité des travaux qui ont été publiés concernent les DARPs monocritères ou multicritères avec une fonction d'agrégation de critères. Pour la version dynamique aucun travail n'a abordé le DARP dynamique avec une optimisation multicritère et la majorité des travaux publiés sont monocritères, ceci peut s'expliquer par le fait que les méthodes d'optimisation multicritères sont gourmandes en temps d'exécution ce qui ne répond pas aux exigences d'un problème dynamique qui nécessite une réponse rapide lors

de l'apparition de chaque nouvelle requête. Nous avons présenté aussi dans ce chapitre un synthèse des méthodes de résolution des problèmes d'optimisation multicritères. Quelques méthodes évolutionnaires multiobjectifs sont présentées. Dans la deuxième partie de ce chapitre, nous avons présenté la définition du problème de DARP statique multicritère que nous avons étudié dans le cadre de cette thèse. Ainsi, nous avons défini les données, les contraintes et les trois critères à minimiser. Nous avons présenté aussi dans ce même chapitre le problème de DARP dynamique que nous avons considéré dans cette thèse. Les méthodes de résolution de ce problème sont présentées dans le chapitre 5.

Chapitre 3

Le Problème de Timing dans une tournée

Le problème de *Timing* a été étudié dans la littérature sous plusieurs angles différents et sous différentes appellations. En général, ce problème consiste à trouver les dates d'exécution optimales d'une séquence d'activités prédéterminée, sous un ensemble de contraintes et d'objectifs à optimiser liés au temps. Quand il s'agit de calculer les dates de début optimales de service, on parle alors du problème d'*Optimal Timing*. Ce problème apparaît dans plusieurs domaines de planification comme la gestion de projet, l'ordonnancement de tâches, les tournées de véhicule, l'optimisation des réseaux (télécommunication, internet), etc. Dans ce qui suit nous utilisons le mot « activité » pour représenter une opération élémentaire qui doit être planifiée (indépendamment du domaine d'application), cette activité peut être une tâche, un travail, une visite d'un client dans le cas d'un problème de tournées de véhicules, etc. Chaque activité est associée à une durée d'exécution (une durée opératoire, en anglais *processing time*), une date de fin exigée ou souhaitée, une fonction de coût, etc. Souvent le problème de *Timing* apparaît comme un sous-problème dans plusieurs problèmes d'optimisation combinatoire qui sont largement étudiés dans la littérature. Par exemple dans les problèmes de collectes et livraisons avec fenêtres de temps (PDPTW), le problème de transport à la demande, etc.

3.1 Définition

Le DARP est souvent décomposé en deux sous-problèmes. Le premier sous-problème consiste à déterminer quels véhicules pour quelles requêtes (problème d'allocation des ressources) et le second sous-problème consiste à déterminer l'ordre dans lequel les véhicules vont visiter les clients qui leur ont été affectés (problème de séquencement). Une fois que de telles séquences sont fixées la résolution du problème n'est pas encore terminée, un autre sous-problème peut s'ajouter pour optimiser les heures de service de chaque requête. Ce dernier sous-problème est appelé « Timing », horodatage ou parfois scheduling. Il consiste à trouver les dates de début de service pour une séquence d'activités. On distingue deux types de problèmes : des problèmes avec une fonction objectif qui dépend des dates de début de service des activités et des problèmes avec une fonction objectif qui ne dépend pas

du temps. Dans quelques situations le problème de *Timing* est résolu même si la fonction objectif ne dépend pas du temps, dans ce cas l'objectif peut être d'étudier la faisabilité d'une solution en vérifiant les contraintes du problème qui sont liées au temps, de chercher un ordonnancement robuste qui peut absorber des aléas ou de laisser le plus de disponibilités possibles pour l'insertion des nouvelles requêtes comme c'est le cas dans le DARP dynamique ([Berbeglia *et al.*, 2010c]). Dans le cas de notre problème la fonction objectif de la qualité de service dépend de la date de début de service à chaque sommet.

Dans le cas du problème de transport à la demande que nous étudions dans cette thèse le sous-problème d'*Optimal Timing* (le problème est résolu d'une manière exacte) consiste à trouver pour chaque tournée les dates de début optimales de service à chaque sommet. L'objectif est de minimiser l'insatisfaction des personnes servies dans chaque tournée. C'est le critère que nous avons défini en détail dans le chapitre 2.

Nous limitons le problème d'*Optimal Timing* à la détermination des dates de début de service. Nous n'intégrons pas les heures de départ du véhicule aux arrêts, nous estimons que celles-ci se déduisent trivialement une fois les dates de début de service fixées. Nous définissons le problème d'*Optimal Timing* comme suit :

On considère une tournée $Vh_k = \{v_1, v_2, ...v_n\}$ associée à un véhicule k, la tournée est constituée d'une séquence fixée de n sommets. Les sommets $v_i, \forall i \in 1, ..., n$ correspondent aux points de départ et d'arrivée des requêtes, on suppose que n est toujours un nombre pair et que $n/2$ est le nombre de requêtes affectées au véhicule k. La séquence est supposée valide, c'est-à-dire que toutes les contraintes dures du problème sont respectées : contraintes de couplage, de précédence et de capacité du véhicule.

Chaque sommet v_i est caractérisé par une fenêtre de temps $[e_i, l_i]$, une pénalité unitaire d'avance α_i, une pénalité unitaire de retard β_i et un temps de service s_i, nous définissons aussi t_{ij} le temps nécessaire pour aller du sommet v_i au sommet v_j sur un trajet direct. On associe aussi à chaque requête r_i un poids ω_i lié aux temps d'attentes inutiles subis par les passagers de cette requête. Rappelons qu'on considère qu'on ne peut pas servir un sommet de départ v_i avant la borne inférieure e_i de sa fenêtre de temps. L'objectif du problème de Timing est de trouver les dates de début de service à chaque sommet qui minimisent l'insatisfaction des passagers, ce problème peut être formulé par le modèle suivant :

$$\begin{aligned} min & \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i) + \sum_{i=1}^{n-1} w_i (A_{i+1} - A_i - p_i) \\ s.c & \quad A_i + p_i \leq A_{i+1} \end{aligned}$$

Où A_i est la date effective de début de service au sommet situé en position i, $E_i = max(0, e_i - A_i)$, $T_i = max(0, A_i - l_i)$, $p_i = s_i + t_{i,i+1}$
et w_i représente la somme des mécontentements des personnes qui sont à bord du véhicule entre le sommet situé à la position i et le sommet situé à la position $i + 1$.

Si on envisage de résoudre le problème de DARP par une métaheuristique, le problème d'optimal timing devra être résolu pour chaque tournée de chaque solution étudiée. Nous avons donc besoin d'une méthode de résolution rapide du problème d'optimal timing.

Ce problème est relativement peu étudié dans la littérature sur le transport, mais on peut s'appuyer sur les études dans le cadre des systèmes de production.

Dans la suite nous présentons rapidement un état de l'art sur quelques problèmes de *Timing*.

3.2 État de l'art

Le problème de *Timing* se rencontre dans plusieurs problèmes de logistique et transports, tels des problèmes de type voyageur de commerce, calcul de plus court chemin avec contraintes de ressources ou tournées de véhicules. Ce problème apparaît souvent comme un sous-problème lors de l'étude de la faisabilité d'une tournée, ou pour calculer la somme des violations des fenêtres de temps des tournées générées lors de la recherche des solutions. L'aspect temporel devient de plus en plus important quand on traite des problèmes dynamiques et stochastiques dans lesquels plusieurs données dépendent du temps : la vitesse de voyage, l'apparition d'une nouvelle requête, le temps de travail des chauffeurs, les pauses obligatoires, etc.

Il est vrai que ce problème de *Timing* est souvent passé sous silence lors de la résolution des problèmes de tournées de véhicules, mais les chercheurs commencent à lui accorder de plus en plus d'importance, notamment dans les problèmes liés aux transport des personnes où la dimension temporelle est très importante.

Dans le cadre de la résolution d'un problème de DARP dynamique, [Berbeglia *et al.*, 2010c] ont proposé trois méthodes pour la résolution d'un problème particulier de *Timing*. Il consiste à calculer les dates de début de service appropriées à chaque sommet afin de maximiser le nombre d'acceptations de nouvelles requêtes qui arrivent au fil de temps, dans ce problème la fonction objectif ne dépend pas des dates de début de service aux sommets, la résolution du problème consiste seulement en une étude de faisabilité sur le respect des contraintes du problème, notamment des contraintes de fenêtre de temps et de temps passé à bord des véhicules.

Les problèmes de Timing peuvent être classifiés selon la fonction objectif utilisée (convexe, linéaire,....) ou selon le type des contraintes temporelles. Pour plus de détails sur la classification des problèmes de Timing ainsi que des méthodes de résolution, le lecteur peut se référer au travail de [Vidal *et al.*, 2011]. Parmi ces catégories de problèmes on distingue en particulier ceux avec une fonction objectif linéaire par morceaux et convexe. Ils sont largement étudiés dans la littérature et des algorithmes de complexité polynomiale ont été proposés pour les résoudre.

Toutefois on peut s'inspirer d'autres domaines de recherche, en effet le problème de calcul des heures de passages dans une tournée, peut être assimilé à un problème d'ordonnancement à une machine de type $1|d_i, seq|F_l(\bar{E}^\alpha, \bar{T}^\beta)$: 1 pour une machine, d_i pour les dates de réalisation au plus tard, *seq* (séquence) pour le type de graphe représentant les contraintes de précédence. $F_l(\bar{E}^\alpha, \bar{T}^\beta)$ est une fonction qui représente la somme pondérée des pénalités des retards et avances à chaque tâche. Ce genre de problème fait partie des problèmes d'ordonnancement de type Juste-à-Temps qu'on appelle également les problèmes

avec pénalités d'avance et de retard. Dans la suite nous présentons une rapide revue de la littérature sur les problèmes d'ordonnancement de type avance/retard qui ressemble au problème de *Timing* que nous abordons dans cette thèse.

Les problèmes d'ordonnancement de type avance/retard dans lesquels la fonction objectif est une combinaison linéaire de l'avance totale pondérée E^α et du retard total pondéré T^β est un modèle particulier de l'ordonnancement Juste-à-Temps qui est largement étudié dans la littérature, les lecteurs intéressés peuvent se référer aux travaux de [Gordon *et al.*, 2002] et [Kaminsky *et al.*, 2004] qui présentent une synthèse de l'état de l'art sur les problèmes d'ordonnancement en Juste-à-Temps.

Le cas des fonctions objectifs exprimables comme combinaison linéaire de fonctions de retard et d'avance de la forme $\bar{E}^\alpha + \bar{T}^\beta = \sum_i \alpha_i E_i + \sum_i \beta_i T_i$ représente la quasi-totalité de la littérature [Desrosiers *et al.*, 1995]. On distingue plusieurs catégories de sous-problèmes en fonction des valeurs prises par les poids α_i et β_i :

1. Les poids sont asymétriques, i.e. $\exists i$, i=1,..,n, tel que $\alpha_i \neq \beta_i$,
2. Les poids sont symétriques, i.e. $\forall i$, i=1,..,n, on a $\alpha_i = \beta_i$,
3. Les poids sont indépendants des travaux, i.e. $\exists \alpha_0, \beta_0$ tels que $\forall i$, i=1,..,n, on a $\alpha_i = \alpha_0$ et $\beta_i = \beta_0$
4. Les poids sont quelconques.

3.2.1 Le problème de minimisation de retards et d'avances pondérés dans un problème à une machine

Les travaux de [Garey *et al.*, 1988] portent principalement sur le problème $1|d_i|F_l(\bar{E}, \bar{T})$, avec $F_l(\bar{E}, \bar{T}) = \bar{E} + \bar{T}$. On a donc $\alpha_i = \beta_i = 1$, $\forall i \in \{1, ..., n\}$ où n est le nombre d'activités. Ils s'intéressent alors à la résolution de ce problème quand la séquence des activités est fixée, ce problème se notant $1|d_i, seq|F_l(\bar{E}, \bar{T})$, et proposent un algorithme optimal dont la complexité est en $O(n \log(n))$ (voir Figure 3.2).

Les activités sont placées itérativement selon l'ordre imposé. On note σ_{i-1} la séquence partielle obtenue à l'itération $i-1$. Soit J_i l'activité à insérer à l'itération i. Si $C_{max}(\sigma_{i-1}) + p_i \leq d_i$ alors l'activité J_i débute à la date $d_i - p_i$ et un temps mort est inséré avant J_i. Dans le cas contraire, on a $C_{max}(\sigma_{i-1}) + p_i > d_i$ et l'activité J_i est en conséquence traitée immédiatement à la fin de la séquence σ_{i-1}. Dans ce second cas, il va ensuite peut être falloir décaler vers la gauche une partie de la nouvelle séquence $\sigma_i = \sigma_{i-1}|J_i$ afin de réduire la valeur du coût de cette séquence.

La nécessité de ce décalage s'explique par les raisons suivantes : si on définit un bloc par un ensemble maximum d'activités consécutives exécutées sans temps mort, et qu'on note B_k le dernier bloc de σ_i et si la majorité des activités dans B_k sont en retard alors il est intéressant de décaler B_k vers la gauche jusqu'à ce que cela ne soit plus le cas ou jusqu'à ce que B_k ne puisse plus être décalé. La Figure 3.1 présente un exemple d'ordonnancement de 6 activités en 6 étapes. Dans cet exemple, on suppose que les tâches 4 et 6 ont des poids β_i, associés aux retards, importants.

Dans [Hoogeveen et Velde, 1996] les auteurs montrent que le même algorithme fonctionne dans le cas $\alpha_i = \alpha$ et $\beta_i = \beta$, $\forall i \in 1, ..., n$ (c'est le cas où les poids sont indépendants

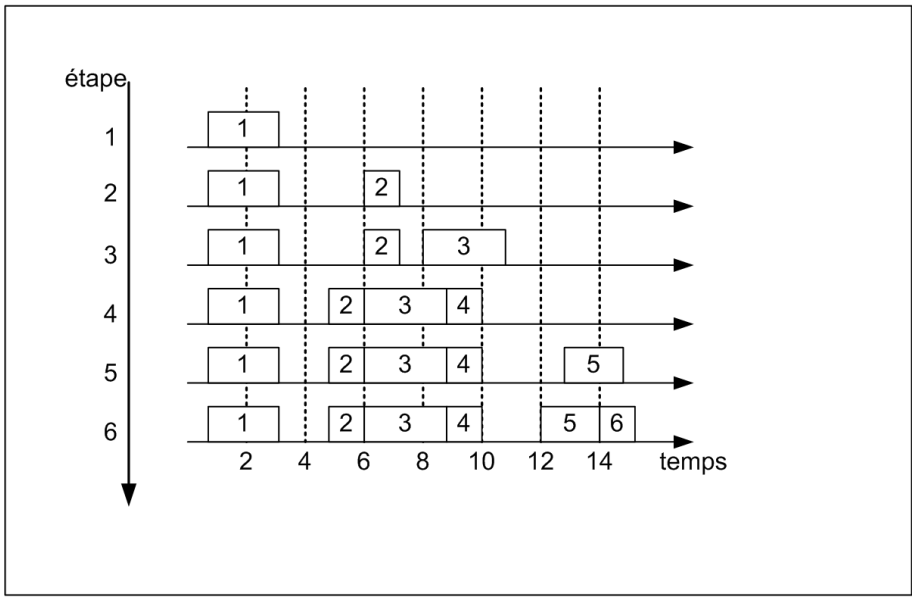


FIGURE 3.1 – Un ensemble d'étapes observables lors de l'application de l'algorithme proposé dans [Garey *et al.*, 1988]

Algorithme pour le calcul des bornes inférieures des dates de début de service

/* $\sigma = \emptyset$: l'ordonnancement en cours
/* on suppose que la séquence imposée est $(J_1, J_2, ..., J_n)$
Pour i=1 à n faire :
  Si $(d_i - p_i > C_{max}(\sigma))$ Alors
    Ordonnancer l'activité J_i au temps $t_i = C_{max}(\sigma)$
    $\sigma = \sigma | J_i$
  Sinon
    Ordonnancer l'activité J_i au temps $t_i = C_{max}(\sigma)$;
    $\sigma = \sigma | J_i$
    Soit B_k le dernier bloc de σ ;
    Soit r le nombre d'activités $J_i \in B_k / C_i - d_i > 0$;
    Si(r>($|B_k| - r$))Alors
      Soit $\delta = \min_{J_i \in B_k / C_i > d_i} (C_i - d_i)$;
      Soit ω la date de début de la première activité de B_k ;
      Soit π la date de fin de la dernière activité de B_{k-1} ;
      $\forall j \in B_k, t_j = t_j - min(\delta; \omega - \pi)$;
    FinSi
  FinSi
FinPour

FIGURE 3.2 – Algorithme d'optimal timing proposé par [Garey *et al.*, 1988]

des travaux).

3.2.2 Le problème $P\infty|prec, f_i\ convex|\sum_i f_i$

Ce problème d'ordonnancement général a été abordé par Chrétienne et Sourd dans [Chrétienne et Sourd, 2003]. Ils présentent tout d'abord des résultats théoriques et un al-

gorithme général d'ordonnancement d'un ensemble de travaux, où une fonction de coût convexe est associée à chaque travail. Notons que chaque travail J_i est défini par une date de début souhaitée φ_i, un temps opératoire p_i et une fonction convexe de coût $f_i(t)$. L'algorithme général proposé par Chrétienne et Sourd peut être implémenté en $O(n \max(n, M))$ où M est le nombre de contraintes de précédence. Lorsque le graphe de précédence a une structure d'arbre, la complexité de l'algorithme devient $O(n^2)$.

L'algorithme général est particularisé à différents cas spéciaux comme par exemple le problème d'ordonnancement avec des fonctions de coûts f_i linéaires par morceaux et reflétant des coûts d'avance/retard. Les Figures 3.3 et 3.4 illustrent une fonction de ce type. La Figure 3.3 représente le cas où le travail est en avance, la Figure 3.4 le cas où le travail est en retard.

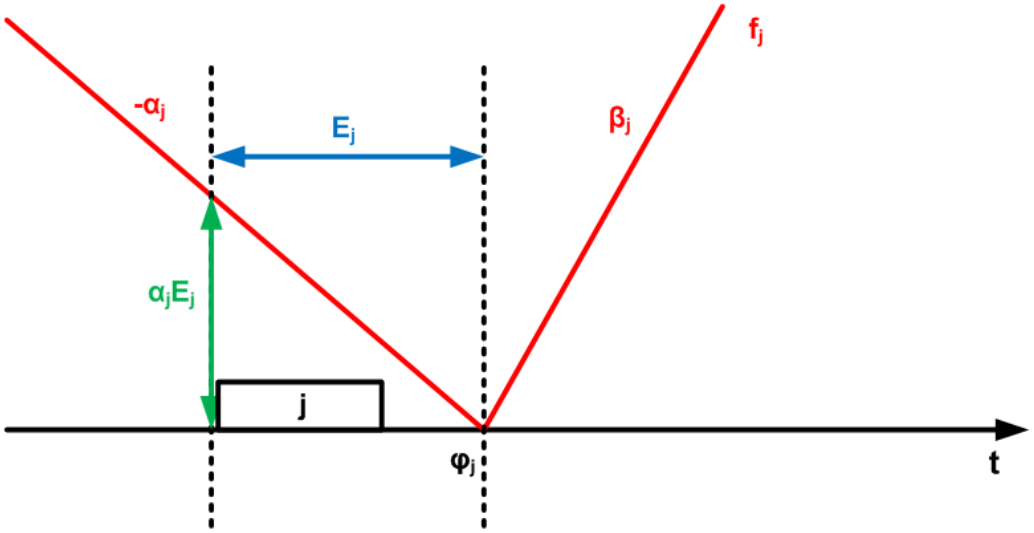


FIGURE 3.3 – Coût d'une activité j en avance

Dans le cas de contraintes de précédence de type chaine, c'est-à-dire quand un travail a au plus un successeur et un prédécesseur, Chrétienne et Sourd ([Chrétienne et Sourd, 2003]) ont proposé un algorithme de complexité temporelle réduite à $O(n \log(n))$ et qui est inspiré de l'algorithme proposé dans [Chrétienne, 2001] pour les problèmes avec des poids d'avance et de retard asymétriques et dépendants des travaux. Les auteurs commencent par remarquer que cela revient à résoudre un ensemble de problèmes indépendants à une machine à séquence fixée, problème qui peut se noter $1|d_i, seq|F_l(\bar{E}^{\alpha}, \bar{T}^{\beta})$. La différence avec les travaux [Hoogeveen et Velde, 1996] réside dans le fait que les poids peuvent être asymétriques.

Pour résoudre ce problème, Chrétienne et Sourd utilisent la notion de bloc qui est défini comme étant une séquence de travaux s'exécutant sans temps mort. Chaque bloc est défini par sa date de début t_B, une pente l_B, sa durée P_B qui est égale à la somme des durées des travaux qui composent le bloc et par le nom de son travail de tête. Sa pente, l_B, peut être interprétée comme étant la contribution unitaire au coût total des travaux dans B. Par exemple, si B est constitué d'un travail J_i en avance et d'un travail J_j en retard, nous avons $l_B = -\alpha_i + \beta_j$. Une conséquence est que les dates de début à considérer pour un

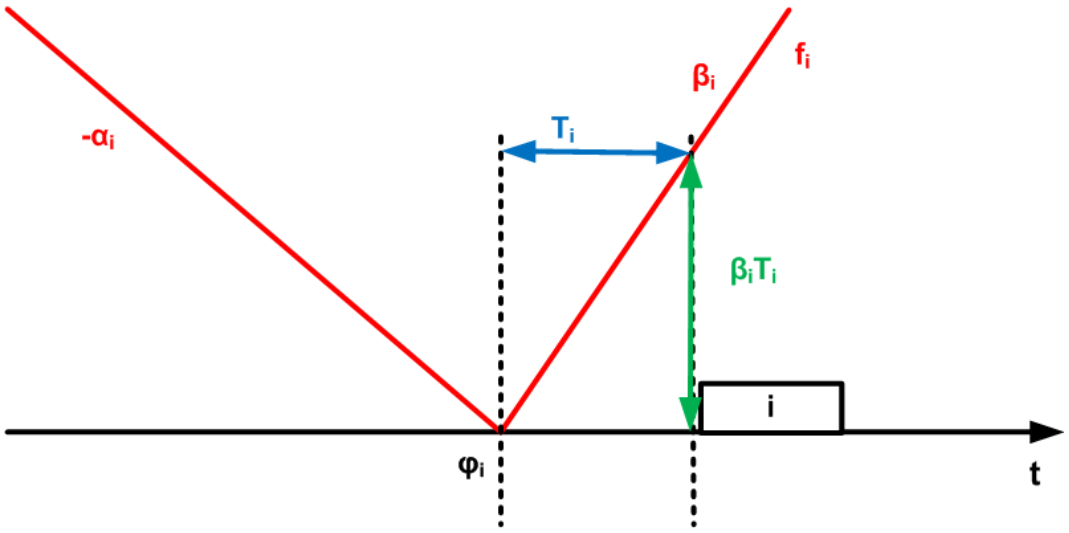


FIGURE 3.4 – Coût d'une activité i en retard

bloc sont les points de rupture de la fonction de coût du bloc.

Notons que les fonctions de coût f_i sont linéaires par morceaux, chacune admettant un point de rupture unique. Ainsi, on définit pour chaque bloc B un ensemble de points singuliers t_j^B, $\forall J_j \in B$. Chaque point singulier correspond à un changement dans la pente de la fonction de coût d'au moins un travail du bloc B.

Si on connait la date de début de chaque bloc et le premier travail de chaque bloc, on peut déduire les dates de début de tous les travaux.

Lors de la construction de chaque bloc, on préserve les informations suivantes : la pente du bloc l_B, la date de début du bloc t_B, le premier travail du bloc $f(B)$ et ϵ_B le temps mort entre t_B et la date de fin du bloc précédent. On note le dernier travail du bloc qui précède B par $f(B) - 1$. Une structure de donnée de type *tas* est utilisée pour stocker les travaux qui sont en retard. L'intérêt de l'utilisation de cette structure de donnée est que l'insertion, la suppression et la fusion de deux tas peuvent se faire avec une complexité $O(\log(n))$. Chaque travail J_j de ce tas est caractérisé par le grandeur $\sum_{i<j} p_i - \varphi_j$. On suppose, sans perdre de généralité que les travaux sont numérotés suivant leur ordre d'apparition dans la séquence.

Principes généraux de l'algorithme :

Fusion de deux blocs : si deux blocs B_1 et B_2 se suivent dans un ordonnancement et si en décalant le bloc B_2 vers la gauche on vient rencontrer B_1, il faut fusionner les deux blocs. On crée ainsi un bloc $B = B_1 \cup B_2$. Les caractéristiques de ce bloc B sont : $t_B = t_{B_1}$, $l_B = l_{B_1} + l_{B_2}$, $f_B = f_{B_1}$, $\epsilon_B = \epsilon_{B_1}$ et le tas associé au bloc B est obtenu en fusionnant les deux tas associés aux blocs B_1 et B_2.

décalage d'un bloc vers la gauche : on travaille sur le dernier bloc B de l'ordonnancement courant. Si la pente l_B, correspondant à la date de début t_B du bloc, est positive cela signifie qu'en avançant B (autrement dit en le décalant vers la gauche) on va diminuer

son coût ($f_B(t_B)$). Il faut se diriger vers le prochain point de rupture qui est donné par le plus petit élément du tas associé à B. Si ce plus petit élément correspond à un travail J_j alors la nouvelle date de début du bloc est $t'_B = \varphi_j - \sum_{i<j} p_i + \sum_{i<f_B} p_i$. Si t'_B est plus petite que la date de fin du bloc précédent alors on fusionne ces deux blocs sinon on met à jour le bloc B de la façon suivante : $\epsilon_B = \epsilon_B - t_B + t'_B$, $t_B = t'_B$, $l_B = l_B - (\alpha_j + \beta_j)$ et on supprime le travail J_j du tas.

L'algorithme s'exécute en ajoutant les travaux les uns après les autres dans l'ordre de la séquence *seq*. Le principe de l'algorithme est de regarder la pente de la fonction de coût associée au bloc à chaque point de rupture pour savoir s'il est possible de décaler le bloc à gauche. Si la pente à gauche du point de rupture est positive alors il est souhaitable d'avancer le bloc pour diminuer le coût. La pente du bloc est mise à jour à chaque décalage jusqu'à arriver à la date minimale de début du bloc ou jusqu'à rencontrer un autre bloc.

La Figure 3.5 montre un exemple de deux travaux qui forme un bloc B. Un travail J_i placé à sa date de début idéale φ_i et un travail J_j en retard par rapport à sa date de début idéale φ_j. La fonction de coût associée à ce bloc est définie par $f_B(t_B) = f_i(t_B) + f_j(t_B + p_i)$. Cette fonction admet le point $\varphi_j - p_i$ comme point de rupture auquel le bloc peut être décalé si le coût du bloc diminue et aucun autre bloc n'est placé entre $\varphi_j - p_i$ et la date de début du bloc courant.

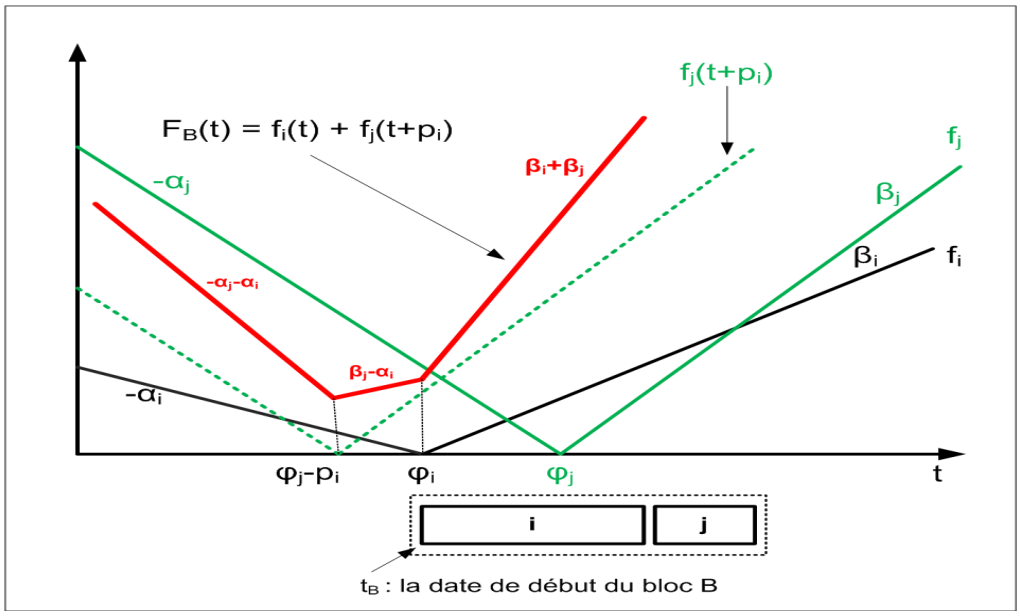


FIGURE 3.5 – Exemple de la fonction de coût associée à un bloc B constitué par deux travaux

3.3 Méthode de résolution du problème d'Optimal Timing dans le DARP

Le sous-problème d'Optimal Timing que nous étudions dans cette thèse consiste à calculer les dates de début optimales de service de n sommets d'une tournée $Vh_k = \{v_1, ..., v_n\}$ associée à un véhicule k. on suppose dans cette partie que les sommets sont numérotés dans l'ordre où le véhicule les visite. On définit A_i l'heure de début de service au sommet v_i, $\forall i \in \{1, ..., n\}$. L'objectif est de minimiser l'insatisfaction des passagers en termes de respect des contraintes de fenêtres de temps et de durée d'attente à bord du véhicule. Le graphe de précédence dans une tournée est formé par une séquence fixée de sommets. Ce problème ressemble au problème $1|chain, f_i\ convex|\sum_i f_i$ traité dans [Chrétienne et Sourd, 2003]. Il reste à vérifier que la fonction objectif, qui modélise l'insatisfaction des passagers, est constituée par une somme de fonctions convexes et linéaires par morceaux.

3.3.1 La fonction de coût

Dans la suite nous nous focalisons sur la fonction $\overline{QoS_k}$ qu'on remplace par une fonction générale F exprimée en fonction des dates de début de service à chaque sommet de la tournée.

$$F(A_1, A_2, ..., A_n) = \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i) + \sum_{i=1}^{n-1} w_i (A_{i+1} - A_i - p_i)$$

Où $[e_i, l_i]$ est la fenêtre de temps associée au sommet v_i, $E_i = max(0, e_i - A_i)$ est l'avance du véhicule au sommet v_i par rapport à la borne inférieure e_i (voir le chapitre 2), $T_i = max(0, A_i - l_i)$ est le retard du véhicule par rapport à la borne supérieure l_i et $p_i = s_i + t_{i,i+1}$ est le temps nécessaire pour servir le sommet v_i et se déplacer vers le sommet v_{i+1} ($t_{i,i+1}$ est la durée du trajet direct entre v_i et v_{i+1}). Les sommets v_0 et v_{n+1} représentent le dépôt.

Lemme 3.3.1 *Minimiser la fonction F sous les contraintes de précédences $v_1 \rightarrow .... \rightarrow v_n$ est équivalent à minimiser $F^T(A_1, ..., A_n) = \sum_{i=1}^{n} f_i^{cvx}(A_i)$ avec $f_i^{cvx}(A_i) = \alpha_i E_i + \beta_i T_i + A_i(w_{i-1} - w_i)$ et $w_0 = 0$.*

Preuve. nous avons :
$F(A_1, ..., A_n) = \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i) + \sum_{i=1}^{n-1} g_i(A_i, A_{i+1})$
avec $g_i(A_i, A_{i+1}) = w_i(A_{i+1} - A_i - p_i), \forall i = 1, ..., n-1$.
or,

$$\begin{aligned}\sum_{i=1}^{n-1} g_i(A_i, A_{i+1}) =& (w_1 A_2 - w_1 A_1 - w_1 p_1) + (w_2 A_3 - w_2 A_2 - w_2 p_2) + ... + w_{n-1} A_n - w_{n-1} A_{n-1} - \\ & w_{n-1} p_{n-1} \\ =& A_1(w_0 - w_1) + A_2(w_1 - w_2) + ... + A_n(w_{n-1} - w_n) - \sum_{i=1}^{n-1} w_i p_i \\ =& \sum_{i=1}^{n} A_i(w_{i-1} - w_i) - \sum_{i=1}^{n-1} w_i p_i\end{aligned}$$

avec $w_0 = w_n = 0$ et $\sum_{i=1}^{n-1} w_i p_i$ est constante puisque la séquence de la tournée est fixée.

On définit $f_i^{cvx}(A_i) = \alpha_i E_i + \beta_i T_i + A_i(w_{i-1} - w_i)$, donc minimiser $F(A_1, ..., A_n)$ est équivalent à minimiser : $F^T(A_1, A_2, ..., A_n) = \sum_{i=1}^{n} f_i^{cvx}(A_i)$. □

Minimiser la fonction F^T sous les contraintes de précédence $v_1 \rightarrow .... \rightarrow v_n$, ressemble à un cas particulier du problème d'Optimal Timing traité par Chrétienne et Sourd ([Chrétienne et Sourd, 2003]). Afin d'adapter l'algorithme général proposé dans [Chrétienne et Sourd, 2003] pour résoudre ce type de problèmes, nous devons vérifier que les fonctions f_i^{cvx} sont convexes et linéaires par morceaux.

C'est bien le cas : la fonction f_i^{cvx} est convexe et linéaire par morceaux en tant que somme de trois fonctions convexes et linéaires par morceaux.

Nous précisons ci-dessous les pentes des trois morceaux qui composent la fonction f_i^{cvx}.

Nous avons :

$$f_i^{cvx}(A_i) = \begin{cases} \lambda_i A_i + \alpha_i e_i, \text{ avec } \lambda_i = w_{i-1} - w_i - \alpha_i, & \text{si } A_i \leq e_i \\ \lambda_i A_i, \text{ avec } \lambda_i = w_{i-1} - w_i, & \text{si } A_i \in]e_i, l_i] \\ \lambda_i A_i - \beta_i l_i, \text{ avec } \lambda_i = w_{i-1} - w_i + \beta_i, & \text{si } A_i > l_i \end{cases}$$

avec λ_i la pente de la fonction f_i^{cvx} et α_i, β_i, e_i, w_i et l_i des valeurs positives.

La table (3.3.1) et la figure (3.6) présentent tous les cas possibles de la pente λ_i qui dépend de la valeur de A_i, e_i et l_i. Dans la suite on remplace $w_{i-1} - w_i$ par Δw_i. On constate que $\Delta w_i - \alpha_i \leq \Delta w_i \leq \Delta w_i + \beta_i$ ce qui confirme la convexité de f_i^{cvx}.

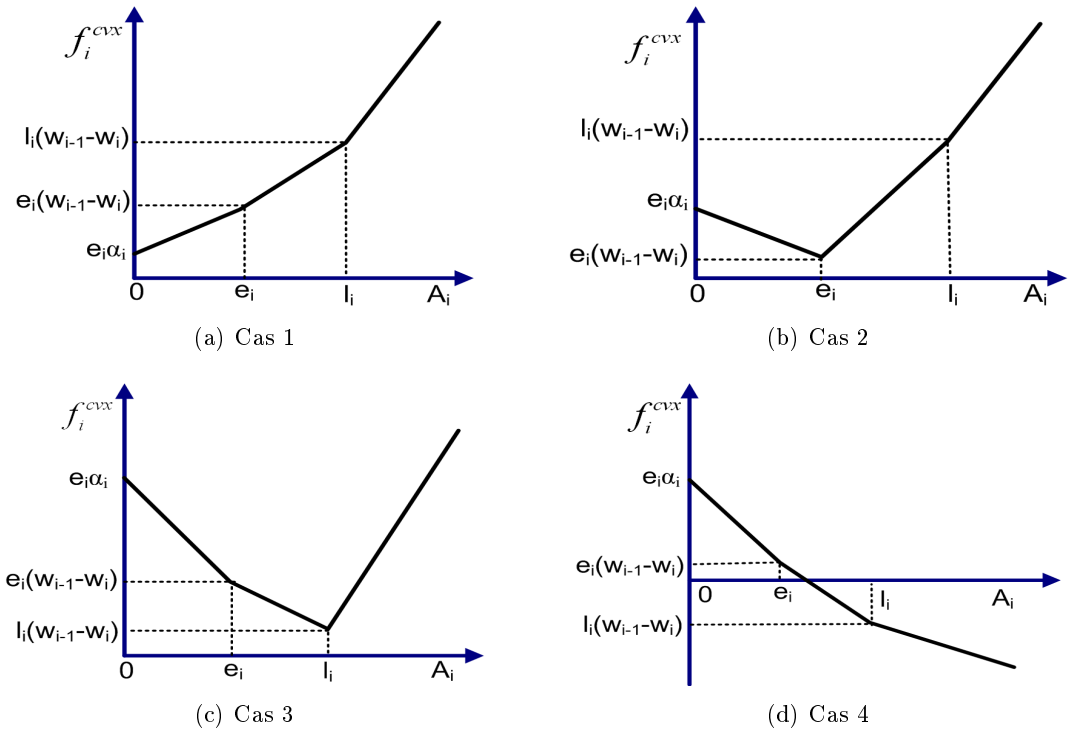
pentes	$\Delta w_i - \alpha_i$	Δw_i	$\Delta w_i + \beta_i$
Cas 1	≥ 0	≥ 0	≥ 0
Cas 2	< 0	≥ 0	≥ 0
Cas 3	< 0	< 0	≥ 0
Cas 4	< 0	< 0	< 0

TABLE 3.1 – Evolutions possibles des pentes λ_i

3.3.2 Algorithme de résolution

L'idée centrale de l'algorithme proposé par Chrétienne et Sourd dans [Chrétienne et Sourd, 2003] est l'utilisation de la notion de *bloc* d'activités. Dans le cas de notre problème, une activité est une opération d'embarquement ou de débarquement, donc un bloc est constitué par une succession d'opérations d'embarquement et de débarquement qui sont exécutées sans temps d'attente.

Le problème d'Optimal Timing que nous cherchons à résoudre dans cette thèse présente quelques différences par rapport au problème étudié dans [Chrétienne et Sourd, 2003], la différence est liée au nombre de points de rupture considérés qui est égal à n (n le nombre



(a) Cas 1 (b) Cas 2

(c) Cas 3 (d) Cas 4

FIGURE 3.6 – Allures générales des fonctions de coût associées aux sommets v_i

d'activités) dans [Chrétienne et Sourd, 2003] et à $2n$ dans le cas de notre problème. Les fenêtres de temps associées aux opérations d'embarquement et de débarquement ne sont pas toutes de la même nature. On distingue deux types de fenêtres de temps : des fenêtres souples associées aux sommets de débarquement pour lesquelles l'avance et le retard sont autorisés et des fenêtres semi-souples associées aux sommets d'embarquement pour lesquelles seulement le retard est autorisé. Ces différences peuvent être prises en compte dans l'algorithme proposé par [Chrétienne et Sourd, 2003] après de légères modifications : cela implique que certains blocs ne pourront pas être décalés vers la gauche au delà d'une certaine limite. Dans la suite nous présentons les détails de l'algorithme que nous avons utilisé pour résoudre notre problème. La Figure 3.7 montre l'exemple d'une fonction du coût associée à un sommet de départ v_i et la Figure 3.8 l'exemple d'une fonction de coût associée à un sommet d'arrivée v_j.

Résoudre le problème d'Optimal Timing revient à identifier les blocs et leurs dates optimales de début de service. La fonction de coût associée à un bloc B est définie par la formule $F_B(A_B) = \sum_{i=1}^{|B|} f_i^{cvx}(A_i)$, où A_B est la date de début du bloc B et les A_i sont les dates de début de service des sommets composant le bloc B. La fonction F_B est convexe et linéaire par morceaux et les dates A_i peuvent se calculer à partir de la date A_B et des durées $p_i = s_i + t_{i,i+1}$. Chaque opération d'embarquement ou de débarquement à un sommet v_i est associée à une fonction de coût $f_i^{cvx}(A_i) = \alpha_i E_i + \beta_i T_i + A_i(w_{i-1} - w_i)$ qui admet deux points singuliers (points de ruptures), donc chaque bloc B_b (b l'indice du

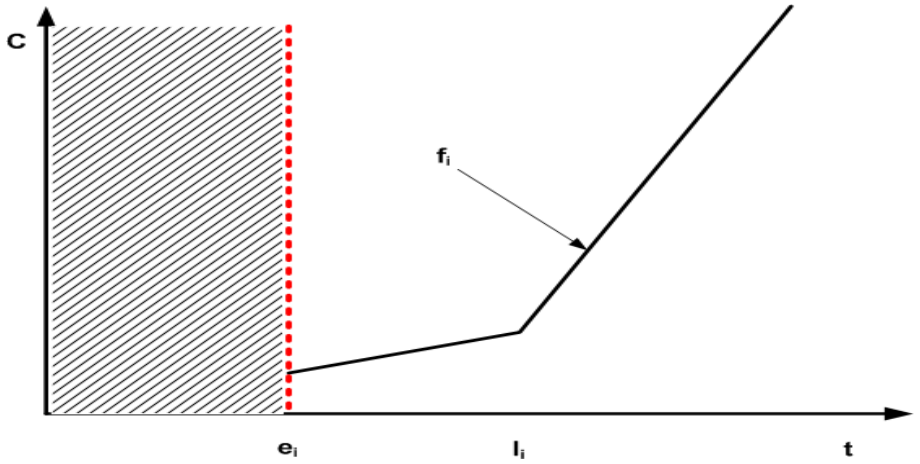


FIGURE 3.7 – La fonction du coût f_i associée à un sommet d'embarquement v_i

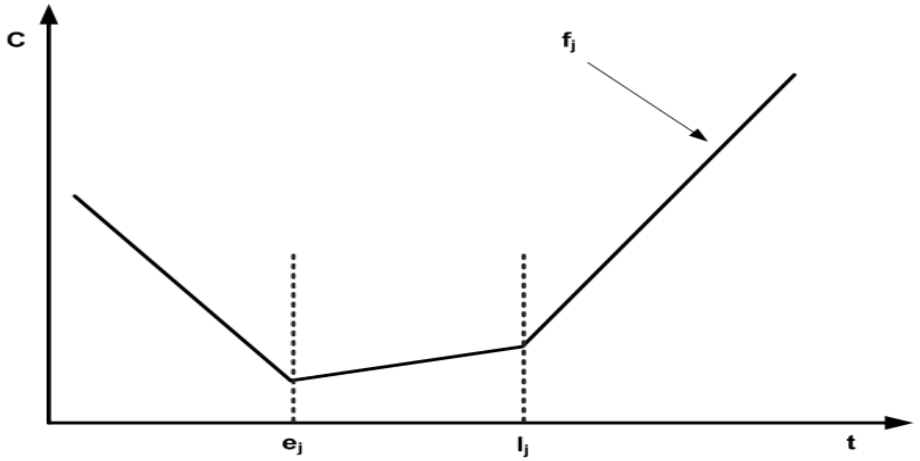


FIGURE 3.8 – La fonction du coût f_j associée à un sommet de débarquement v_j

bloc) admet au plus $2|B_b|$ points ruptures. L'ensemble des points de ruptures du bloc B_b est noté R_b, chaque point de R_b peut être mis en correspondance avec un changement de pente de la fonction de coût f_i^{cvx} d'au moins un sommet v_i du bloc B_b.

Notons ici aussi l_{B_b} la pente liée au bloc B_b qui peut être interprétée comme étant la contribution unitaire au coût total du bloc B_b. Par exemple, si le bloc B_b contient deux sommets, un sommet v_i en avance et un sommet v_{i+1} en retard alors $l_{B_b} = (w_i - w_{i+1} + \beta_{i+1}) + (w_{i-1} - w_i - \alpha_i)$. La date optimale de début de service d'un bloc est le point de rupture qui conduit vers un coût minimal. Un bloc B_b est en retard si la pente relative à sa fonction de coût F_{B_b} est positive et quand le bloc est en avance sa pente est négative.

Si un bloc B_b est en retard (la date de début de service du bloc n'est pas optimale) alors il faut le décaler vers la gauche d'un point de rupture vers un autre pour trouver la date de début optimale du bloc. Notons que si ce bloc contient un sommet d'embarquement v_i, ce sommet doit respecter la date de début minimale e_i, donc le bloc devra également respecter une date de début minimale $A_{B_b}^{min}$ dont la formule de calcul est précisée plus bas

dans le texte. Le décalage vers la gauche du bloc est réitéré tant que la date minimale de début du bloc n'est pas atteinte ou qu'on ne rencontre pas un autre bloc (la date de début du bloc arrive sur un autre bloc).

Dans le cas de deux blocs conflictuels (un bloc qui arrive sur un autre), les deux blocs vont être fusionnés en un seul bloc qui sera décalé de nouveau vers la gauche s'il est en retard. Il faut noter que durant l'exécution de l'algorithme, la date de début de service à chaque sommet de la séquence ne peut que décroître et que chaque bloc construit ne peut pas être divisé en deux sous-blocs, il ne peut qu'être fusionné avec d'autres blocs.

Notations utilisées dans l'algorithme

Dans la suite nous utilisons les notations suivantes :

- A_i^{min} : la date minimale de début service au sommet situé à la position i dans la séquence des sommets à ordonnancer (la tournée),
- A_i^{max} : la date maximale de début service au sommet situé à la position i dans la séquence des sommets à ordonnancer,
- p_i : la durée opératoire liée au sommet situé à la position i, $p_i = s_i + t_{i,i+1}$,
- $l_i^g(t)$: la pente de la fonction f_i^{cvx} à gauche du point t, où $t \in \left\{A_i^{min}, e_i, l_i, A_i^{max}\right\}$,
- $l_i^d(t)$: la pente de la fonction f_i^{cvx} à droite du point t, où $t \in \left\{A_i^{min}, e_i, l_i, A_i^{max}\right\}$,
- p_{B_b} : la durée opératoire (processing time) du bloc B_b, $p_{B_b} = \sum_{v_i \in B_b} p_i$,
- A_{B_b} : la date de début de service du bloc B_b,
- $A_{B_b}^{end}$: la date de fin du service du bloc B_b avec $A_{B_b}^{end} = A_{B_b} + p_{B_b}$,
- $A_{B_b}^{min}$: la date minimale de début de service du bloc, $A_{B_b}^{min} = \max_{v_i \in B_b} \left(A_i^{min} - \sum_{j=0 v_j \in B_b}^{i-1} p_j \right)$, avec $p_0 = 0$,
- R_b : une structure de données de type *tas* qui va stocker les points de ruptures liés au bloc B_b. A chaque sommet v_i du bloc B_b correspond trois points de ruptures dans le tas définis par les valeurs : $A_i^{min} - \sum_{j<i} p_j$, $e_i - \sum_{j<i} p_j$ et $l_i - \sum_{j<i} p_j$. Ces grandeurs sont triées selon un ordre partiel décroissant (le premier élément du tas correspond à la plus grande valeur). On associe également à chaque point de rupture t les pentes $l^d(t)$ observée à droite et $l^g(t)$ observée à gauche sur la fonction de coût du sommet correspondant. Il faut noter que la pente à gauche associée à A_i^{min} est égale à $-\infty$,
- BL : la liste des blocs, qui va contenir l'ensemble des blocs qui seront formés durant l'exécution de l'algorithme d'Optimal Timing. Elle contient $|BL|$ blocs : $BL[|BL|]$ se réfère au dernier bloc ajouté dans la liste BL et $BL[1]$ au premier bloc.

Calcul des bornes inférieures et supérieures des dates de début de service de chaque sommet d'une tournée

Soit Vh_k une tournée composée d'une séquence de sommets $(v_1, v_2, ..., v_n)$, la date minimale de début de service A_i^{min} et la date maximale de début de service à un sommet v_i sont données par les formules suivantes :

$$A_i^{min} = \begin{cases} \max\left(\sum_{j=1}^{i-1} p_j, e_i\right) & \text{si } v_i \text{ est un sommet de départ} \\ \sum_{j=1}^{i-1} p_j & \text{sinon} \end{cases}$$

$$A_i^{max} = \max_{j \in \{1,\ldots,n\}} (l_j) + \sum_{j=1}^{i-1} p_j.$$

Calcul des dates de début de service idéales aux sommets d'une tournée

Chaque sommet v_i à une date idéale de début de service A_i^T (target starting time), A_i^T est la date pour laquelle $f_i^{cvx}(A_i^T)$ est minimale. Cela correspond à la date optimale de début de service au sommet v_i quand il est servi individuellement et indépendamment des autres sommets de la tournée (sans contrainte de précédences), la valeur minimale de f_i^{cvx} est atteinte à l'un des arguments suivants :

$$A_i^T = argmin(f_i(A_i^{min}), f_i(e_i), f_i(l_i), f_i(A_i^{max})).$$

L'algorithme principal d'Optimal Timing

La Figure 3.9 présente l'algorithme développé. Il reprend la structure et les idées de celui de Chrétienne et Sourd, si ce n'est qu'il gère deux points de rupture par activité (sommet) et des dates de début minimales.

3.4 Conclusion

Le problème d'Optimal timing que nous avons présenté dans ce chapitre représente un intérêt majeur dans la résolution de notre problème de DARP. Notamment que le critère de la qualité de service rendue aux passagers est essentiel dans un service de Transport à la Demande. Dans ce service la dimension temporelle est très importante pour répondre aux exigence des passagers et par conséquent améliorer l'image du service vis-à-vis des usagers. Ce problème présente des similarités avec plusieurs problèmes qui sont largement étudiés dans la littérature notamment dans le domaine de l'ordonnancement. On s'est inspiré de ces problèmes pour résoudre notre problème après l'application de quelques modifications. Ce travail a été présenté dans [Atahran *et al.*, 2011b] et [Atahran *et al.*, 2012].

Algorithme principal de résolution de notre problème d'Optimal Timing

Entrée :
- $(v_1, v_2, ..., v_n)$: une séquence fixée de sommets ;
- $(p_1, p_2, ..., p_n)$: les durées opératoires des sommets ;
- $(A_1^T, A_2^T, ..., A_n^T)$: les dates de débuts de service idéales des sommets ;
- $(A_1^{min}, A_2^{min}, ..., A_n^{min})$: les dates minimales de début de service des sommets ;
- $(A_1^{max}, A_2^{max}, ..., A_n^{max})$: les dates maximales de début de service des sommets ;
- P : La somme des durées des sommets placés dans les blocs qui précèdent le bloc courant ;

- $b = 1$;
- $P = 0$;
- $B_1 = \{v_1\}$ // on construit le premier bloc avec le sommet v_1 ;
- $A_{B_1} = A_1^T$ // la date de début du bloc B_1 ;
- $p_{B_1} = p_1$ // la durée opératoire du bloc B_1 ;
- $R_1 = \left\{t, t \in \left\{A_1^{min}, e_1, l_1\right\}, \ \ avec \ \ t < A_1^T\right\}$;

Pour $i = 2$ à n **Faire** // on ajoute les sommets les uns après les autres ;

- **Si** $(A_i^T > A_{B_b} + p_{B_b})$ // le nouveau sommet va débuter un nouveau bloc ;
  - $BL = BL \cup B_b$ // on ajoute le bloc courant B_b dans l'archive des blocs BL ;
  - $P = P + P_{B_b}$
  - $b = b + 1$ // créer un nouveau bloc ;
  - $B_b = \{v_i\}$ // on construit un nouveau bloc ;
  - $A_{B_b} = A_i^T$ // le sommet v_i est placé à sa date de début de service idéale ;
  - $A_{B_b}^{min} = A_i^{min}$ // date de début minimale du bloc B_b ;
  - $p_{B_b} = p_i$ // la durée opératoire du bloc B_b ;
  - $R_b = \left\{t - \sum_{j<i} p_j, t \in \left\{A_i^{min}, e_i, l_i\right\}, \ \ avec \ \ t < A_i^T\right\}$ // On initialise le tas en y ajoutant également // les informations de pentes à gauche et à droite des points de rupture ;
  - $l_{B_b} = l^g(A_i^T)$ // la pente à gauche, de la fonction f_i^{cvx}, au point A_i^T,

- **Sinon** // on ajoute le sommet v_i au bloc courant
  - $B_b = B_b \cup \{v_i\}$;
  - $A_{B_b}^{min} = \max(A_{B_b}^{min}, A_i^{min} - p_{B_b})$;
  - $p_{B_b} = p_{B_b} + p_i$,
  - $\tilde{A} = \min\{t \in \{A_i^{min}, e_i, l_i, A_i^{max}\} \ tq \ t \geq A_B^{end}\}$
  - $l_{B_b} = l_{B_b} + l^g(\tilde{A})$ // mise à jour de la pente associée au bloc B_b
  - $A_B^{end} = A_B^{end} + p_i$;

  - **Tant que** $l_{B_b} \geq 0$ et $|R_b| > 0$ et $R_b[1] > A_{B_b}^{min}$ **Faire** // tant qu'on a intérêt à décaler B_b vers la gauche, //qu'il reste des points de rupture vers lesquels se déplacer //et qu'on n'a pas atteint la date de début minimale du bloc ;

    - Décaler le bloc R_b vers le point de rupture $R_b[1]$;
    - $l_{B_b} = l_{B_b} - (l^d(R_b[1]) - l^g(R_b[1]))$;
    - $A_{B_b} = R_b[1] + P$;
    - Supprimer $R_b[1]$ du tas R_b ;
    - **Si** $(A_{B_b} \leq A_{BL[|BL|]}^{end})$ **Alors**
      - $P = P - P_{BL[|BL|]}$
      - Fusionner le bloc courant B_b et le dernier bloc de la liste BL ;
      - Appeler la fonction *Merge*(B_b, $BL\,[|BL|]$), (voir 3.10) ;
      - Supprimer $BL\,[|BL|]$ de la liste des blocs BL ;
      - $b = b - 1$;
    - **Fin Si**

  - **Fin Tant que**

- **Fin Si**

Fin Pour

FIGURE 3.9 – Algorithme d'Optimal Timing

fusionnement de deux blocs B_1 et B_2 : ajouter les sommets de B_2 à la fin du bloc B_1

Procedure Merge (B_1, B_2) :

$B_1 = B_1 \cup B_2$ // ajouter les sommets du bloc B_2 à la fin du bloc B_1
$l_{B_1} = l_{B_1} + l_{B_2}$;
$A^{min}_{B_1} = \max\left(A^{min}_{B_1}, A^{min}_{B_2} - p_{B_1}\right)$;
$p_{B_1} = p_{B_1} + p_{B_2}$;
$\forall r \in R_2$, Si $r + P < A_{B_1}$ alors $R_1 = R_1 \cup \{r\}$;

Sortie : le bloc B_1

FIGURE 3.10 – la mise à jour des informations liées au fusionnement de deux blocs

Chapitre 4

Méthode de résolution du problème de DARP statique tricritère

4.1 Introduction

Dans ce chapitre nous nous intéressons à la résolution du problème de DARP statique multicritère. Nous commençons par présenter quelques résultats obtenus par une énumération des optima de Pareto sur quelques instances de petite taille. Ensuite, nous présentons l'algorithme NSGA-II qui a été utilisé pour résoudre ce problème. Nous présentons aussi trois heuristiques qui sont utilisées pour la création de la population initiale dans NSGA-II. A la fin de ce chapitre nous présentons les résultats obtenus sur des instances aléatoires ainsi que la comparaison avec des résulats issus de la littérature.

Avant de procéder à la résolution de notre problème de DARP statique, nous avons effectué quelques simulations afin d'observer l'évolution de la taille du front de Pareto quand on passe de deux à trois critères. Nous nous concentrons sur le problème de l'énumération de l'ensemble des optima de Pareto stricts sur des instances de petite taille. Cette énumération a été réalisé par algorithme simple de type *force brute*.

Nous avons considéré quatre petites instances constituées de n requêtes avec $n \in \{3, 4, 5, 6\}$ et de 2 véhicules hétérogènes. Chaque requête consiste à servir une seule personne. Les informations liées aux véhicules sont données dans la Table 4.1. La Table 4.2 présente les résultats obtenus.

Véhicule	Capacité	Coût kilométrique	Emission de CO_2 (g/km)
1	6	200	400
2	9	300	300

TABLE 4.1 – Les données liées aux véhicules

La Table 4.2 montre qu'avec trois critères le front de Pareto peut contenir un grand nombre de points et qu'il est significativement plus grand que la taille du front de Pareto généré avec deux critères. La première conclusion est que le critère CO_2 peut être conflictuel avec les deux autres critères. La deuxième conclusion est liée à l'évolution de la taille du front de Pareto pour le problème tricritère qui augmente avec la taille du problème (le

Taille du problème	Taille du front de Pareto	
n	2 critères (TC et QoS)	3 critères (TC, QoS et CO_2)
3	4	20
4	10	58
5	11	87
6	14	119

TABLE 4.2 – La taille de l'ensemble des optima de Pareto obtenus

nombre de requêtes) : pour les instances à 5 requêtes, le nombre d'optima de Pareto est égale à 87 et il vaut 119 pour les instances à 6 requêtes. Par conséquent, on peut déduire que le nombre de ces solutions risque d'être gigantesque pour des dizaines de requêtes et leur énumération complète s'avère difficile à réaliser. La Figure 4.1 montre l'évolution du nombre d'optima de Pareto stricts avec deux et trois critères.

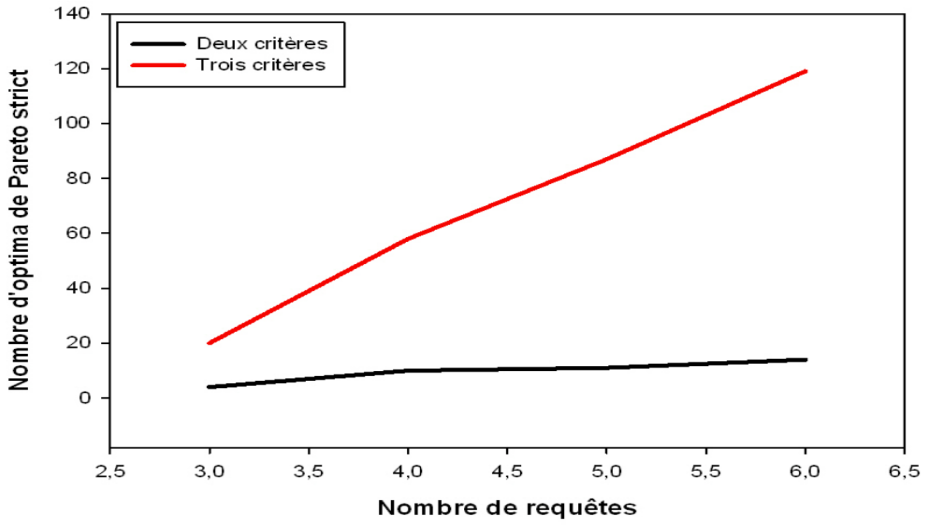


FIGURE 4.1 – Evolution du nombre d'optima de Pareto stricts

Par conséquent, il est évident qu'une méthode d'échantillonnage est nécessaire pour choisir un sous-ensemble représentatif, et de bonne diversité, de solutions non-dominées. Parmi les méthodes dédiées à l'énumération des optima de Pareto, la méthode NSGA-II semble être un bon choix ([Deb *et al.*, 2002]). Elle est connue pour son efficacité à résoudre des problèmes complexes d'optimisation multicritère et elle intègre naturellement une procédure d'échantillonnage basée sur la distance de *Crowding*. En effet, si la taille du front de Pareto s'avère bien supérieure à la taille de la population, cette méthode, en convergeant, va naturellement fournir un échantillonnage bien dispersé. Nous avons donc fait le choix d'implémenter cette méthode pour résoudre notre problème de DARP statique tricritère.

4.2 Echantillonnage par l'algorithme NSGA-II

Pour résoudre les problèmes d'optimisation multiobjectifs, de nombreux algorithmes génétiques ont été développés. Parmi les plus significatifs d'entre eux figurent : l'algorithme génétique à évaluation vectorielle (VEGA) présenté dans ([Schaffer, 1985]), l'algorithme SPEA-II proposé dans ([Zitzler *et al.*, 2001]) et l'algorithme NSGA-II (Non dominated Sorting Algorithm with elitism) développé dans ([Deb *et al.*, 2002]). Ces deux derniers algorithmes évolutionnaires élitistes ont montré leur efficacité pour résoudre plusieurs problèmes de transport multiobjectifs dans la littérature ([Chevrier *et al.*, 2010], [Sharma *et al.*, 2009] et [Lacomme *et al.*, 2006] parmi d'autres). Nous présentons dans la suite une description détaillée de l'algorithme NSGA-II ainsi que sa mise en oeuvre.

4.2.1 L'algorithme NSGA-II : principes généraux

Le principe de cet algorithme est illustré dans la Figure 4.2, on commence par créer une population initiale P_t de taille N, ensuite, une population d'enfants Q_t de même taille est générée à partir de la population P_t en utilisant les opérateurs de croisement et de mutation. Une population $R_t = P_t \cup Q_t$ est formée par l'union des deux population P_t et Q_t. Cette population composée de $2N$ individus et est triée selon la dominance de Pareto en différents fronts $(F_1, F_2, ..., F_g)$. Les solutions du front F_1 dominent celles du front F_2 qui dominent celles du front F_3 et ainsi de suite. L'algorithme NSGA-II est élitiste et consiste à préserver les meilleures solutions d'une itération à l'autre. C'est-à-dire que les individus du premier front sont tous copiés dans la prochaine population P_{t+1} si leur nombre ne dépasse pas N. Tant que la taille de la population P_{t+1} est inférieure à N, des solutions sont ajoutées pour compléter la population. Ainsi les solutions du front F_2 sont ajoutées, celles de F_3 et ainsi de suite. Si les solutions d'un front ne peuvent pas rentrer toutes dans la population P_{t+1}, alors les solutions de ce front sont classées suivant une mesure de diversité (par exemple : la distance de *Crowding*). Les solutions les plus dispersées (de plus grande distance de Crowding) sont ajoutées jusqu'à compléter la population P_{t+1}. La Figure 4.3 illustre les grandes lignes de cet algorithme.

Dans la suite nous présentons en détail les principaux composants de l'algorithme NSGA-II, ainsi que leur mise en oeuvre dans le cadre de cette thèse pour résoudre le problème de DARP tricritère. Les principaux composants de cet algorithme sont :

- La représentation ou le codage d'une solution,
- L'initialisation de la population,
- L'évaluation d'une solution,
- Les opérateurs de reproduction : croisement et mutation,
- La stratégie de préservation de la diversité,
- La stratégie de sélection,
- La condition d'arrêt.

4.2.2 Encodage d'une solution

Le codage d'une solution est le point de départ de tout type d'algorithme évolutionnaire : une solution doit être représentée dans l'espace décisionnel avec une représentation

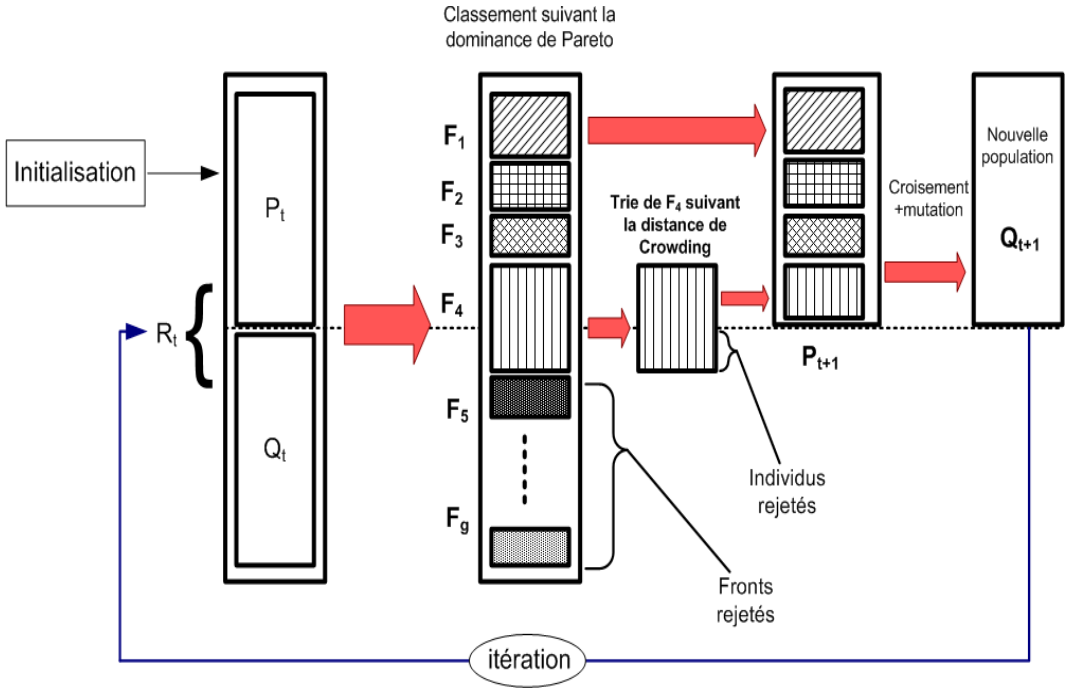


FIGURE 4.2 – Itération de l'algorithme NSGA-II

Algorithme NSGA-II

Entrée : P_0, Q_0 (populations initiales), N (taille de la population) ;
tmax (nombre maximal de générations) ;
t=0 ;
Tant que $(t < tmax)$
  $R_t = P_t \cup Q_t$;
  effectuer un tri selon la dominance de Pareto sur la population R_t pour identifier les fronts F_j ;
  $P_{t+1} = \emptyset$; $j = 1$;
  Tantque $(|P_{t+1}| + |F_j| < N)$
    $P_{t+1} = P_{t+1} \cup F_j$;
    $j = j + 1$
  Fin Tant que
  Remplir la population P_{t+1} à partir de F_j en utilisant la distance de *crowding* si nécessaire ;
  $i = 1$; $Q_{t+1} = \emptyset$;
  Tant que $(i \leq N/2)$
    Sélectionner deux chromosomes C et C' de la population P_{t+1} ;
    Appliquer l'opérateur de croisement sous la probabilité p_N^C sur C et C' pour créer C_f et C'_f ;
    Appliquer l'opérateur de mutation sous le probabilité p_N^M sur C_f et C'_f ;
    $Q_{t+1} = Q_{t+1} \cup \{C_f\} \cup \{C'_f\}$;
    $i = i + 1$;
  Fin Tant que
  $t = t + 1$;
Fin Tant que
Sortie : le premier front F_1 de l'ensemble $P_{tmax} \cup Q_{tmax}$

FIGURE 4.3 – Pseudo-code de l'algorithme NSGA-II

pertinente et adaptée au problème. Le choix de cette représentation aura une influence considérable sur la façon dont les solutions seront manipulées par les opérateurs de croise-

ment et de mutation et lors de l'étape de l'évaluation. Différentes représentations peuvent être utilisées pour un même problème.

Il existe plusieurs manières de coder une solution au problème de DARP. Particulièrement, on distingue deux types de représentations : une représentation directe et une représentation indirecte. Avec cette première toute l'information de la solution est présente dans le chromosome. Avec ce type de codage l'évaluation des solutions est calculée directement sur les solutions et ne nécessite aucune procédure supplémentaire pour construire une solution correcte et complète. Tandis que dans le cas d'une représentation indirecte, le chromosome ne représente pas directement une solution et il est nécessaire de passer par une étape complémentaire pour obtenir une solution correcte et complète. En utilisant ce codage, le problème de DARP peut être représenté par un vecteur qui contient une suite de sommets à visiter et les tournées ne sont pas distinguées directement sur le chromosome. L'avantage de ce type de codage est qu'il présente une facilité au niveau de l'application des opérateurs de croisement et de mutation et que quelques contraintes peuvent être reportées dans la phase de décodage. L'inconvénient est que ce codage nécessite la définition d'une procédure efficace pour construire des solutions complètes.

Pour le codage du chromosome dans le cas de notre problème de DARP tricritère, nous avons utilisé un codage direct pour éviter de définir des procédures complémentaires pour le décodage. Le codage choisi consiste à considérer un ensemble de vecteurs de tailles différentes, chaque vecteur représentant une tournée affectée à un véhicule et contenant une suite de sommets (points de départ et d'arrivée des requêtes) à servir dans l'ordre (de gauche vers la droite). Le dépôt ne figure pas dans la représentation des tournées, mais il est virtuellement placé au début et à la fin de chaque tournée. La Figure 4.4 montre un exemple de codage d'une solution composée de 5 tournées.

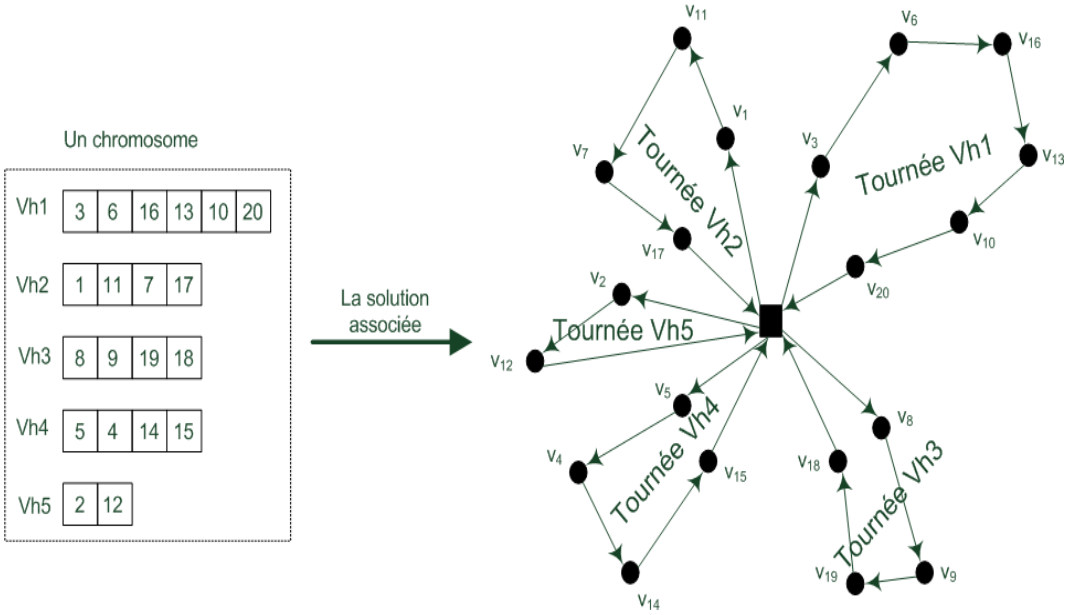


FIGURE 4.4 – Un exemple de codage d'une solution de 5 tournées et 10 requêtes

Pour connaitre les heures de passages pour une tournée donnée, aux sommet il suffit

alors de résoudre le problème d'optimal timing tel qu'introduit dans le chapitre 3. Cela se fait en temps polynomial.

4.2.3 Initialisation de la population

La phase d'initialisation de la population dans NSGA-II peut influencer la qualité des résultats obtenus. Une population initiale constituée par des solutions de bonne qualité et diversifiées peut augmenter la chance de trouver une approximation du front de Pareto de bonne qualité, aussi bien en termes de convergence que de diversité. Pour cela, il faut garder à l'esprit que la population initiale doit être bien construite et bien diversifiée dans l'espace de recherche afin d'éviter une convergence prématurée.

Nous avons proposé trois heuristiques différentes pour initialiser la population dans l'algorithme NSGA-II. Chaque heuristique est guidée dans un sens différent de l'autre, c'est-à-dire que pour chaque heuristique l'objectif à optimiser est différent d'une autre heuristique. Le but est d'assurer un certain niveau de diversité dans les solutions de la population initiale qui seront générées par ces heuristiques.

4.2.3.1 Heuristique H1

La première méthode que nous avons proposé consiste à générer N solutions en se basant sur les caractéristiques temporelles du problème et en cherchant à minimiser l'insatisfaction des passagers. Le principe de cette heuristique consiste à trier les sommets selon les bornes supérieures des fenêtres de temps liées à l'ensemble des sommets à servir. Il faut noter que pour chaque requête r_i la borne supérieure de la fenêtre du temps du sommet de départ v_i est strictement inférieure à la borne supérieure du sommet d'arrivée v_{i+n}.

Une fois les sommets triés, on parcourt la liste des sommets (du sommet qu'a la plus petite valeur l_i vers le sommet de plus grande valeur) et pour chaque sommet, un véhicule est choisi aléatoirement pour servir ce sommet. S'il s'agit d'un sommet d'arrivée d'une requête alors il sera obligatoirement affecté au même véhicule qui a servi le sommet de départ de la même requête. Notons que pour générer N solutions on répète la phase d'affectation des sommets aux véhicules N fois.

Le tri des sommets selon l_i garantit le respect des contraintes de précédence dans chaque tournée entre le sommet de départ et d'arrivée de chaque requête. Pour satisfaire la contrainte de capacité du véhicule, nous avons élaboré une procédure de correction des solutions qui sont non réalisables vis-à-vis de la capacité maximale du véhicule. Cette procédure consiste à repérer les sommets mal placés qui conduisent à une violation de cette contrainte et à rectifier leur position. L'algorithme de la procédure de correction de chaque tournée vis-à-vis de la contrainte de la capacité du véhicule est donné dans la Figure 4.5. Le principe de cette procédure est de parcourir la tournée sommet par sommet et calculer le nombre de personnes à bord du véhicule à chaque sommet servi. Si le nombre de personnes excède la capacité maximale du véhicule alors la requête (sommet de départ et d'arrivée) est supprimée de la tournée pour être réinsérée plus tard dans la même tournée.

Il faut noter que la capacité d'un véhicule ne peut être violée qu'au niveau d'un sommet d'embarquement. La procédure de suppression des requêtes qui sont mal placées se poursuit

jusqu'à la fin de la tournée. Ensuite une autre procédure de réinsertion est définie pour replacer les requêtes qui ont été supprimées dans la même tournée tout en respectant la capacité du véhicule. La Figure 4.6 illustre les lignes principales de l'algorithme utilisé pour créer une solution réalisable.

Dans la suite $q_{[i]}$ symbolise le nombre de personnes servies après avoir visité le sommet situé à la position i de la tournée Vh_k : la valeur de $q_{[i]}$ peut être négative ou positive selon que le sommet servi à la position i est un sommet d'embarquement ou de débarquement.

Correction d'une tournée

Entrée :
- Vh_k : une tournée de p sommets, associée à un véhicule k de capacité Q_k ;
- $i = 1$;
- $cpt = 0$: un compteur ;

Tant que $(i < p)$

- $E_i^k = E_{i-1}^k + q_{[i]}$ // $E_i^k = 0$;
- Si $(E_i^k > Q_k)$
  - $cpt = cpt + 1$;
  - Si $(cpt == 1)$ Alors $J_{ins} = i$;
  - Ajouter le sommet de départ $Vh_k[i]$ à la liste R ;
  - Supprimer les sommets $Vh_k[i]$ et $(Vh_k[i] + n)$ de la tournée Vh_k ;
  - $p = p - 2$;
- Else
  - $i = i + 1$;
- Fin Si

Fin Tant que

Si $(R \neq \emptyset)$
- Appeler la procédure `Insert_Procedure` $(Vh_k, R, J_{ins} + 1)$;

Fin Si

Sortie : Tournée Vh_k corrigée

FIGURE 4.5 – Procédure de correction d'une tournée vis-à-vis de la contrainte de capacité du véhicule

Insertion d'une liste de requêtes R dans la tournée Vh_k

`Insert_Procedure` (Vh_k, R, f) ;

Pour $i = 1$ à $|R|$ Faire
- Enumérer toutes les insertions possibles de $R[i]$ entre la position f et $|Vh_k|$;
- Choisir la meilleure position qui minimise la distance parcourue ;
- Insérer le sommet de destination $R[i] + n$ (relatif à $R[i]$) de la même façon que $R[i]$ et
- Choisir la position qui minimise la distance parcourue ;

Fin Pour ;

Sortie : Tournée Vh_k

FIGURE 4.6 – Procédure d'insertion d'une liste de requêtes dans une tournée

4.2.3.2 Heuristique H2

L'objectif de cette heuristique est de générer des solutions de bonne qualité en termes de distance totale parcourue. Cette heuristique est basée sur la méthode de balayage qui est largement utilisée dans la littérature pour résoudre les problèmes de tournées de véhicules ([Renaud et Boctor, 2002]). Cette méthode proposée initialement par [Gillett et Miller, 1974] consiste à regrouper les sommets en tournées selon les angles polaires par rapport à un axe de référence donné. Plusieurs auteurs ont utilisé cette méthode comme heuristique de construction pour résoudre des problèmes de tournées de véhicules sans fenêtres de temps (voir [Cordeau *et al.*, 1997], [Renaud et Boctor, 2002] et [Solomon, 1987] parmi d'autres).

Nous nous sommes inspirés de cette méthode pour résoudre notre problème de DARP. L'objectif est de produire un ensemble de solutions réalisables qui seront utilisées pour NSGA-II. Seule la distance totale parcourue est minimisée : ce critère touche implicitement les critères de coût de transport et d'impact environnemental.

Supposons que nous disposons de n requêtes à servir avec M véhicules. La première étape de cet algorithme consiste à choisir un axe et une origine de référence. Ensuite nous calculons les angles polaires pour les $2n$ sommets par rapport à l'origine et l'axe de référence et dans le sens anti-horaire. Les sommets sont triés dans l'ordre croissant de ces angles polaires.

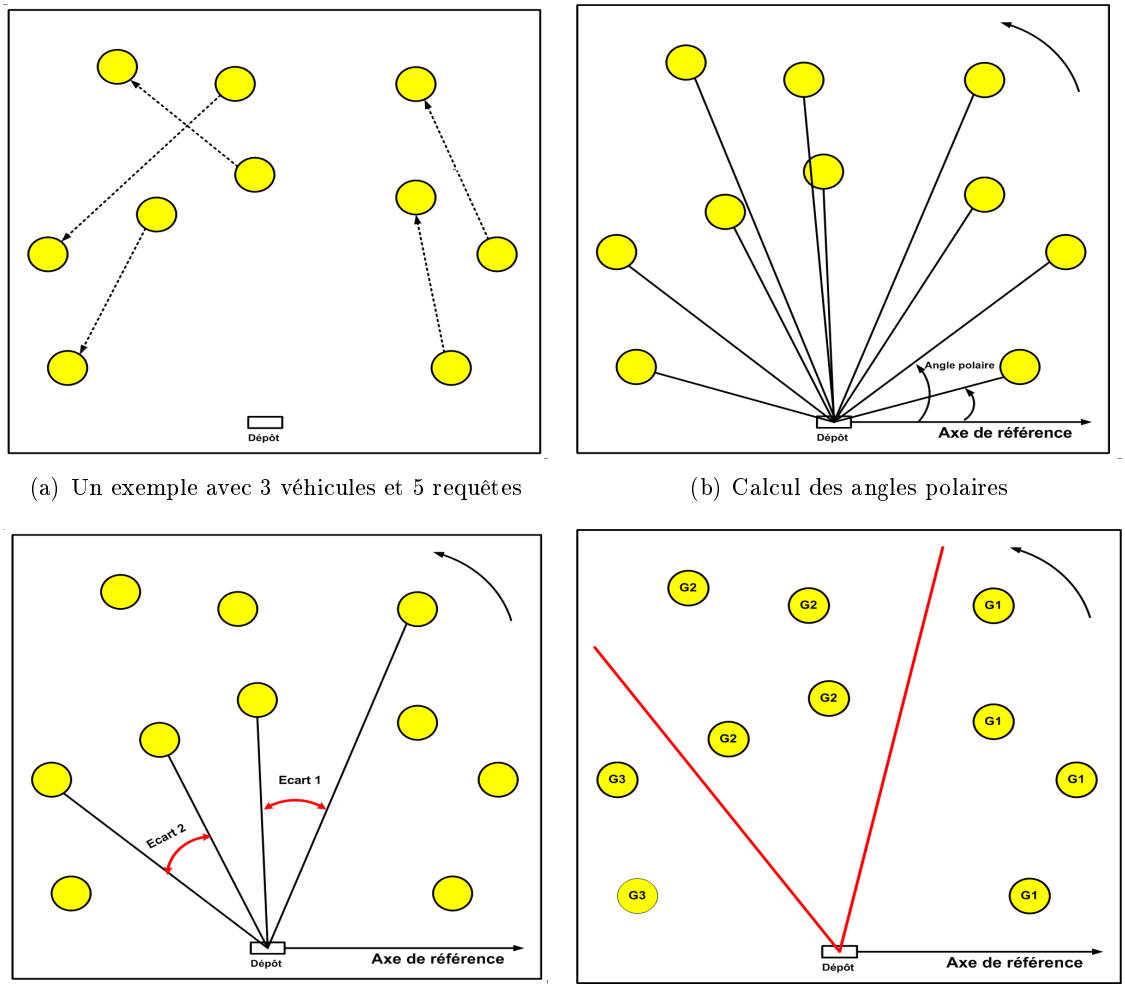
La deuxième étape consiste à calculer l'écart entre deux angles successifs. Ensuite, $M-1$ sommets sont sélectionnés selon les $M-1$ plus grand écart, les sommets sélectionnés sont utilisés comme étant des délimiteurs lors de la construction des M groupes. Dans la troisième étape, les groupes de sommets doivent être vérifiés vis-à-vis de la contrainte de couplage (le sommet de départ et d'arrivée d'une même requête doivent appartenir au même groupe). Si les deux sommets de la même requête n'appartiennent pas au même groupe alors une affectation aléatoire est effectuée en déplaçant le sommet de départ (ou sommet d'arrivée) vers le groupe qui contient son sommet d'arrivée (ou sommet de départ) correspondant. La Figure 4.9 présente le pseudo de code de l'heuristique H1 pour générer une solution.

Lors de la construction des groupes de sommets, seule la proximité géographique a été prise en considération (en utilisant les angles polaires). La quatrième étape de cet algorithme consiste à trier les sommets de chaque groupe dans l'ordre croissant de la borne supérieure l_i des fenêtres de temps, ce qui permet d'avoir un séquencement cohérent vis-à-vis des contraintes temporelles.

L'objectif est de prendre en considération, même légèrement, les fenêtres de temps afin de ne pas trop détériorer la qualité de service rendue aux passagers. En se basant sur le séquencement de chaque tournée (dans la suite les groupes de sommets sont appelés des tournées), les véhicules sont affectés aux tournées selon le nombre de personnes affectées à chaque tournée. Un véhicule plus petit est affecté à une tournée avec moins de demandes. La dernière étape de cet algorithme consiste à vérifier la contrainte de capacité du véhicule pour chaque tournée. La même procédure que dans l'heuristique H1 (voir les Figures 4.5 et 4.6) est utilisée pour corriger une tournée dans le cas de la contrainte de capacité du véhicule est violée.

Les Figures 4.7 et 4.8 montrent les étapes principales de la construction des tournées

par l'heuristique H2.



(a) Un exemple avec 3 véhicules et 5 requêtes

(b) Calcul des angles polaires

(c) Calcul des écarts maximaux

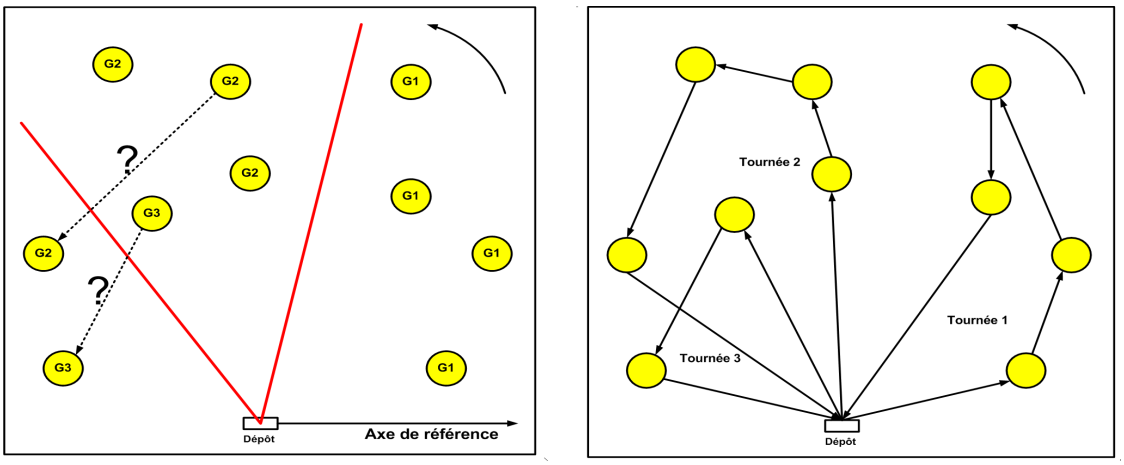
(d) Construction des groupes de sommets

FIGURE 4.7 – Exemple de calcul des angles polaires

4.2.3.3 Heuristique H3

Cette heuristique fait partie des méthodes en deux phases qu'ont été proposées initialement pour résoudre les problèmes de tournées de véhicules (VRP) ([Bräysy et Gendreau, 2005]) avant d'être adaptées pour résoudre les VRPs avec les contraintes de fenêtres de temps ([Dondo et Cerda, 2007]). Dans la première phase, cette méthode consiste à décomposer les sommets sous forme de *Clusters*. Ensuite, dans une seconde phase, ces sommets sont mis dans l'ordre de visite afin de construire des tournées réalisables.

Pour créer des *Clusters* nous utilisons une mesure de distance entre les sommets. Dans le problème de DARP que nous avons considéré dans le cadre de cette thèse chaque sommet est caractérisé par ses coordonnées géographiques et une fenêtre de temps. Il est donc



(a) Correction des groupes (b) Construction des tournées

FIGURE 4.8 – Construction des tournées de véhicules

Heuristique H2
Entrée : $v_1, v_2, ..., v_{2n}$: l'ensemble de sommets (points de départ et d'arrivée de requêtes) ; M : le nombre de véhicules ; *Etape* 1 : Calculer les angles polaires entre chaque sommet et l'axe de référence ; Trier par ordre croissant les angles polaires ; Calculer l'écart entre chaque deux angles successifs ; *Etape* 2 : Sélectionner les $M-1$ sommets correspondant aux $M-1$ plus grand écarts ; ces écarts vont servir de délimiteurs lors de la construction de M groupes, notés G_i , $\forall i \in 1, ..., M$; *Etape* 3 : Pour chaque groupe G_i les procédures de vérification et de correction suivantes sont appliquées : • Vérification de la contrainte de couplage : si deux sommets d'une même requête n'appartiennent pas au même groupe alors une affectation aléatoire est effectuée en déplaçant le sommet de départ (ou sommet d'arrivée) vers le groupe qui contient son sommet d'arrivée (ou sommet de départ) correspondant ; • Affectation des véhicules aux groupes de sommets selon le nombre de personnes à servir dans chaque groupe ; *Etape* 4 : Les sommets de chaque groupe G_i sont triés selon la borne supérieure de la fenêtre de temps l_i *Etape* 5 : Vérification de la contrainte de capacité du véhicule pour chaque groupe G_i ; la correction est effectuée en utilisant le même algorithme de l'heuristique H1 (voir la Figure 4.5) Pour générer P solutions, on répète P fois les étapes *Etape* 3, *Etape* 4 et *Etape* 5 **Sortie** : P solutions

FIGURE 4.9 – Les étapes principales de l'heuristique H2

raisonnable d'intégrer simultanément les caractéristiques spatiales et temporelles lors de la mesure de la proximité entre les sommets. C'est-à-dire un ensemble de sommets sont regroupés dans un même *Cluster* s'ils sont proches géographiquement mais aussi s'ils tournent sur les mêmes plages horaires. Pour cela nous avons utilisé une mesure de distance qui intègre les caractéristiques spatiales et temporelles des sommets.

Soit un sommet v_i défini par ses coordonnées géographiques planaires (X_i^1, X_i^2) et sa fenêtre de temps $[e_i, l_i]$. Nous définissons une troisième dimension temporelle par :

$$X_i^3 = \frac{e_i + l_i}{2}.$$

Par conséquent, chaque sommet v_i est représenté par un vecteur à trois dimensions

$[X_i^1, X_i^2, X_i^3]$. Pour mesurer la distance entre deux sommets, nous utilisons la métrique euclidienne définie par la formule suivante :

$$d(v_i, v_j) = \sqrt{\sum_{f=1}^{3} \left(X_i^f - X_j^f\right)^2}.$$

où $d(v_i, v_j)$ est la distance entre les sommets v_i et v_j. Cette distance est utilisée dans la première phase de l'heuristique H3 pour créer des *Clusters*. Chaque *Cluster* est affecté à un seul véhicule pour lequel un séquencement des sommets est alors défini. La Figure 4.10 illustre deux tournées représentées avec une dimension temporelle.

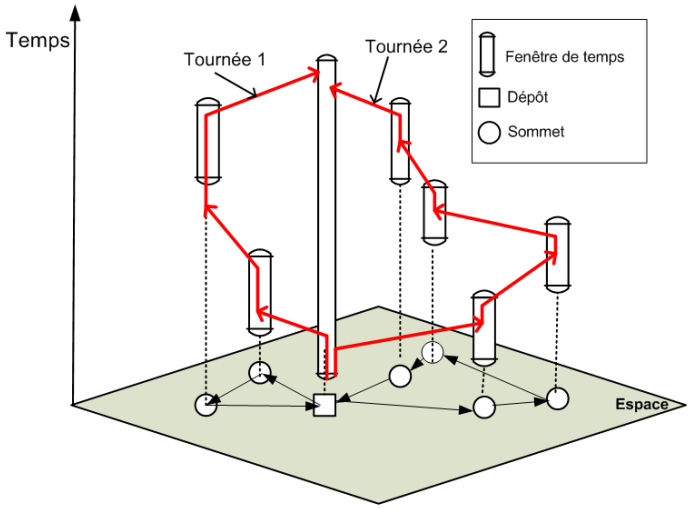


FIGURE 4.10 – Deux tournées représentées dans un espace avec une dimension temporelle

D'un point de vue technique, les coordonnées X_i^1, X_i^2 et X_i^3 n'ont pas les mêmes ordres de grandeurs : il est donc nécessaire de les transformer pour les mettre sur une échelle comparable. Nous avons donc considéré des valeurs centrées et réduites des X_i^f. Nous posons :

$$\overline{X^f} = \frac{1}{2n}\sum_{j=1}^{2n} X_j^f \qquad et \qquad \sigma^f = \sqrt{\frac{1}{2n}\sum_{j=1}^{2n}\left(X_j^f - \overline{X^f}\right)^2} \quad ,\forall f = 1, 2, 3.$$

Ainsi, chaque X_j^f est remplacé par sa valeur centrée et réduite $X_j'^f$ définie par :

$$X_j'^f = \frac{X_j^f - \overline{X^f}}{\sigma^f}.$$

Revenons maintenant sur la méthode de création des *Clusters*. Dans le problème de DARP que nous abordons dans cette thèse, le nombre de véhicules à utiliser est aussi une variable de décision. Nous connaissons le nombre maximal de véhicules qui sont disponibles

au dépôt, mais nous ne sommes pas obligés de les utiliser tous pour servir l'ensemble des requêtes des clients. Ce qui signifie que le nombre de *Clusters* (tournées) à former n'est pas connu à l'avance et le choix du nombre de *Clusters* est déterminé a posteriori. Pour cela nous avons utilisé une méthode de Classification Ascendante Hiérarchique (CAH) pour tester plusieurs configurations au niveau du nombre de tournées à construire.

Cette méthode est très utilisée dans le domaine de l'analyse de données pour faire de la classification non-supervisée : elle permet de construire hiérarchiquement un ensemble de M classes à partir de n données en entrée. Chaque donnée est représentée par ses coordonnées qui peuvent être d'une dimension supérieure à deux, comme dans le cas dans notre problème.

Pour agréger deux *Clusters*, plusieurs critères (ou indices d'agrégation) ont été utilisés dans la littérature ([Jain *et al.*, 1999]). Ce critère consiste à définir une métrique qui mesure la proximité entre chaque paire de *Clusters*. Dans la littérature on distingue plusieurs types d'indices d'agrégation :

- Distance minimum (ou *single linkage*),
- Diamètre maximal (ou *complète linkage*),
- Distance moyenne,
- Plus proches voisins,
- Moment d'ordre deux (Distance de Ward),
- Centre de gravité.

Nous avons choisi d'utiliser l'indice *Diamètre maximal* comme critère d'agrégation des *Clusters*, car ce critère résiste à la présence de données bruitées et aberrantes. La distance entre deux *Clusters* C_g et C_l est définie par :

$$D(C_g, C_l) = \underset{v_i \in C_g, v_j \in C_l}{max}(d(v_i, v_j)) .$$

Initialement chaque sommet représente un *Cluster* à lui tout seul. Soit M le nombre de *Clusters* à construire, nous fusionnons itérativement deux *Clusters* C_l and C_g qui ont une distance $D(C_l, C_g)$ minimale jusqu'à ce que le nombre de *Clusters* souhaité soit atteint. La Figure 4.11 illustre les lignes principales de l'algorithme CAH.

Algorithme CAH
Initialisation : Chaque sommet est placé dans son propre *Cluster*, Calcul de la matrice de distances D entre chaque couple de *Clusters* **Répéter** Sélectionner dans D les deux *Clusters* les plus proches C_I et C_J ; Fusionner C_I et C_J dans un *Cluster* C_G plus général ; Mise à jour de D en calculant la proximité entre C_G et les *Clusters* existants **Jusqu'à** la fusion des 2 derniers *Clusters*

FIGURE 4.11 – Principales étapes de l'algorithme de classification CAH

Le principal inconvénient de la classification ascendante hiérarchique est sa complexité algorithmique qui n'est pas linéaire : pour passer de $k+1$ classes à k classes, il faut calculer $\frac{(k+1)k}{2}$ distances et réunir les deux classes les plus proches. Si n est le nombre d'individus

à classer, la complexité de l'algorithme de base est de l'ordre de $O(n^3)$. La situation est un peu améliorée par l'algorithme des voisins réciproques, qui permet de passer d'une complexité de $O(n^3)$ en $O(n^2)$, en regroupant judicieusement plus de deux observations (pas seulement les deux plus proche). On peut aussi utiliser des méthodes de classification mixte (on commence par *k-means* pour calculer par exemple 10 classes après on se base sur ces dix classes pour faire la CAH).

La Figure 4.16 présente le pseudo code de l'heuristique H3 pour construire P solutions en se basant sur l'algorithme de classification CAH de la Figure 4.11.

L'heuristique H3 exécute l'algorithme de CAH pour créer des groupes de sommets. Le nombre de *Clusters* N_c à construire n'est pas connu, mais on connait le nombre maximal M de véhicules disponibles au dépôt. Théoriquement, on peut tester toutes les possibilités N_c avec $N_c \in \{1, 2, ..., M\}$. Dans l'heuristique H3 nous avons choisi de faire varier le nombre de *Clusters* N_c dans $\left[N_C^{min}, M\right]$, avec $N_C^{min} <= M$ un paramètre à définir a priori.

Après l'exécution de l'algorithme CAH on récupère les solutions générées avec M, $M-1$,...,N_C^{min} *Clusters*. Notons que ces solutions peuvent être incomplètes (non réalisables) et il est nécessaire de prévoir des procédures pour la correction des solutions pour obtenir des solutions sous forme de tournées réalisables. Pour chaque solution et pour chaque *Cluster* on vérifie la contrainte de couplage, c'est-à-dire que l'on vérifie si les sommets d'arrivée et de départ d'une même requête appartiennent au même *Cluster*. Si ce n'est pas le cas, une affectation aléatoire est utilisée pour mettre les deux sommets dans un même cluster. Ensuite, les sommets de chaque cluster sont triés selon la borne supérieure des fenêtres de temps l_i. L'affectation des véhicules est effectuée selon le nombre maximal de personnes qui peuvent être à bord du véhicule le long de la tournée (cluster). Par la suite on corrige les tournées vis-à-vis de la contrainte de capacité du véhicule, si nécessaire, de la même manière que dans l'heuristique H1.

Maintenant que les tournées sont réalisables, nous appliquons un opérateur de voisinage 2-Opt sur chaque tournée pour créer au plus S_I tournées alternatives qui sont meilleures au niveau de la distance totale parcourue mais qui peuvent être mauvaises au niveau du critère de l'insatisfaction des passagers. A partir de ces tournées alternatives, nous construisons au plus S_p solutions non-dominées en combinant deux à deux les tournées de chaque alternatives. La méthode 2-Opt consiste à découper deux arcs dans une tournée ce qui donne deux sous-séquences disconnectées. Ensuite, on reconnecte les deux segments en inversant le sens de parcours de la séquence qui ne contient pas le dépôt. Les Figures 4.14 et 4.15 montrent un exemple de l'application de l'opérateur 2-Opt sur une tournée composée de 6 sommets ainsi que leur représentation dans l'espace des deux critère : distance parcourue et insatisfaction des passagers.

Les Figures 4.12 et 4.13 montrent un exemple de l'application du processus de génération de nouvelles solutions à partir d'une solutions rélisable en utilisant l'opérateur 2-Opt : A partir d'une solution initiale composée de trois tournées (*"Tournée R1"*, *"Tournée R2"*, *"Tournée R3"*), nous appliquons l'opérateur 2-Opt à chaque tournée pour générer au plus S_I tournées alternatives. Ensuite, les tournées alternatives de *"Tournée R1"* sont combinées avec celles de *"Tournée R2"*, nous obtenons des solutions partielles de types (*"Tournée R1_ V1"*, *"Tournée R2_ V1"*), (*"Tournée R1_ V1"*, *"Tournée R2_ V2"*),etc. Ensuite, ces solutions partielles sont combinées avec les tournées alternatives de *"Tournée R3"*

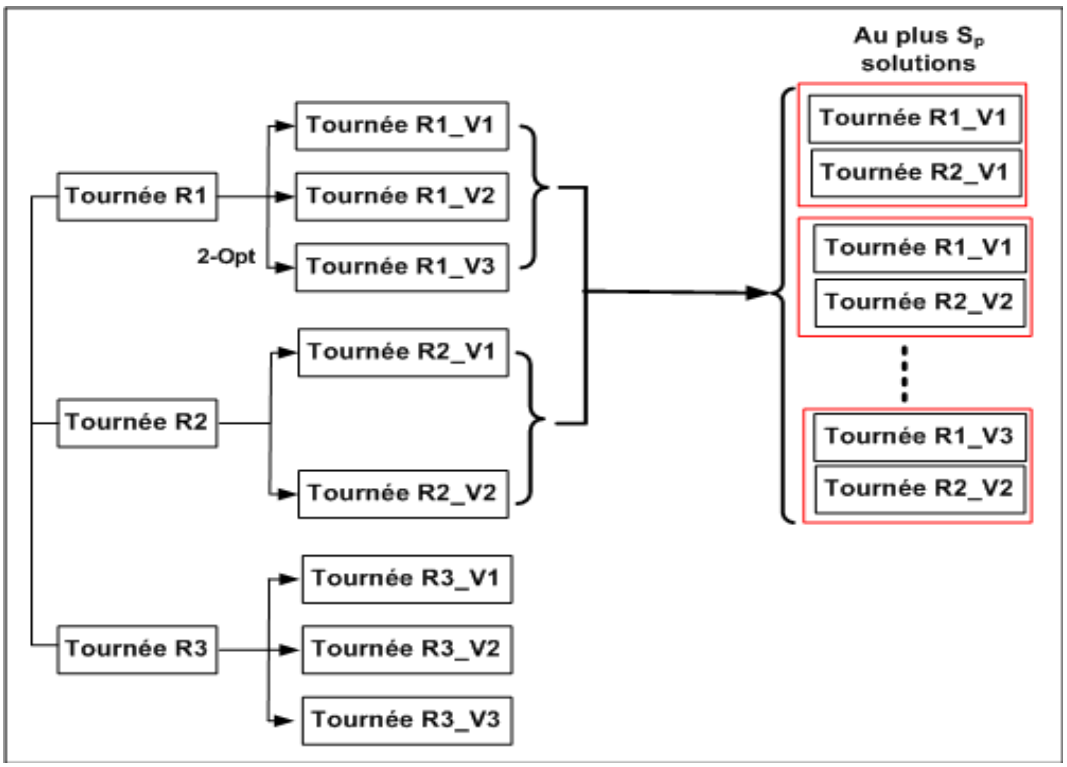


FIGURE 4.12 – Exemple de combinaison des tournées alternatives générées par la méthode 2-Opt pour générer de nouvelles solutions

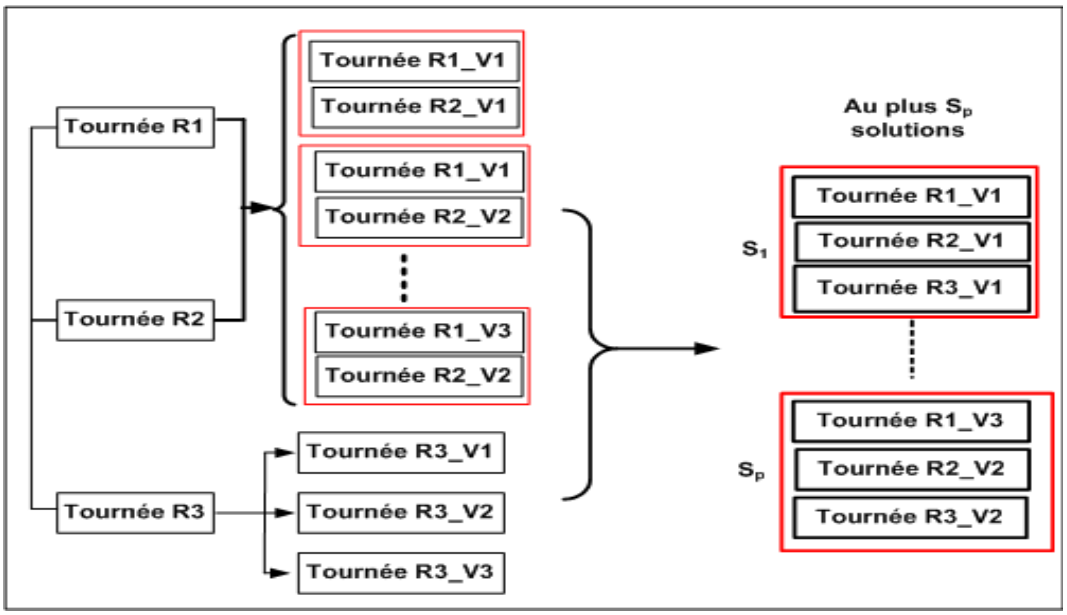


FIGURE 4.13 – Les solutions générées à la fin du processus de combinaison des tournées

pour obtenir à la fin de nouvelles solutions de la forme suivante : (*"Tournée R1_ V1"*, *"Tournée R2_ V1"*, *"Tournée R3_ V1"*), (*"Tournée R1_ V1"*, *"Tournée R2_ V1"*, *"Tournée R3_ V2"*),etc.

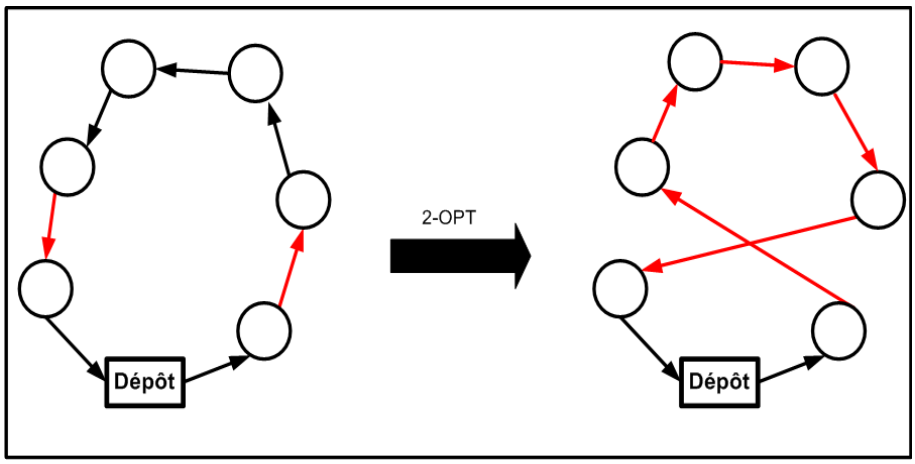


FIGURE 4.14 – Exemple de l'application de l'opérateur 2-Opt sur une tournée de 6 sommets

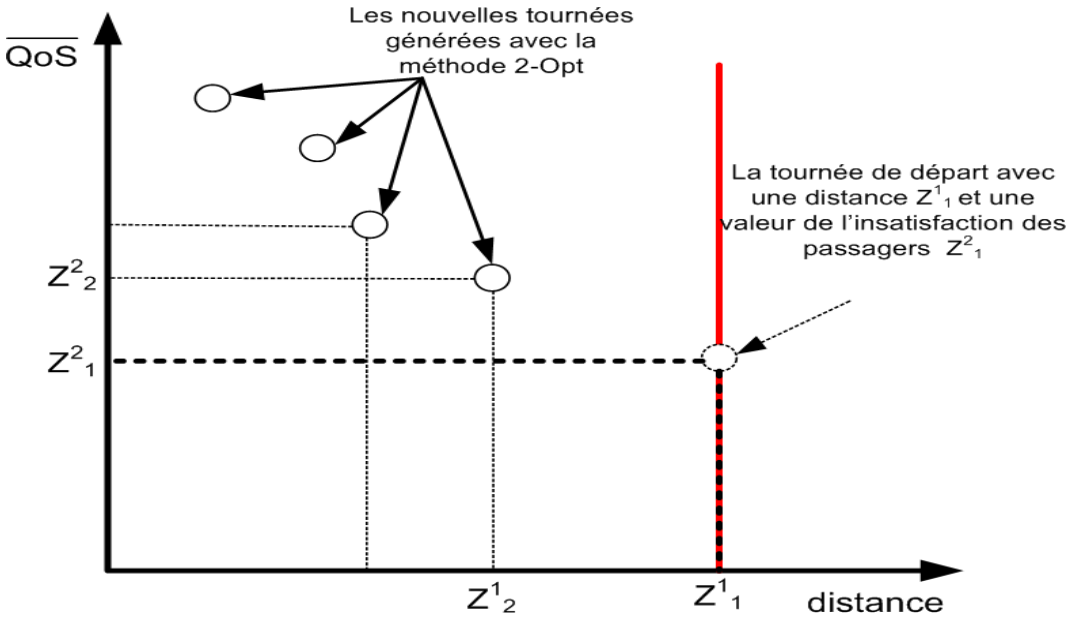


FIGURE 4.15 – Exemple de tournées générées avec la méthode 2-Opt représentées par leurs évaluations selon la distance parcourue et le critère de l'insatisfaction des passagers

4.2.4 L'évaluation

L'évaluation associée à une solution réalisable (un chromosome) est un vecteur de valeurs dont chaque élément correspond à la qualité de la solution selon la fonction objectif correspondante. Ainsi, l'évaluation d'une solution réalisable correspond à un vecteur

Heuristique H3

Entrée : une liste de sommets (points de départ et d'arrivée des requêtes) $v_1, v_2, ..., v_{2n}$ et M véhicules,
N_C^{min} : le nombre minimal de *clusters* souhaité, avec $N_C^{min} <= M$,
S_p : le nombre de solutions à générer
S_I : le nombre maximal de tournées améliorées à générer par la méthode 2-Opt pour chaque tournée,
L_S : une liste initialement vide $L_S = \emptyset$,
L_F : une liste initialement vide $L_F = \emptyset$,,

Etape 1 :
- Créer N_c solutions partiales à l'aide de la classification CAH, la $1^{\text{ère}}$ solution avec M Clusters, la $2^{\text{ème}}$ avec $M-1$ clusters,..., et la N_C^{min} ème avec N_C^{min} clusters. Les solutions sont ajoutées dans une liste S

Pour chaque solution dans S *Faire*

Etape 2 :
- Corriger les clusters vis-à-vis de la contrainte de couplage,

Etape 3 :
- Les sommets de chaque cluster sont triés selon la borne supérieure de la fenêtre de temps l_i,

Etape 4 :
- Affecter les véhicules aux clusters selon le nombre maximal de personnes à bord de chaque véhicule le long de chaque cluster (tournée),

Etape 5 :
- Vérifier la contrainte de la capacité du véhicule pour chaque tournée, les corrections sont effectuées de la même manière que l'heuristique H2,

Etape 6 :
- Générer au plus S_I tournées alternatives pour chaque tournée en utilisant une recherche locale 2-Opt (voir Figure 4.14),

Etape 7 :
- Nous construisons des solutions en combinant à chaque itération les alternatives de chaque tournée avec les alternatives d'une autre tournée de la même solution,
- D'une combinaison à l'autre nous gardons au plus S_p solutions non-dominées en terme de la distance totale parcourue et le critère de qualité de service (voir les Figure 4.12, **??**),
- Les solutions S_p sont ajoutées dans la liste L_S,
- Vider la liste S_p,

Fin Pour

Etape 8 :
- Trier la liste L_S sous forme de sous ensembles non-dominées selon le critère de distance totale parcourue et de qualité de service et remplir la liste L_F par les p meilleures solutions,

Sortie : L'ensemble des p solutions de L_F.

FIGURE 4.16 – Les étapes principales de l'heuristique H3

(Z_1, Z_2, Z_3) : la première valeur Z_1 correspond au coût total de transport, la deuxième valeur Z_2 correspond à l'insatisfaction des passagers et la troisième valeur Z_3 représente la quantité de CO_2 émise par l'ensemble des véhicules. Notons que pour calculer la valeur Z_2 nous appliquons l'algorithme d'optimal timing sur chaque tournée (Chapitre 3).

Une fois les trois valeurs des trois objectifs calculées pour une population, une procédure efficace de calcul du front de Pareto est nécessaire afin de le trier . Cette procédure est appelée d'une manière intensive dans l'algorithme NSGA-II. [Kung *et al.*, 1975] ont proposé une méthode efficace qui permet de calculer l'ensemble des solutions non-dominées F_1 dans une population de solutions P avec une complexité de $O\left(NlogN\right)^{M-2}$ si $M \geq 4$ et $O\left(NlogN\right)$ pour $M = 2, 3$, avec M le nombre d'objectifs et N la taille de la population P. C'est l'algorithme que nous avons utilisé. Son pseudo-code est donné dans la Figure 4.17.

Pour classer une population P sous forme d'un ensemble de fronts de Pareto $F_1, ..., F_p$, nous utilisons une procédure qui fait appelle plusieurs fois à la méthode $Front\_Pareto\left(P\right)$. La Figure 4.18 en présente l'algorithme.

Calcul de l'ensemble de solutions non dominées dans une population de solutions

Front_Pareto (P)
Trier la liste P selon le premier objectif Z_1
Si$(|P| = 1)$
  Retourner(P)
Sinon
  LF=*Front_Pareto* $(P_1, ..., P_{\lfloor |P|/2 \rfloor})$
  UF=*Front_Pareto* $(P_{\lfloor |P|/2 \rfloor + 1}, ..., P_{|P|})$
  $i = 1$, $B = \emptyset$
  Tantque $(i \leq |UF|)$
    $j = 1$
    Tan tque $(j \leq |LF|)$
      Si $(X_j$ `ne domine pas` $X_i)$
        $j = j + 1$
      Sinon Stop
    Fin Tant que
    Si $(j = |LF|)$
      $B = B \cup \{i\}$
    Fin Si
    $i = i + 1$
  Fin Tant que
Retourner$(LF \cup B$

FIGURE 4.17 – Algorithme de calcul des solutions non-dominées dans une population de solutions ([Kung *et al.*, 1975])

La procédure pour trier une population sous forme de sous ensembles non-dominées

NonDominatedSorting (P)
$j = 1$
Tant que $(P \neq \emptyset)$
  $F' = Front\_Pareto\,(P)$
  $F_j = F'$
  $P = P \backslash F'$
  $j = j + 1$
Fin Tant que
Retourner$(F_i, \quad i = 1, 2, ..., g)$

FIGURE 4.18 – Une procédure pour trier une population de solutions sous forme de sous-ensembles non-dominées

4.2.5 Les opérateurs de reproduction

Dans les algorithmes génétiques on distingue deux types d'opérateurs de reproduction : les opérateurs de croisement et les opérateurs de mutation. Les opérateurs de croisement sont pour la plupart des opérateurs binaires (entre deux solutions) : ils consistent à prélever les caractéristiques des deux solutions parent afin de générer une ou deux solutions enfant. Les opérateurs de mutations sont quant à eux unaires car ils agissent sur une solution unique : ils consistent à faire une légère modification de certains individus de la population en utilisant une heuristique simple (Swap, 2-Opt,...).

4.2.5.1 L'opérateur de croisement

Le croisement est le processus de création de deux nouveaux individus (appelés Enfants) en échangeant un ou plusieurs gènes de deux individus existants (appelés Parents). Après avoir sélectionné deux chromosomes C et C' l'opérateur de croisement est appliqué pour générer deux autres individus C_f et C'_f avec une probabilité p_N^C. Nous appliquons l'opérateur de croisement PMX (*Parially Mapped Crossover*) qui a été proposé dans [Goldberg et ., 1985]. Cet opérateur est appliqué sur deux tournées parents choisies aléatoirement, une tournée dans C et l'autre dans C'.

Le but est de construire deux tournées enfants par le choix de sous-séquences ordonnancées de l'une des tournées parents et de préserver l'ordre et la position d'autant de séquences que possibles des autres tournées parents. Les sous-séquences sont créées par le choix aléatoire de deux points de coupures dans chaque tournée. Ces points servent de frontière pour l'opération de substitution.

Il faut noter que dans le cas de notre problème l'ordre des séquences n'est pas toujours conservé dans le processus de croisement. Ceci est dû essentiellement à la contrainte de précédence, entre le sommet de départ et d'arrivée d'une même requête qu'il faut respecter, ainsi que la capacité maximale du véhicule qu'il ne faut pas dépasser. Pour illustrer cet opérateur de croisement, nous utilisons dans la suite un exemple de 7 requêtes et 3 tournées : les sommets de 1 à 7 représentent les points de départs et les sommets de 8 à 14 représentent les points de destination (voir Figure 4.19).

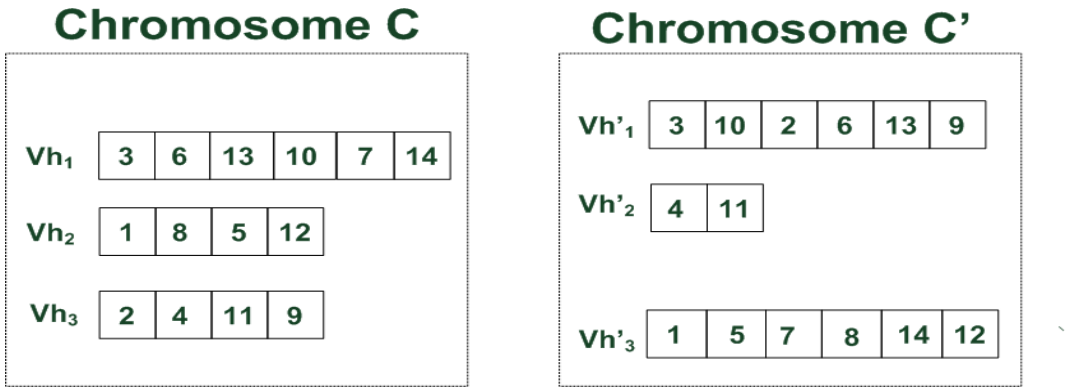


FIGURE 4.19 – Exemple de deux chromosomes C et C'

Premièrement, nous sélectionnons aléatoirement deux tournées, une tournée de chaque chromosome, par exemple Vh_1 et Vh'_3 (voir Figure 4.20). Ensuite, deux points pour chaque individus sont générés aléatoirement pour diviser la tournée Vh_1 et Vh'_3 en trois parties (voir Figure 4.21). Ces six parties sont combinées pour construire deux nouvelles tournées Vf_1 et Vf'_3 comme illustré dans la Figure 4.22.

La tournée Vf_1 (resp.Vf'_3) qui résulte du croisement remplace la tournée Vh_1 (resp. Vh'_3) pour donner deux chromosomes fils C_f et C'_f (voir Figure 4.23). En passant du chromosome C et C' vers C_f et C'_f seules les tournées Vh_1 et Vh'_3 ont été changées et les autres tournées restantes n'ont pas été touchées.

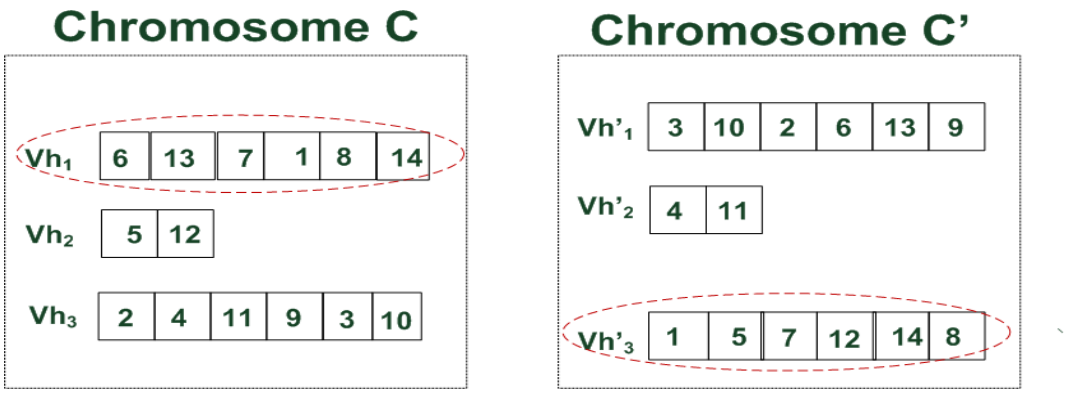


FIGURE 4.20 – Sélectionner deux tournées, par exemple : Vh_1 et Vh_3

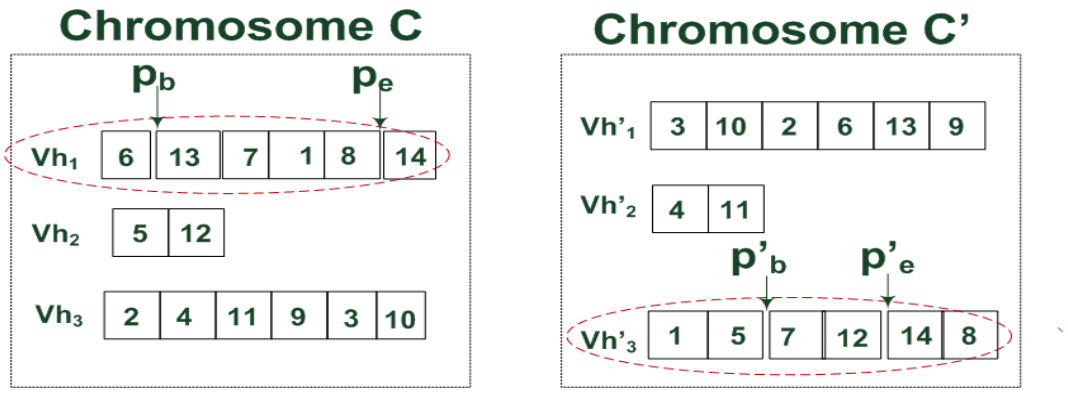


FIGURE 4.21 – Choisir deux points de croisement dans chaque tournée

Pour un problème fortement contraint, tel que le DARP, il est très difficile de maintenir la faisabilité de la solution après chaque croisement. L'opérateur de croisement peut conduire à des incohérences dans les individus Enfants qui doivent alors être corrigées. Dans la suite, nous expliquons le processus de correction que nous avons suivi pour réparer les deux chromosomes qui résultent de l'opération de croisement.

– **Suppressions des doublons dans les tournées Vf_1 et Vf'_3 :**

Pour l'exemple illustré dans la Figure 4.23, les sommets v_1 et v_8 sont deux doublons dans la tournée Vf_1, tandis que la tournée Vf'_3 ne contient aucun doublon. On parcourt la tournée sommet par sommet et on supprime un des sommets doublon.

– **Ajouter les sommets manquants dans les tournées Vf_1 et Vf'_3 :**

La Figure 4.25 montre un exemple de sommets manquants dans chaque tournée ainsi que leurs insertions. Dans la tournée Vf_1 il manque les sommets v_6 et v_{12} et dans la tournée Vf'_3 il manque les sommets v_5 et v_{13}.

Pour compléter ces tournées il faut insérer les sommets manquants tout en respectant la contrainte de couplage et de précédence pour les sommets à insérer (le sommet de départ doit être placé avant le sommet d'arrivée pour chaque requête). On commence par insérer les sommets de destination manquants dans chaque tournée, la Figure 4.26 présente l'algorithme utilisé pour insérer un sommet de destination v_d dans une tournée Vh_k. Notons

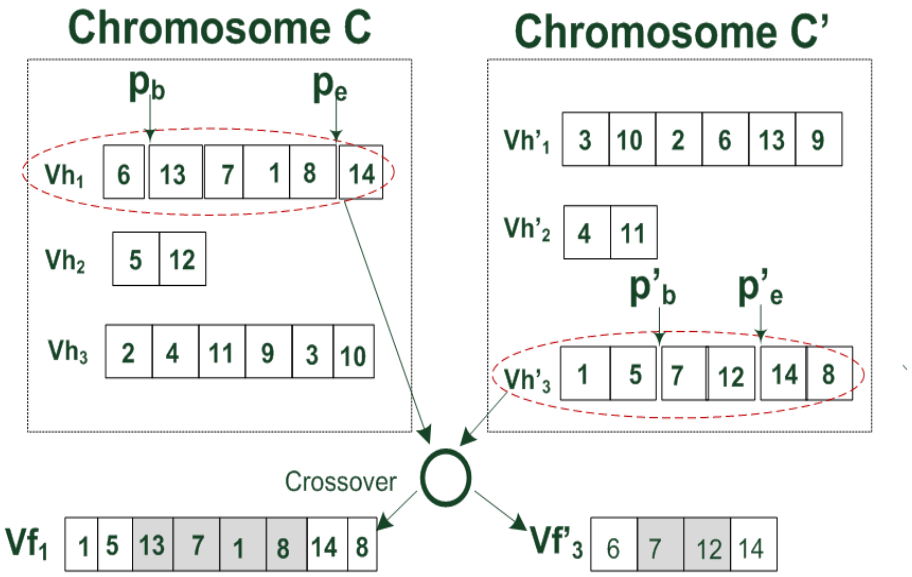


FIGURE 4.22 – Croisement de deux tournées

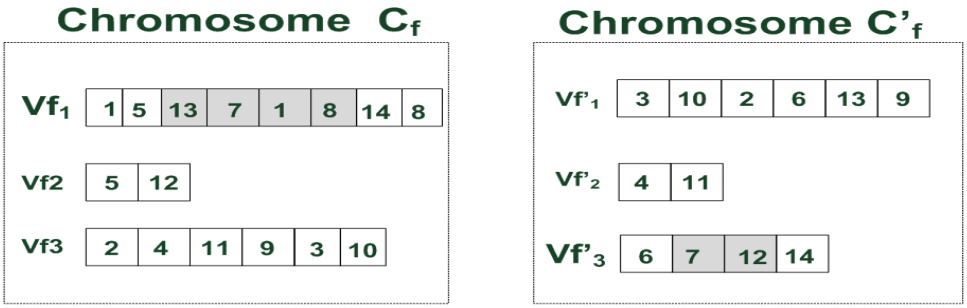


FIGURE 4.23 – Les deux chromosomes fils

que lors de l'insertion, la contrainte de capacité du véhicule n'est pas vérifiée car la tournée est partielle et il faut d'abord la compléter avant de procéder à sa vérification vis-à-vis de cette contrainte.

Une fois les sommets de destination manquants insérés, nous procédons à l'insertion des sommets de départ manquants. La Figure 4.27 présente l'algorithme utilisé pour l'insertion de ces sommets.

Nous procédons ensuite à la correction de chaque tournée vis-à-vis de la contrainte de précédence : si pour une requête donnée le sommet de départ est placé avant le sommet de destination alors une permutation de la position de ces deux sommets est effectuée. Ensuite, nous corrigeons la tournée vis-à-vis de la contrainte de capacité du véhicule. Les mêmes procédures de corrections (voir Figure 4.5 et Figure 4.6) présentées dans la Section (4.2.3) sont réutilisées pour corriger les tournées.

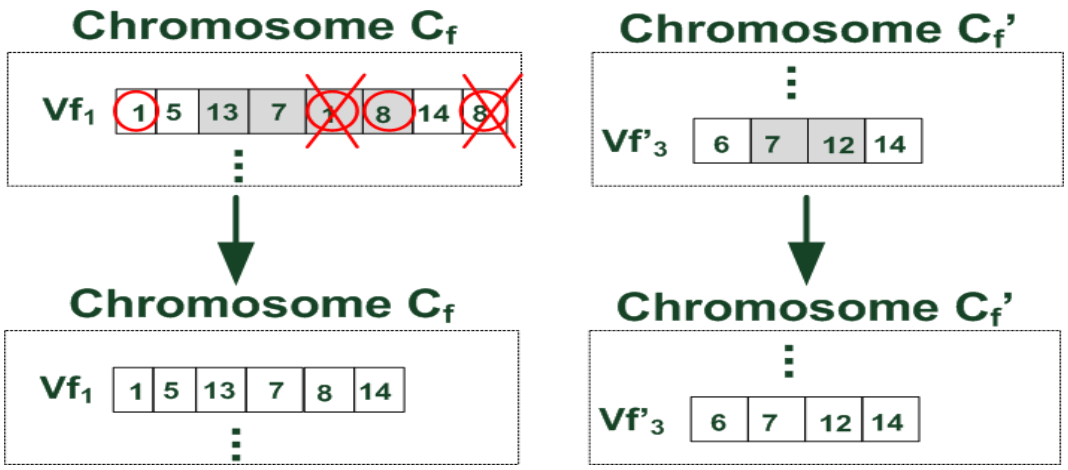


FIGURE 4.24 – Suppression des doublons

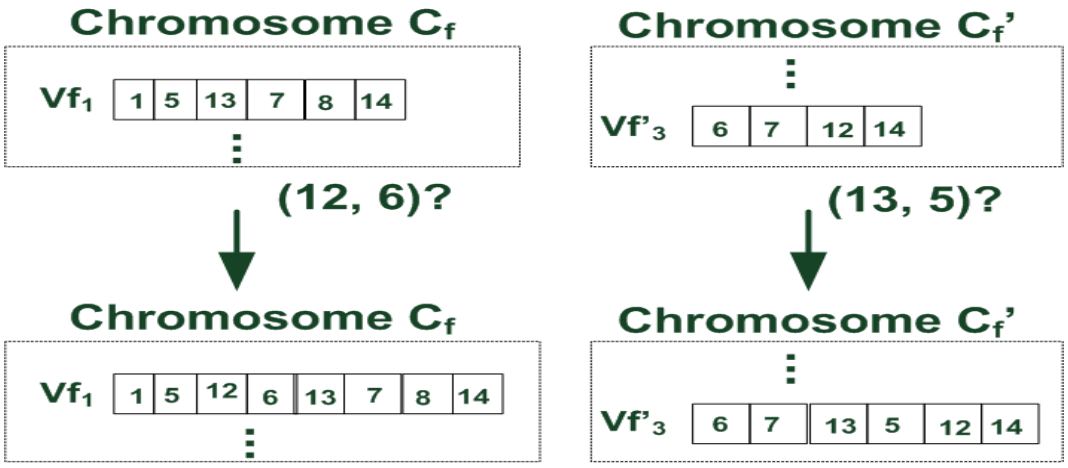


FIGURE 4.25 – Insertion des sommets manquants

Maintenant que les deux tournées qui résultent du croisement sont réalisables, nous passons à la vérification et la correction des deux chromosomes fils C_f et C'_f, notamment l'élimination des requêtes redondantes ou l'insertion des requêtes manquantes.

– **La suppression des doublons dans les chromosomes C_f et C'_f :**

Dans l'exemple de la Figure 4.28, la requête r_5 (v_5, v_{12}) est redondante dans le chromosome C_f car elle figure dans la tournée Vf_1 et Vf_2. Cette requête doit être supprimée de la tournée Vf_2. De la même façon la requête r_6 (v_6, v_{13}) est redondante dans le chromosome C'_f, elle est donc supprimée de la tournée Vf'_1.

– **L'insertion des requêtes manquantes dans les chromosomes C_f et C'_f :**

La Figure 4.29 illustre l'exemple de requêtes manquantes ainsi que leur insertion dans chaque chromosome. Les requêtes manquantes dans un chromosome sont celles qui figurent dans la tournée choisie pour le croisement et qui ne figurent pas dans la tournée résultante. Dans l'exemple illustré dans la Figure 4.29, la requête r_1 (v_1, v_8) est manquante pour le

Insertion d'un sommet de destination manquant dans une tournée

Entrée :
- Vh_k : une tournée de p sommets ;
- v_d : le sommet de destination à insérer ;
- $i = 1$;
- n : le nombre de requêtes ;

Tant que $(i \leq p)$ *Faire*
- Si $Vh_k[i] = v_{d-n}$ // $Vh_k[i]$ est le sommet de départ correspondant à v_d
  - Choisir une position aléatoire *posInsert* entre $i+1$ et $p+1$;
  - Insérer le sommet v_d à la position *posInsert* ;
  - Stop (arrêter la boucle Tant que) ;
- Sinon
  - $i = i + 1$;
- Fin SI

Fin Tant que

Sortie : Tournée Vh_k

FIGURE 4.26 – Procédure d'insertion d'un sommet de destination v_d dans une tournée Vh_k

Insertion d'un sommet de départ manquant dans une tournée

Entrée :
- Vh_k : une tournée de p sommets ;
- v_d : le sommet de départ à insérer ;
- $i = 1$;
- n : le nombre de requêtes ;

Tant que $(Vh_k[i] = v_{d+n})$ Faire
- $i = i + 1$

Fin Tant que
- Choisir une position aléatoire *posInsert* entre 1 et i ;
- Insérer le sommet v_d à la position *posInsert* ;

Sortie : Tournée Vh_k

FIGURE 4.27 – Procédure d'insertion d'un sommet de départ v_d dans une tournée Vh_k

chromosome C'_f car elle été présente dans la tournée Vh'_3 et ne figure pas dans la tournée résultante Vf'_3.

La procédure d'insertion d'une requête manquante dans un chromosome consiste à insérer d'abord le sommet de départ dans l'une des tournées qui composent le chromosome et ensuite insérer le sommet de destination. Lors de l'insertion du sommet de départ v_j, on choisit une position dans la tournée entre t et $t+1$ pour laquelle $l_j \in \left[l_{[t]}, l_{[t+1]}\right]$, $\forall t \in \{0, 1, ..., p\}$ et la distance supplémentaire engendrée par cette insertion est minimale tout en respectant la contrainte du capacité de véhicule. Si aucun intervalle de temps n'est trouvé alors le sommet de départ est inséré à la fin de la tournée. La Figure 4.30 présente l'algorithme d'insertion d'un sommet de départ dans une tournée.

Dans cet algorithme on utilise les notation suivantes :
- I : la position d'insertion du sommet de départ dans la tournée,
- *distance* : représente la distance supplémentaire qui résulte de l'insertion du sommet de départ dans une position donnée,

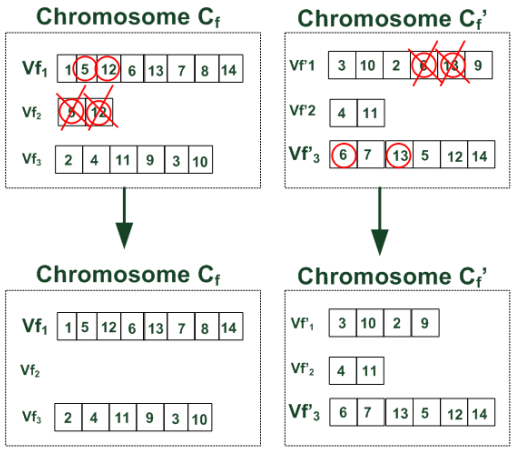


FIGURE 4.28 – Elimination des requêtes redondantes dans les chromosomes C_f et C'_f

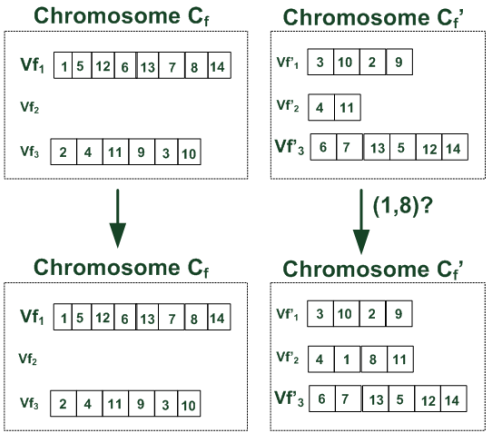


FIGURE 4.29 – Compléter chaque chromosome par les requêtes manquantes

- *Load* : une variable qui calcule le nombre de personnes à bord du véhicule à chaque sommet servi dans une tournée,
- *minDist* : la distance minimale engendrée par la meilleure insertion.

Il faut noter que l'algorithme d'insertion est appliqué sur toutes les tournées qui composent le chromosome et la tournée qui a la plus petite augmentation de la distance est choisie pour accueillir le sommet.

Le sommet de destination est inséré évidement dans la tournée qui a été sélectionnée précédemment pour accueillir le sommet de départ. La Figure 4.31 présente l'algorithme utilisé pour l'insertion d'un sommet de destination v_{j+n} dans une tournée Vh_k.

4.2.5.2 L'opérateur de mutation

La mutation est le processus utilisé pour changer le chromosome obtenu après l'application de l'opérateur de croisement : elle permet d'introduire de la diversification dans la

Insertion d'un sommet de départ dans une tournée

Entrée :
- Vh_k : une tournée de p sommets associée à un véhicule k de capacité Q_k ;
- v_j : le sommet de départ à insérer ;
- $minDist = \infty$;
- n : le nombre de requêtes ;
- $Load = 0$ et $I = -1$;

Pour $i = 1$ à p Faire
- $Load = Load + q_{[i]}$
- Si ($l_j \in \left[l_{[i-1]}, l_{[i]}\right]$ et $Load + q_j \leq Q_k$)
  - distance $= d_{[i-1]j} + d_{j[i]} - d_{[i-1,i]}$
  - Si ($distance < minDist$) //
    - $minDist = distance$
    - $I = i$
- Fin Si

Fin Pour
Si $I = -1$
- $I = p$ //Insertion à la fin de la tournée
- $minDist = d_{[p]j} + d_{j0} - d_{[p]0}$

Fin Si

Sortie : $minDist$ et I

FIGURE 4.30 – Procédure d'insertion d'un sommet de départ v_j dans une tournée Vh_k

Insertion d'un sommet de destination dans une tournée

Entrée :
- Vh_k : une tournée de p sommets qui contient déjà le sommet de départ v_j ;
- v_{j+n} : le sommet de destination à insérer ;
- $i = 1$;
- n : le nombre de requêtes ;
- $minDist = \infty$;
- $I = -1$;

Tant que($Vh_k[i]! = v_j$)
- $i = i + 1$

Fin Tant que
Pour $t = i + 1$ à p Faire
- Si ($l_{j+n} \in \left[l_{[t-1]}, l_{[t]}\right]$)
  - $distance = d_{[t-1],j+n} + d_{j+n,[t]} - d_{[t-1,t]}$
  - Si $distance < minDist$
    - $minDist = distance$
    - $I = t$

Fin Pour
Si $I = -1$
- $I = p + 1$ // Insertion à la fin de la tournée
- $minDist = d_{p,j+n} + d_{j+n,0} - d_{p0}$

Fin Si

Sortie : Tournée Vh_k

FIGURE 4.31 – Insertion d'un sommet de destination v_{j+n} dans une tournée Vh_k

population et elle est appliquée sous une probabilité p_N^M. Il est recommandé de choisir une valeur faible pour cette probabilité afin d'éviter de tomber dans une recherche aléatoire et de conserver les concepts de sélection et d'évolution de NSGA-II. La mutation sert à

éviter une convergence trop rapide de l'algorithme génétique vers un minimum local. Dans la suite nous présentons les étapes principales de l'opérateur que nous avons utilisé dans le cas de notre problème : étant donné un chromosome C_f, on commence par sélectionner aléatoirement deux tournées. Ensuite on sélectionne une requête de la première tournée. Si le nombre de personnes de la requête sélectionnée est inférieur à la capacité du véhicule associé à la deuxième tournée alors la requête est insérée dans cette dernière. Sinon on sélectionne une requête de la deuxième tournée sélectionnée que nous insérons dans la première tournée sélectionnée. L'insertion est effectuée en utilisant les mêmes procédures présentées dans Les Figure 4.30 et 4.31 de la Section 4.2.5.1. La Figure 4.32 présente un exemple de l'application de l'opérateur de mutation.

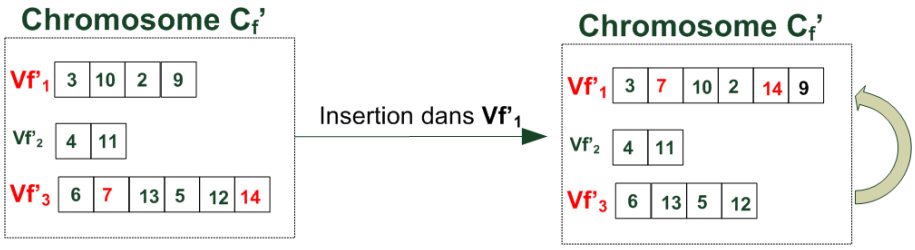


FIGURE 4.32 – Exemple de l'opérateur de mutation appliqué sur le chromosome Cf'

4.2.6 Stratégie de la préservation de la diversité

Calculer une approximation de l'ensemble des optima de Pareto n'est pas uniquement une question de convergence, mais aussi de répartition des solutions dans l'espace des critères. Afin de garantir une diversification des solutions le long du front de Pareto, un mécanisme de préservation de la diversité est intégré au sein NSGA-II. Il existe plusieurs stratégies d'affectation d'une valeur de diversité, habituellement basée sur une mesure de distance entre les vecteurs de critères (exemple de la distance de *Crowding*).

Cette valeur consiste à estimer la densité des solutions autour d'un point particulier de l'espace des critères à l'aide d'une mesure de surpeuplement (distance de*Crowding*). Cette valeur représente la qualité de chaque solution en terme de diversité.

Notons que la distance de *Crowding* est calculée pour les solutions qui appartiennent au même front de Pareto. Elle est illustrée dans la Figure 4.33.

Une solution est considérée meilleure qu'une autre au sein d'un même front, si elle a la meilleure valeur de distance de *Crowding* (de plus grande valeur). La Figure 4.34 présente l'algorithme de calcul de la distance de *Crowding* pour l'ensemble des solutions d'un front de Pareto donné.

4.2.7 Sélection des solutions

Le processus de sélection des chromosomes est appliqué deux fois : lors de la sélection de deux chromosomes dans une population P_t pour le croisement et la mutation et lors de la sélection des chromosomes pour compléter une population P_{t+1} à partir d'une population $P_t \cup Q_t$.

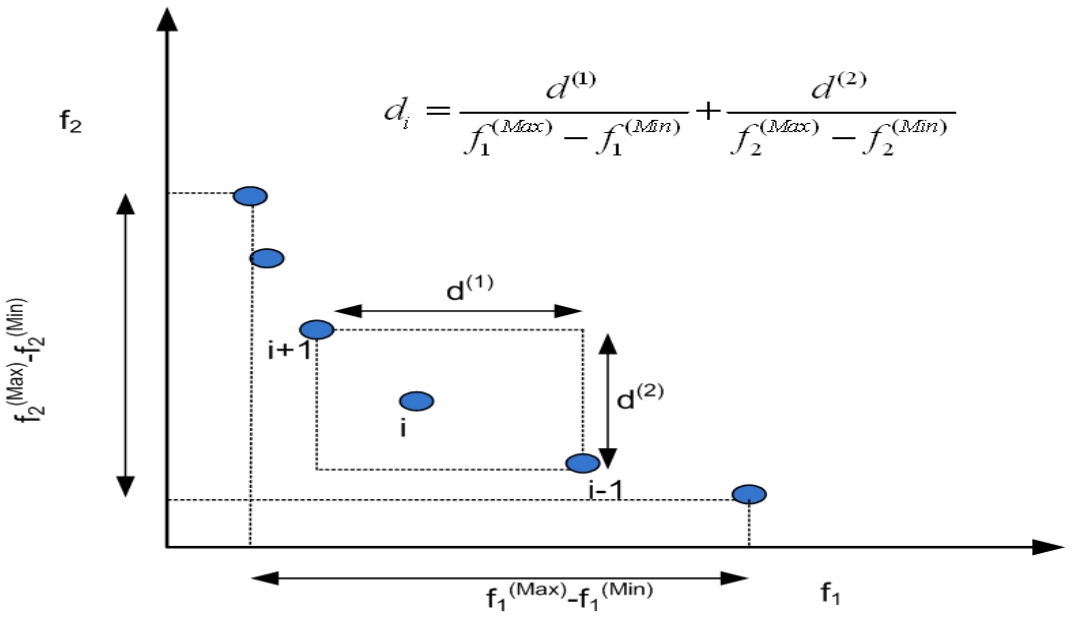


FIGURE 4.33 – Illustaration de distance de Crowding

Calcul de la distance de *Crowding* pour les solutions qui composent un front de Pareto F

Entrée :
- $T = |F|$: le nombre d'individus du front F ;
- $d_i = 0, \forall i \in \{1, ..., T\}$: la distance de *Crowding* pour chaque solution i de F ;
- $m = 1$: le compteur d'objectifs (on suppose qu'on a M objectifs) ;

Etape 1 :
Trier l'ensemble des solutions de F dans une liste I^m selon
- la fonction objectif Z_m par ordre décroissant ;

Etape 2 :
Mettre à jour les valeurs de d_i :
- $d_{I_i^m} = d_{I_i^m} + \frac{f_m^{I_{i+1}^m} - f_m^{I_{i-1}^m}}{f_m^{Max} - f_m^{Min}}, \forall i = 1, ..., T$;

Si $m == M$;
- Fin ;

Sinon
- $m = m + 1$ et retourner à l'Etape 1 ;

FinSi
Sortie : $d_{I_i^m}, \forall i = 1, ..., T$

FIGURE 4.34 – Calcul de la distance de *Crowding*

4.2.7.1 Sélection de deux chromosomes pour la reproduction

La sélection de deux chromosomes pour le croisement utilise une sélection par roulette classique basée sur la valeur de fitness de chaque chromosome (voir Figure 4.35) . Plus la valeur de fitness d'un chromosome est petite, plus sa chance d'être sélectionné aléatoirement est grande. Nous donnons plus de chance aux individus qui appartiennent au premier front d'être sélectionnées qu'aux individus de deuxième front et ainsi de suite. Si la population P_t est constituée d'un seul front alors les individus de cette population auront la même

chance d'être sélectionnés aléatoirement.

Plus précisément, cette méthode consiste à associer à chaque chromosome un segment dont la longueur est inversement proportionnelle au numéro du front de Pareto auquel il appartient. Ces segments sont ensuite concaténés sur un axe gradué que l'on normalise entre 0 et 1. Un tirage aléatoire est effectué pour sélectionner un nombre entre 0 et 1, puis on identifie le segment sélectionné et le chromosome correspondant.

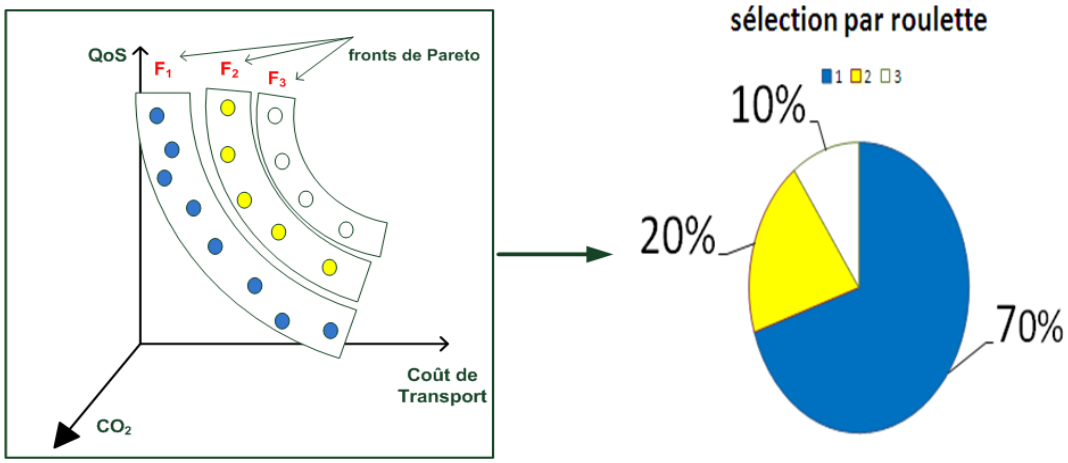


FIGURE 4.35 – Exemple de processus de sélection basé sur la roulette

4.2.7.2 Sélection de N chromosomes à partir de la population $P_t \cup Q_t$

Pour créer une population P_{t+1}, le processus de sélection de N chromosomes à partir de la population $P_t \cup Q_t = \bigcup_{j=1}^{g} F_j$ de taille $2N$ consiste à choisir les solutions du premier front de Pareto F_1, ensuite les solutions du deuxième front F_2 et ainsi de suite jusqu'à remplir la population P_{t+1}. Si les solutions d'un front F_i ne peuvent pas être toutes ajoutées dans la population alors les solutions de ce front sont triées par ordre décroissant de leur valeurs de la distance de *Crowding*. Elles sont donc ajoutées dans cet ordre jusqu'à compléter la population P_{t+1}.

4.2.8 Condition d'arrêt

Les critères d'arrêt utilisés dans les algorithmes évolutionnaires mono-objectifs sont repris dans le cas multiobjectifs. Les exemples les plus classiques consistent à stopper l'algorithme évolutionnaire au bout d'un certain nombre d'itérations ou un certain temps d'exécution. Une autre méthode utilisée consiste à arrêter le processus de recherche au bout d'un nombre d'itérations sans amélioration. Mais dans le cas multiobjectifs il est difficile de définir ce qu'est une amélioration d'une itération à l'autre. Dans le cas de notre problème, l'algorithme NSGA-II est arrêté après un nombre donné d'itérations.

4.3 Introduction aux expérimentations

Nous allons nous focalisons sur l'évaluation de la performance de l'algorithme NSGA-II qui a été proposé pour résoudre le problème de DARP statique multicritère. Nous commençons par la description du protocole de génération des instances aléatoires. Puis nous présentons les indicateurs de qualité utilisés pour l'évaluation de la performance des algorithmes d'optimisation multiobjectifs. Ensuite, nous présentons la méthodologie que nous avons suivi pour régler les paramètres de NSGA-II. Nous présentons aussi les résultats obtenus avec des instances générées aléatoirement et une comparaison avec les résultats obtenus dans la littérature sur des problèmes similaires mais mono-objectif .

Nous avons généré aléatoirement trente groupes d'instances et chaque groupe contient 10 instances générées aléatoirement et est caractérisé par : un nombre n de requêtes avec $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200\}$ et un nombre M de véhicules, avec $M \in \{2, 4, 6, 8, 10, 12\}$. Il faut noter que nous disposons de cinq types de véhicules, chaque véhicule étant caractérisé par une capacité, un coût de transport kilométrique et une quantité de CO_2 émise. La Table 4.3 regroupe les caractéristiques des véhicules qui ont été utilisés pour générer les instances.

Véhicule	Capacité	Coût (Euro/Km)	Quantité de CO_2 émise (g/km)
1	4	400	200
2	9	200	350
3	18	500	400
4	27	400	420
5	50	400	480

TABLE 4.3 – Différentes categories de véhicules

Le sommet de départ v_i et le sommet d'arrivée v_{i+n} d'une requête r_i sont générés aléatoirement dans le carré $[-20, 20]^2$ selon une distribution uniforme. Le dépôt v_0 est localisé au centre de ce carré, donc de coordonnées (0,0). Les temps de service s_i et s_{i+n} valent tous 10 et le nombre de personnes q_i est généré aléatoirement entre 1 et 5. La fenêtre de temps $[e_i, l_i]$ associée au sommet v_i est générée de la manière suivante : la borne inférieure e_i est générée dans l'intervalle $[0, 1000]$ et la borne supérieure l_i est égale à e_i plus une quantité générée aléatoirement entre $[10, 30]$.

La fenêtre de temps $[e_{i+n}, l_{i+n}]$ associée au sommet de destination v_{i+n} est générée à partir de l'intervalle $[e_i, l_i]$ de la façon suivante : e_{i+n} est égale à $e_i + 1,5t_{i,i+n}$ plus une quantité aléatoire entre 10 et 30, où $t_{i,i+n}$ est le temps de trajet entre les sommets v_i et v_{i+n}. Enfin l_{i+n} est défini par e_{i+n} plus une quantité aléatoire entre 10 et 30. Notons que dans ces instances on considère que le temps du trajet $t_{i,j}$ entre le sommet v_i et v_j est égale à la distance euclidienne d_{ij} qui sépare ces deux sommets.

Il faut noter que dans toutes les instances, la fenêtre de temps associée au dépôt est égale à $[0, 1440]$ et à chaque sommet v_i sont associées deux pondérations α_i et β_i qui apparaissent dans la définition du critère de l'insatisfaction des passagers $\overline{QoS}$: α_i pour pénaliser les avances et β_i pour pénaliser les retards. Rappelons que l'avance n'est autorisée que pour les sommets d'arrivée, alors que pour les sommets de départ aucune avance n'est

considérée : pour un sommet de départ, aucun α_i n'est utilisé. Notons que dans tous les tests qui suivent, les poids α_i et β_i sont générés aléatoirement à partir des valeurs suivantes : 1, 10, 100, 1000, 10000, 20000, 30000. Le paramètre w_j est calculé selon la position du sommet v_j dans la tournée et w_j est égale à la somme des passagers à bord du véhicule au sommet v_j.

L'ensemble des tests ont été effectués avec une machine Intel Core Duo CPU 2.2 Ghz avec 3 Go de RAM et tous les algorithmes ont été développés avec le langage C++.

4.4 Les indicateurs de qualité pour l'évaluation des méthodes multiobjectifs

L'évaluation des méthodes multiobjectifs est une tâche plus difficile que celle des méthodes monocritères. Plusieurs indicateurs peuvent être trouvés dans la littérature ([Zitzler *et al.*, 2000]) pour mesurer la qualité des ensembles de solutions non-dominées obtenus avec des méthodes d'optimisation multiobjectifs. Ces indicateurs reflètent la qualité d'un ensemble de solutions selon plusieurs aspects : la convergence, la diversité, le nombre de solutions, etc.

Nous avons développé certains de ces indicateurs pour l'évaluation de l'algorithme NSGA-II. Dans la suite nous présentons en détail trois indicateurs de qualité Q_1, Q_2 et Q_3 qui permettent de comparer deux sous-ensembles de solutions, un sous-ensemble approché A et un sous-ensemble de référence ZE^*.

4.4.1 Calcul de l'indicateur Q_1

Cette mesure est l'une des mesures les plus naturelles : elle consiste à calculer le pourcentage de solutions de ZE^* qui ont été trouvées par l'algorithme soumis à l'évaluation. L'ensemble ZE^* peut être l'ensemble exact des optima de Pareto stricts , comme il peut être la meilleure approximation connue de cet ensemble, pour le problème que nous voulons résoudre. Cet ensemble de solutions est appelé aussi *front de Pareto de référence*. L'indicateur de qualité Q_1 a été utilisé par nombreux auteurs dans la littérature. Il est défini par la formule suivante :

$$Q_1(A) = 100 \times \frac{|ZE^* \cap A|}{|ZE^*|}$$

Où ZE^* est le front de Pareto de référence et A le front de Pareto à évaluer. La Figure 4.36 illustre un exemple de calcul de cet indicateur.

L'inconvénient majeur de cette mesure est qu'elle nécessite la connaissance du front de référence exact qui n'est pas souvent facile à calculer. Si ce front n'est pas connu une bonne approximation de ce front peut être utilisée. Cependant si une telle approximation est utilisée comme front de référence il peut arriver que l'ensemble des solutions soumis à l'évaluation contienne des solutions non-dominées qui ne figurent pas dans le front de référence.

4.4.2 Calcul de l'indicateur Q_2

Nous avons utilisé un autre indicateur de qualité Q_2 qui a été proposé par [Veldhuizen, 1999] et qui est similaire à Q_1. Il représente le pourcentage de solutions de l'ensemble approximatif A qui sont non-dominées par les solutions de l'ensemble de référence (voir Figure 4.36). Cette mesure permet d'avoir une idée sur la quantité de solutions non-dominées qui ont été générées par rapport aux solutions dominées. La mesure Q_2 est définie par la formule suivante :

$$Q_2(A) = 100 \times \frac{|ZE^* \cap A|}{|A|} \ .$$

Plus Q_2 est proche de 1, plus l'ensemble soumis à évaluation contient une majorité de solutions non-dominées que de solutions dominées.

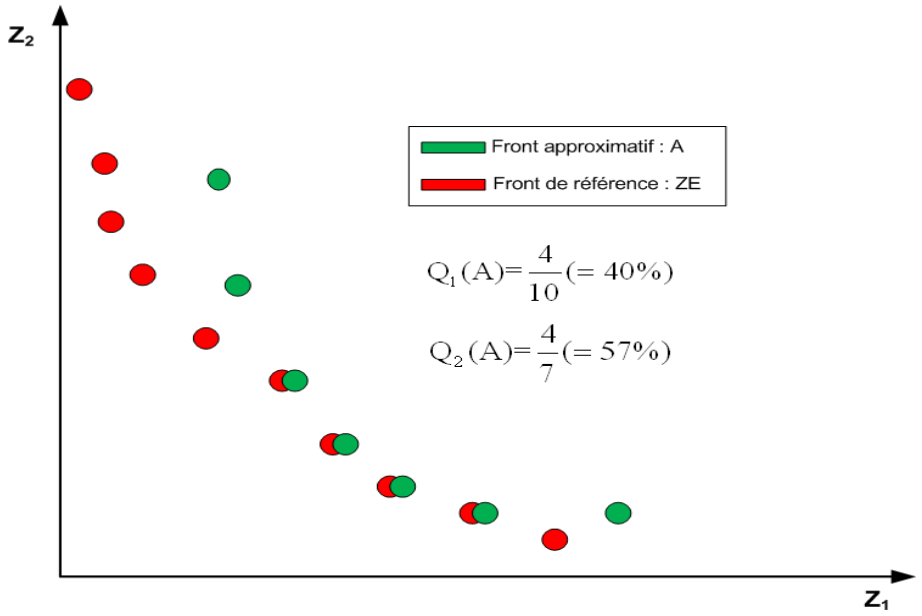


FIGURE 4.36 – Exemple de calcul des indicateurs de qualité Q_1 et Q_2

4.4.3 Calcul de l'indicateur Q_3

Il existe d'autres catégories de mesures qui tendent à mesurer une distance entre deux sous-ensembles de solutions (entre l'ensemble approximatif et l'ensemble de référence). Dans la littérature plusieurs types de métriques ont été proposés pour calculer cette distance. Dans la suite, nous présentons une mesure qui a été proposée par [Jaszkiewicz, 2004], notée Q_3, et définie par la formule suivante :

$$Q_3(A) = \frac{1}{|ZE^*|} \sum_{r \in ZE^*} min_{z \in A} \{d(z, r)\}$$

Où $d()$ représente la distance euclidienne dans l'espace des objectifs.

$Q_3(A)$ représente la distance moyenne entre chaque solution du front de référence ZE^* et la solution qui lui est plus proche dans l'ensemble approximatif A. Plus la valeur de Q_3 est petite, plus le front approximatif est proche du front de référence. Cette mesure a été utilisée dans la littérature par plusieurs autres auteurs ([Van Veldhuizen et Lamont, 1998], [Viana et Pinho de Sousa, 2000] et [Ulungu *et al.*, 1999]). La Figure 4.37 montre un exemple du calcul de l'indicateur de qualité Q_3.

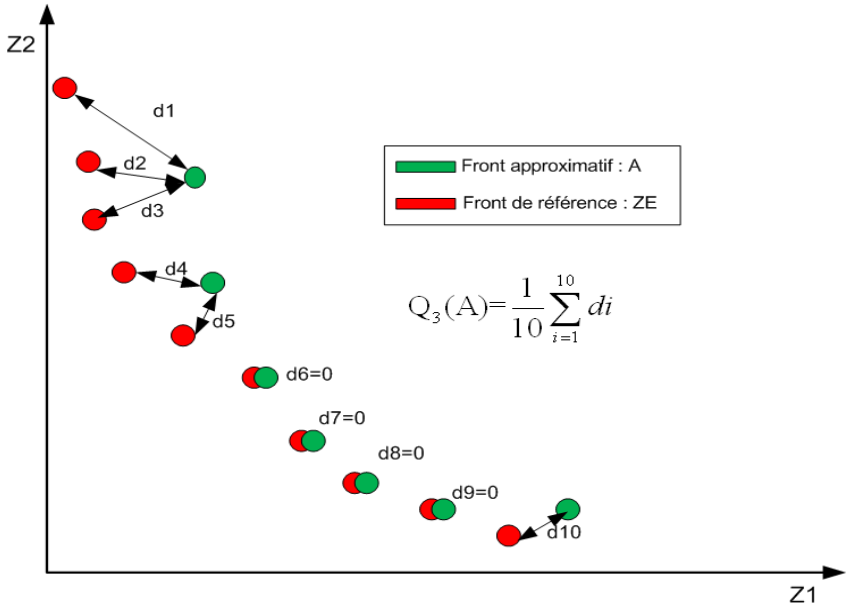


FIGURE 4.37 – Exemple de calcul des indicateurs de qualité Q_3

Notons que cet indicateur est normalisé dans tous les résultats que nous allons fournir dans les sections suivantes. La normalisation consiste à diviser les valeurs de Q_3 par la valeur maximale obtenue.

4.5 Comparaison des performances des trois heuristiques

Nous cherchons maintenant à identifier la meilleure combinaison des trois heuristiques H1, H2 et H3 pour initialiser la population dans l'algorithme NSGA-II. Ces trois heuristiques ont été testées afin de trouver l'heuristique qui contribue le plus à l'accélération de la convergence de NSGA-II vers un front de meilleure qualité. Lors de la génération de la population initiale, un pourcentage de solutions est généré par chaque heuristique. Nous avons testé 231 combinaisons de pourcentages.

NSGA-II est exécuté pour chaque combinaison de pourcentages de solutions des trois heuristiques. Pour exécuter l'algorithme NSGA-II, nous avons utilisé les paramètres suivant : une probabilité de croisement de 100%, une probabilité de mutation de 15%, une population de taille 200 et l'algorithme est exécuté pour 1500 itérations.

Pour une instance donnée, le front de Pareto qui résulte du test de l'ensemble des combinaisons de pourcentages des trois heuristiques est utilisé comme ensemble de référence

ZE^* dans le calcul des indicateurs de qualité Q_1, Q_2 et Q_3. La Table 4.4 montre les résultats obtenus avec les différentes combinaisons de pourcentages. La table ne contient que quelques combinaisons pour lesquelles les résultats obtenus sont significatifs, c'est-à-dire les combinaisons qui ont permis de générer des solutions qui n'ont pas été toutes dominées par le front de référence ZE^*. Ces résultats sont obtenus en utilisant l'instance de 40 requêtes et 5 véhicules.

Les trois premières colonnes contiennent le pourcentage de solutions, dans la population initiale, générés par les heuristiques H1, H2 et H3. Les trois dernières colonnes représentent dans l'ordre les trois indicateurs de qualité Q_1, Q_2 et Q_3.

Pourcentages			Indicateurs de qualité			Pourcentages			Indicateurs de qualité		
$\%H_1$	$\%H_2$	$\%H_3$	Q_1	Q_2	Q_3	$\%H_1$	$\%H_2$	$\%H_3$	Q_1	Q_2	Q_3
5	90	5	6.00	9.00	0.63	45	50	5	2.00	4.00	0.51
5	45	50	0.00	0.00	0.43	45	5	50	0.00	0.00	0.33
5	25	70	14.00	23.00	0.67	50	5	45	1.00	2.00	0.28
5	**5**	**90**	**36.00**	**54.00**	**0.57**	50	25	25	0.00	0.00	0.62
10	70	20	9.00	18.00	0.64	50	45	5	0.00	0.00	1.00
10	40	50	0.00	0.00	0.47	60	10	30	0.00	0.00	0.76
10	30	60	9.00	15.00	0.62	70	25	5	7.00	13.00	0.60
20	10	70	0.00	0.00	0.34	70	10	20	4.00	7.00	0.52
25	70	5	2.00	3.00	0.56	100	0	0	1.00	2.00	0.51
25	25	50	0.00	0.00	0.41	0	100	0	7.00	9.00	0.68
35	30	35	2.00	4.00	0.32	0	0	100	10.00	21.00	0.26

TABLE 4.4 – Les résultats obtenus en combinant H_1, H_2 et H_3

Plus de résultats sur la comparaison des trois heuristiques sur des instances de taille différente sont donnés en Annexe de ce manuscrit, Table **??**, **??** et **??**.

Selon les résultats de la Table (4.4) il est préférable d'utiliser une combinaison qui permet de générer plus de solutions avec l'heuristique H3. Dans la suite nous avons choisi d'utiliser la combinaison suivante : 5 % de solutions sont générées par l'heuristique H1, 5% par H2 et 90% par H3.

4.6 Calibrage des paramètres de l'algorithme NSGA-II

Dans cette section nous allons présenter les tests expérimentaux qui ont été effectués pour régler les paramètres de l'algorithme NSGA-II.

Dans la littérature, la plupart des chercheurs qui utilisent l'algorithme NSGA-II règlent expérimentalement les paramètres de cet algorithme. Autrement dit, l'algorithme génétique est lancé plusieurs fois en utilisant différentes combinaisons de réglages et la combinaison qui donne les meilleurs résultats est retenue. Ces paramètres sont :

- la probabilité de croisement,
- la probabilité de mutation,
- la taille de la population,

– le nombre d'itérations.

L'objectif de cette phase de calibrage n'été pas de faire un plan d'expérience complet sur l'ensemble des paramètres qui agissent sur l'algorithme NSGA-II, mais seulement de traiter quelques interactions isolées entre quelques paramètres sur quelques instances de tailles moyennes.

Il est difficile de réaliser une analyse complète de toutes les interactions des paramètres, notamment à cause du temps de calcul que cela requiert.

Dans la suite nous présentons les expérimentations qui ont été effectuées sur une instance de 40 requêtes et 5 véhicules générés aléatoirement. La mise en oeuvre de cette phase de calibrage consiste à effectuer des alternances entre les deux scénarii suivants :

– On fixe la taille de la population N et le nombre d'itérations I. Ensuite, nous faisons varier la probabilité de croisement p_N^C et la probabilité de mutation p_N^M.
– On fixe la probabilité de croisement p_N^C et la probabilité de mutation p_N^M. Ensuite, nous faisons varier la taille de la population N et le nombre d'itérations I.

4.6.1 Les probabilités de croisement et de mutation

Pour le calibrage de ces paramètres, nous avons réalisé l'essai suivant : nous fixons la taille de la population à 200 et le nombre d'itérations à 1500 et nous faisons varier la probabilité de croisement $p_N^C \in \{50\%; 60\%; 70\%; 80\%; 90\%; 100\%\}$ et la probabilité de mutation $p_N^M \in \{5\%; 10\%; 15\%; 20\%\}$. Ensuite, on lance l'exécution de l'algorithme pour chaque paire de paramètres (p_N^C, p_N^M) sur l'instance de 40 requêtes et 5 véhicules. Pour chaque exécution, on récupère le front de Pareto généré.

Pour évaluer les résultats obtenus par chaque jeu de paramètres, nous utilisons les trois indicateurs de qualité Q_1, Q_2 et Q_3. Pour calculer ces indicateurs nous avons utilisé comme front de Pareto de référence ZE^*, le front obtenu en fusionnant l'ensemble des fronts obtenus avec les différents jeux de paramètres.

La Table 4.5 montre les résultats obtenus avec les différentes valeurs de la probabilité de croisement et de mutation. Chaque triplet représente les trois indicateurs de qualité (Q_1, Q_2, Q_3).

p_N^M \ P_N^C	50%	60%	70%	80%	90%	100%
0%	(0,00 ;0,00 ;0,32)	(0,00 ;0,00 ;0,31)	(0,00 ;0,00 ;0,35)	(0,00 ;0,00 ;0,31)	(2,94 ;7,06 ;0,38)	(0,00 ;0,00 ;0,27)
5%	(0,00 ;0,00 ;0,39)	(0,00 ;0,00 ;0,34)	(0,00 ;0,00 ;0,33)	(0,00 ;0,00 ;0,40)	(0,00 ;0,00 ;0,30)	(1,96 ;4,00 ;0,30)
10%	(0,00 ;0,00 ;0,34)	(8,33 ;15,18 ;0,39)	(0,00 ;0,00 ;0,34)	(0,98 ;1,67 ;0,46)	(2,94 ;6,00 ;0,31)	(2,94 ;4,88 ;0,34)
15%	(0,00 ;0,00 ;0,36)	(13,24 ;36,49 ;0,37)	(0,00 ;0,00 ;0,32)	(0,00 ;0,00 ;0,36)	(2,94 ;3,8 ;0,32)	**(29,41 ;49,59 ;0,30)**
20%	(0,00 ;0,00 ;0,25)	(0,00 ;0,00 ;0,34)	(0,49 ;0,84 ;0,33)	(8,33 ;20,00 ;0,35)	(0,00 ;0,00 ;0,41)	(2,45 ;4,85 ;0,38)
100%	(1,96 ;4,40 ;0,31)	(0,49 ;0,78 ;0,32)	(0,49 ;0,79 ;0,34)	(11,76 ;23,30 ;0,35)	(6,37 ;9,63 ;0,29)	(1,96 ;5,41 ;0,33)

TABLE 4.5 – Probabilities de croisement et de mutation

Selon les résultats obtenus dans la Table 4.5, de bons résultats sont obtenus avec une probabilité de croisement de 100% et une probabilité de mutation de 15%. Plus précisément, selon les indicateurs de qualité Q_1, Q_2 et Q_3, le front de Pareto généré par cette combinaison de paramètres est meilleur que les fronts de Pareto obtenus avec les autres combinaisons.

Les solutions obtenues avec la combinaison $(p_N^C = 100\%, p_N^M = 15\%)$, qu'on note A,

représentent $Q_1 = 29,41\%$ du front de référence ZE^*. L'indicateur Q_2 montre que presque 50% des solutions qui ont été générées par cette combinaison de paramètre sont non-dominées. L'indicateur Q_3 montre que le sous-ensemble A généré avec cette combinaison est plus proche de l'ensemble ZE^* que les fronts générés avec des autres combinaisons, sauf pour trois combinaisons ($p_N^C = 50\%, p_N^M = 20\%$), ($p_N^C = 90\%, p_N^M = 100\%$) et ($p_N^C = 100\%, p_N^M = 0\%$) pour lesquelles l'indicateur Q_3 est meilleur mais moins bon au niveau des indicateurs Q_1 et Q_2. Cet indicateur Q_3 est à utiliser avec précaution parce qu'il dépend du nombre de solutions qui composent le front de Pareto obtenu avec chaque combinaison des paramètres : les tailles de ces fronts peuvent être différentes d'une combinaison à l'autre.

4.6.2 La taille de la population et le nombre d'itérations

4.6.2.1 Choisir une taille de population en fixant le nombre d'itérations

Une fois les probabilités de croisement et de mutation fixées, le nombre d'itérations et la taille de la population doivent être calibrées. Le premier test effectué consiste à fixer le nombre d'itérations à 3000 et à faire varier la taille de la population entre 100, 200, 300, et 400. Le front de référence ZE^* est l'ensemble des solutions non-dominées obtenues à partir des fronts de Pareto générés par NSGA-II en utilisant les différentes tailles de population. Les résultats des différents indicateurs de qualité sont rapportés dans la Table 4.6.

La taille du problème	Q_1 (%)	Q_2 (%)	Q_3
100	1,45	4,00	1,00
200	1,09	1,49	0,75
300	17,45	16,11	0,81
400	80,00	88,35	0,60

TABLE 4.6 – Les résultats obtenus avec les différentes tailles de population

Selon les résultats de la Table 4.6, il est préférable de choisir une taille de population de 400 individus. Mais il faut noter que pour un nombre d'itérations donné, le temps d'exécution augmente avec l'augmentation de la taille de population. Donc le test avec une taille de population de 400 a pris beaucoup plus de temps que les autres tests.

4.6.2.2 Choix du nombre d'itérations en fixant le taille de la population à 400

Ce test consiste à fixer la taille de la population à 400 et à faire varier le nombre d'itérations. Nous avons choisi un nombre d'itérations assez grand, égale à 20000, et nous observons le comportement de NSGA-II à partir de l'itération 9000 et après chaque millier d'itérations. Précisément, à partir de l'itération 9000 nous récupérons le front de Pareto généré après chaque 1000 itérations écoulées (i.e pour l'ensemble des itérations : 9000, 10000, 11000,..., 20000). Les fronts de Pareto qui ont été récupérés sont évalués en utilisant les trois indicateurs de qualité Q_1, Q_2 et Q_3. Le front de référence utilisé pour calculer ces indicateurs est l'ensemble des solutions non-dominées qui ont été retenues le long de l'exécution de l'algorithme NSGA-II. La Table 4.7 montre l'évolution des trois indicateurs à chaque nombre d'itérations considéré.

#iterations	Q_1 (%)	Q_2 (%)	Q_3
9000	3,00	4,50	1,00
10000	8,08	11,84	0,62
11000	9,50	14,28	0,45
12000	13,67	20,56	0,27
13000	18,17	27,18	0,20
14000	24,17	36,34	0,15
15000	30,00	44,89	0,11
16000	41,00	61,50	0,11
17000	51,17	76,55	0,08
18000	55,67	83,29	0,08
19000	56,16	83,83	0,07
20000	62,33	93,50	0,01

TABLE 4.7 – Convergence de NSGA-II : le nombre d'itérations

La Table 4.7 montre qu'à l'itération 14000, NSGA-II a déjà généré un quart de solutions ($Q_1 = 24,17\%$) de l'ensemble de référence. Les solutions générées à ce nombre d'itérations sont proches du front de Pareto de référence selon l'indicateur de qualité $Q_3 = 0.15$. Cependant, à cause de la méthode d'échantillonnage (la distance de Crowding) utilisée par NSGA-II, plus de 38% (100% - 62,33%) des solutions générées entre les itérations 9000 et 19000 ne figurent pas dans le front de Pareto généré à l'itération 20000, même si ces solutions ne sont dominées par aucune solution du front de référence. Cela montre un comportement paradoxal au principe d'élitisme sur lequel se base l'algorithme NSGA-II.

Au fil des générations des solutions, on remarque qu'il y un cycle répétitif d'apparition puis de disparition des solutions. Ceci est dû essentiellement au mécanisme de maintien de la diversité mis en oeuvre par NSGA-II en utilisant la distance de Crowding et l'utilisation d'une taille de population limitée.

Tant que le nombre de solutions du premier front n'excède pas la taille de la population, NSGA-II est capable de préserver toutes les solutions de ce front. Dans le cas contraire (le nombre de solutions non-dominées dépasse la taille de la population), certaines solutions intéressantes doivent être supprimées : le choix de ces solutions à éliminer se fait simplement sur la base de la distance de Crowding afin de maintenir les solutions les plus diversifiées. Rien ne garantit que les solutions éliminées ne soient pas des optimas de Pareto et qu'elles aient laissé leur place au profit des autres solutions qui ne sont pas des optimas de Pareto. Pour cela une analyse approfondie du comportement de l'algorithme NSGA-II doit être effectuée pour ne pas laisser l'algorithme entrer dans un cycle répétitif d'apparition puis de disparition des mêmes solutions. Ce qui peut rendre la recherche des solutions très longue en terme de temps de calcul.

Selon les résultats de la Table 4.7 et si nous devons faire un arbitrage entre le temps d'exécution et la qualité des solutions obtenues, nous préférons fixer le nombre d'itérations à 14000 plutôt que de laisser NSGA-II tomber dans un cycle répétitif de productions et de suppressions de solutions. Ceci est clairement visible en regardant l'indice de qualité Q_3

qui montre que NSGA-II commence à générer des fronts de Pareto qui sont relativement plus proche du front de référence à partir de l'itération 14000.

4.6.2.3 Choix d'une taille de population en fixant le temps d'exécution

Une fois le nombre d'itérations fixé, un autre test est effectué afin de choisir une taille de population adapté pour l'algorithme NSGA-II et qui sera utilisée pour résoudre notre problème de DARP tricritère. Ce test consiste à prendre le temps d'exécution de NSGA-II correspondant à 14000 itérations et une population de 400 individus et faire varier la taille de la population entre 100, 200, 300 et 400. Pour chaque taille de population, NSGA-II est lancé avec le même temps d'exécution. La Table 4.8 donne les résultats des indicateurs de qualité obtenus pour chaque taille de population. Rappelons que l'ensemble de référence ZE^* est construit à partir des fronts de Pareto générés en utilisant les différentes tailles de population.

Taille de population	Q_1 (%)	Q_2 (%)	Q_3
100	3.58	9.90	1.00
200	58.06	80.60	0.008
300	9.67	9.00	0.24
400	28.67	20.10	0.64

TABLE 4.8 – Convergence de NSGA-II : la taille de population

Selon les résultats de la Table 4.8, on remarque qu'avec une taille de population de 200, NSGA-II génère plus de $Q_1 = 58\%$ de solutions de l'ensemble de référence. Ces solutions représentent $Q_2 = 80\%$ de l'ensemble de solutions générées avec cette taille de population.

Le front de Pareto qui a été construit avec ces solutions est proche de l'ensemble de référence ZE^* par rapport aux fronts générés en utilisant d'autres tailles de population. Donc pour un même temps d'exécution, il est préférable d'utiliser une taille de population de 200 individus.

4.6.2.4 Choix du nombre d'itérations en fixant le taille de la population à 200

Ce test consiste à fixer la taille de population à 200 individus et faire varier le nombre d'itérations. Nous avons choisi un nombre d'itérations assez grand, égale à 20000 et nous observons le comportement de NSGA-II à partir de l'itération 9000 et après chaque millier d'itérations comme dans la Section 4.6.2.2. La Table 4.9 présente l'évolution des trois indicateurs à chaque nombre d'itérations considéré.

Comme c'est le cas dans la Section 4.6.2.2, nous observons le même comportement de l'apparition et de disparition de solutions non-dominées par l'ensemble de référence. Plus de 42% (100% - 57,34%) des solutions générées entre les itérations 9000 et 19000 ne figurent pas dans le front de Pareto généré à l'itération 20000, même si ces solutions ne sont dominées par aucune solution du front de référence.

La Table 4.9 montre qu'à l'itération 14000, NSGA-II a déjà généré plus d'un tiers de solutions ($Q_1 = 34,11\%$) de l'ensemble de référence. Les solutions générées à ce nombre

#iterations	Q_1 (%)	Q_2 (%)	Q_3
9000	1,03	5,20	1,00
10000	5,03	15,42	0,77
11000	9,05	17,66	0,52
12000	14,16	20,56	0,37
13000	21,10	43,28	0,31
14000	34,11	51,43	0,19
15000	37,05	55,96	0,17
16000	42,38	59,54	0,14
17000	47,36	70,11	0,10
18000	51,12	85,91	0,08
19000	53,23	89,13	0,05
20000	57,34	94,12	0,02

TABLE 4.9 – Convergence de NSGA-II : le nombre d'itérations

d'itérations sont proches du front de Pareto de référence selon l'indicateur de qualité $Q_3 = 0,19$. Pour cela, si nous devons faire un arbitrage entre le temps d'exécution et la qualité des solutions obtenues, nous préférons fixer le nombre d'itérations à 14000 plutôt que de laisser NSGA-II tomber dans un cycle répétitif de productions et de suppressions de solutions.

Pour résumer, les expérimentations effectuées précédemment nous ont permis de fixer la taille de la population à 200 et le nombre d'itérations à 14000.

4.6.3 Les résultats obtenus sur des instances aléatoires

Taille de population	Nombre d'itérations	Probabilité de croisement	Probabilité de mutation
200	14000	100	15

TABLE 4.10 – Les paramètres de NSGA-II retenus

En utilisant les valeurs de ces paramètres, nous exécutons NSGA-II sur les trente catégories d'instances qui ont été générées aléatoirement. Des résultats obtenus sont rapportés dans la Table 4.11 qui contient des informations sur le nombre de solutions non-dominées générées par NSGA-II et le temps moyen d'exécution (en secondes) de chaque groupe d'instances.

Les colonnes de la Table 4.11 ont les significations suivantes :

- n : le nombre de requêtes,
- m : le nombre de véhicules,
- $\#Front_{min}$: la taille minimale du front de Pareto pour chaque catégorie d'instances,
- $\#Front_{avg}$: la taille moyenne du front de Pareto pour chaque catégorie d'instances,
- $\#Front_{max}$: la taille maximale du front de Pareto pour chaque catégorie d'instances,
- $\#CPU_{min}$: le temps d'exécution minimale pour chaque catégorie d'instances,
- $\#CPU_{avg}$: le temps d'exécution moyen pour chaque catégorie d'instances,

Taille du problème		Taille du front de Pareto			Temps CPU (secondes)		
n	M	$\#Front_{min}$	$\#Front_{avg}$	$\#Front_{max}$	CPU_{min} (s)	CPU_{avg} (s)	CPU_{max} (s)
10	2	207	216,60	222	511,50	523,72	532,70
10	4	206	213,90	222	530,81	554,40	584,48
10	6	205	209,10	216	565,98	586,25	626,73
20	2	8	185,40	208	374,39	658,87	735,23
20	4	196	207,20	213	763,50	796,98	828,17
20	6	201	205,00	211	799,71	855,22	889,75
30	2	199	202,10	205	398,26	842,29	962,54
30	4	200	204,80	209	1028,44	1049,91	1081,98
30	6	202	205,20	210	1062,28	1101,82	1134,09
40	2	201	203,80	208	1165,80	1201,31	1258,22
40	4	201	203,50	205	1235,38	1283,18	1313,72
40	6	201	203,20	208	1317,36	1336,18	1359,44
50	2	199	202,30	205	1436,38	1471,57	1504,23
50	4	200	202,50	207	1462,34	1496,35	1516,75
50	6	199	201,60	203	1524,27	1566,77	1618,27
60	2	197	201,00	204	1692,78	1724,58	1750,91
60	4	200	201,80	204	1696,84	1732,56	1768,41
60	6	199	202,40	206	1717,64	1761,30	1811,45
70	2	198	201,10	204	1861,98	1922,19	1975,16
70	4	199	201,40	204	1705,84	1838,67	1929,02
70	6	197	201,70	205	1853,5	1886,57	1919,14
80	2	198	201,60	203	2146,77	2231,09	2338,99
80	4	200	203,00	209	1891,88	2038,22	2276,97
80	6	200	201,70	205	1925,36	2092,31	2188,97
90	6	198	201,40	206	2087,69	2211,77	2425,20
100	6	200	201,70	205	2335,38	2511,82	2771,11
150	8	197	201,60	206	2812,98	3006,51	3128,76
200	8	157	195,60	202	3872,16	4007,59	4103,02
200	10	184	198,60	202	3451,34	3687,44	3970,70
200	12	190	198,60	202	3510,06	3907,02	6142,30

TABLE 4.11 – Les résultats des instances aléatoires

– $\#CPU_{max}$: le temps d'exécution maximale pour chaque catégorie d'instances.

Selon les résultats de la Table 4.11, on remarque que dans la majorité des catégories des instances la taille des moyenne des fronts de Pareto dépasse la taille de la population initiale fixée à 200 dans l'algorithme NSGA-II. Au niveau du temps d'exécution, on remarque que le temps moyenne d'exécution des instances avec un nombre de requêtes 200 dépasse une heure de calcul.

4.6.4 Comparaison avec les résultats de [Cordeau et Laporte, 2003]

Du fait du manque d'instances de référence adaptées à notre problème de DARP, les instances utilisées pour les tests sont inspirées de celles proposées par [Cordeau et Laporte, 2003]. L'objectif des tests complémentaires présentés dans cette section est d'essayer d'évaluer l'algorithme NSGA-II sur un problème proche pour lequel il existe des algorithmes de résolution. Ces instances ont été utilisées par Cordeau et Laporte pour évaluer la performance de leur algorithme de recherche Tabou qu'ils ont proposé pour résoudre un problème de DARP statique monocritère. Nous allons ainsi pouvoir évaluer la performance de NSGA-II à générer des bonnes solutions au moins sur l'aspect du coût de transport.

4.6.4.1 Les instances

Les instances proposées par [Cordeau et Laporte, 2003] contiennent entre 24 et 144 requêtes générées aléatoirement dans le carré $[-10, 10]^2$. La flotte de véhicules est homogène et leur capacité est égale à 6, le nombre de véhicules varie entre 3 et 13. Chaque requête possède une fenêtre de temps associée soit à son point de départ s'il s'agit d'une requête de *Retour*, soit à son point de destination s'il s'agit d'une requête d'*Aller*. Chaque requête ne concerne qu'une seule personne à servir et un temps de service de 10 est associé à chaque point de départ et d'arrivée. Une durée maximale du trajet est fixée à 90 et une durée maximale de chaque tournée est fixée à 480. Les auteurs ont proposés 20 instances divisées en deux groupes : $(R1a, R2a, ..., R10a)$ et $(R1b, R2b, ..., R10b)$. Dans le premier groupe, les fenêtres de temps sont plus serrées que pour les instances du deuxième groupe.

La distance entre deux sommets est définie comme étant la distance euclidienne entre leur coordonnées (x, y) et la durée du trajet entre ces deux sommets est égale à la distance entre ces deux sommets.

4.6.4.2 Adaptation des instances

Pour que nous puissions effectuer des testes expérimentaux sur les instances de Cordeau et Laporte [Cordeau et Laporte, 2003] nous avons effectué des adaptations de la façon suivante : pour tout sommet qui n'est pas associé à une fenêtre de temps, un intervalle de temps artificiel est généré. Pour chaque requête r_i nous avons deux cas : le premier cas, quand seulement le sommet de destination v_{i+n} qu'est associé à une fenêtre de temps, alors la fenêtre de temps associé au sommet de départ est égale à $e_i = max\{0, l_{i+n} - L\}$ et $l_i = min\{H, l_{i+n} - t_{i,i+n}\}$ avec H est l'horizon de planification et L le temps maximal que peut passer les passagers à bord d'un véhicule.

Dans le cas où seule le point de départ est associé à une fenêtre de temps, alors la fenêtre de temps du point de destination est définie par : $e_{i+n} = min\{H, e_i + t_{i;i+n}\}$ et $l_{i+n} = min\{H, e_i + L\}$.

La contrainte du temps maximal passé à bord du véhicule (Ride time) utilisée dans le problème de DARP étudié par [Cordeau et Laporte, 2003] est prise en compte dans notre problème lors de l'adaptation des fenêtres de temps. Des grandes valeurs sont attribuées aux paramètres α_i et β_i afin de pénaliser la violation de ces fenêtres de temps.

La structure de notre problème même après l'adaptation des instances ne reflètent pas le problème entier étudié dans ([Cordeau et Laporte, 2003]). Dans la suite nous présentons quelques différences principales qui existent entre ces deux problèmes.

Dans le problème étudié dans ([Cordeau et Laporte, 2003]), un sommet v_i d'embarquement ou de débarquement associé à une fenêtre de temps $[e_i, l_i]$ ne peut pas être servi en dehors de sa fenêtre de temps et si le véhicule arrive en avance alors il doit attendre avant de servir ce sommet. Tandis que dans notre problème le véhicule attend seulement s'il arrive avant la borne inférieure de la fenêtre de temps d'un sommet de départ. Un sommet de destination peut être servi en avance et une pénalité d'avance est calculée lors de l'évaluation de la solution.

Une autre différence est due aux fenêtres de temps artificielles qui ont été calculées

afin de respecter la contrainte du temps maximal passé à bord du véhicule. La prise en compte de la contrainte du temps maximal passé à bord en utilisant ces intervalles de temps rend le problème plus contraignant. Par exemple, dans le cas d'une requête de *Retour* r_i (c'est-à-dire dans le cas où seulement la fenêtre de temps de départ $[e_i, l_i]$ est définie dans le problème étudié par Cordeau et Laporte), une solution dans laquelle l'heure d'arrivée au sommet de destination est entre $e_i + L$ et $l_i + L$ peut être considérée faisable dans le problème étudié dans ([Cordeau et Laporte, 2003]).

Cependant, cette solution est considérée comme non réalisable dans notre problème et elle est pénalisée lors du calcul du critère de la qualité de service.

Il est important de mentionner que le critère de la durée maximale d'une tournée n'a pas été considéré dans notre problème. Donc la contrainte n'est pas contrôlée durant la recherche des solutions. Cependant, pour faire la comparaison, seules les solutions qui respectent cette contrainte sont sélectionnées parmi l'ensemble des solutions générées par NSGA-II.

Par conséquent les deux problèmes que nous voulons comparer n'ont pas exactement les mêmes caractéristiques, donc la comparaison des résultats obtenus avec les deux méthodes est une tâche difficile, mais il faut garder à l'esprit que l'objectif n'est pas de comparer la performance des deux méthodes mais seulement de voir la qualité des solutions générées par NSGA-II au niveau du coût de transport.

Pour avoir une idée sur la qualité des solutions générées par NSGA-II, nous avons comparé la meilleure solution obtenue avec NSGA-II en terme de coût de transport avec la meilleure solution obtenue par la recherche Tabou développée dans ([Cordeau et Laporte, 2003]). Rappelons que la fonction objectif dans le problème de DARP mono-objectif traité par Cordeau et Laporte consiste à minimiser le coût total de transport, qui correspond à la minimisation de la distance totale (ils considèrent que le coût de transport par kilomètre est égal à 1, i.e. $b_k = 1$).

Pour effectuer cette comparaison, nous sélectionnons, à partir de l'ensemble des optima de Pareto approchés générés par NSGA-II, la solution avec le meilleur coût de transport et qui respecte en plus les contraintes que nous avons considéré dans notre problème : les contraintes de temps maximal passé à bord du véhicule et la contrainte de la durée maximale des tournées. La solution est comparée avec celle obtenue avec l'algorithme de recherche Tabou.

Le Tableau 4.12 montre les résultats obtenus par la méthode NSGA-II. Seuls les résultats de 13 instances ont été rapportés dans cette table, car dans ([Cordeau et Laporte, 2003]) les auteurs ne fournissent pas toutes les informations sur les solutions trouvées pour les 7 autres instances restantes. Dans ce tableau, les colonnes ont la signification suivante :

- Instance : le nom de l'instance,
- Cost : le coût de transport,
- Route duration : la durée totale des tournées,
- Veh.wait.time : la durée totale d'attente des véhicules,
- Ride time : le temps total passé à bord des véhicules par l'ensemble des passagers,
- CPU (min) : le temps d'exécution en minutes. Ce temps d'exécution est multiplié par 2,5 pour obtenir le temps d'exécution équivalent au temps qu'on peut obtenir sur un PC Pentium 4 et 2 GHz utilisé dans ([Cordeau et Laporte, 2003]). Pour comparer

la performance des PC nous nous sommes basés sur les résultats fournis par *The Standard Performance Evaluation Corporation* (SPEC), disponible sur le site web : http ://www.spec.org/cpu2000/results/res2007q1.

Les résultats obtenus montrent que notre méthode peut produire des solutions qui sont meilleures en terme de coût de transport, notamment pour les petites instances. Cependant, pour les instances de grande taille, NSGA-II n'a pas pu produire de solutions de meilleure qualité sans générer des solutions qui ne respectent pas la contrainte de la durée maximale d'une tournée ou la contrainte de temps maximal passé à bord du véhicule.

Au niveau du coût de transport moyen et de la durée totale des tournées, les résultats obtenus dans ([Cordeau et Laporte, 2003]) sont meilleurs par rapport à ceux obtenus avec NSGA-II. Cependant, au niveau du temps passé à bord des véhicules par l'ensemble des passagers, NSGA-II est nettement meilleur que la recherche Tabou. Plus précisément avec notre méthode le temps moyen passé à bord est de 2690,30 alors qu'avec la recherche Tabou il est de 4376,92.

Les mêmes instances de Cordeau et Laporte ont été résolues par des algorithmes proposés dans ([Jorgensen *et al.*, 2007]) et dans ([Parragh *et al.*, 2010]). Les Tables 4.14 et 4.15 montrent les résultats obtenus.

Instance	Cost	Route duration	Veh, wait,time	Ride time	CPU (min)
R1a	182,10	967	305	857	7,21
R2a	298,60	1707	449	2094	30,19
R3a	595,79	2622	586	2493	64,90
R5a	801,60	3795	593	3876	172,88
R9a	941,96	3278	176	3455	187,62
$R10a^*$	1228,77	4575	466	4673	362,01
R1b	167,66	1068	420	685	8,16
R2b	343,20	1670	367	1242	32,91
R5b	866,91	4425	1158	3064	212,43
R6b	997,63	4815	937	4225	293,00
R7b	275,68	1244	249	1311	16,82
$R9b^*$	855,62	3420	405	2903	201,44
$R10b^*$	1203,58	4367	283	4096	366,40
Moyenne	673,06	2919,45	491,86	2690,30	150,26

* la contrainte de la durée maximale d'une tournée est violée

TABLE 4.12 – Résultats obtenus par la méthode NSGA-II

Selon les Tableaux 4.14 et 4.12, les résultats obtenus avec NSGA-II sont nettement meilleurs par rapport à ceux obtenus dans ([Jorgensen *et al.*, 2007]), notamment au niveau du coût de transport moyen et de la durée totale des tournées. Cependant, au niveau du temps d'attente et de temps passé à bord des véhicules, l'algorithme génétique proposé dans ([Jorgensen *et al.*, 2007]) produit des solutions de meilleur qualité.

Les résultats obtenus par l'algorithme NSGA-II sont meilleurs, au niveau du coût de transport, par rapport à ceux obtenus par Parragh et al. (voir Table 4.15). Mais au niveau

Instance	Cost	Route duration	Vehi,wait,time	Ride time	CPU (min)
R1a	190,02	881	211	1095	19
R2a	302,08	1985	724	1977	80,60
R3a	532,08	2579	607	3587	171,80
R5a	636,97	3870	833	6154	462,40
R9a	672,44	3155	323	5622	505,10
R10a	878,76	4480	721	7164	875,30
R1b	164,46	965	321	1042	19,30
R2b	296,06	1565	309	2393	82,90
R5b	589,74	3596	606	6105	543,30
R6b	743,60	4072	449	7347	737,00
R7b	248,21	1097	129	1762	42,30
R9b	601,96	3249	487	5581	512,80
R10b	798,63	4041	362	7072	924,10
Moyenne	511,92	2733,61	467,84	4376,92	382,76

TABLE 4.13 – Résultats obtenus par la méthode de recherche Tabou ([Cordeau et Laporte, 2003])

Instance	Cost	Route duration	Vehi,wait,time	Ride time	CPU (min)
R1a	309,00	1041	252	477	5,57
R2a	539	1969	470	1367	11,43
R3a	1047	2779	292	3081	21,58
R5a	1350	4250	500	5099	58,23
R9a	1343	3597	94	6251	40,78
R10a	1811	5006	315	8413	65,98
R1b	284	907	143	630	5,45
R2b	561	1719	198	1214	11,72
R5b	1344	4296	552	4615	58,93
R6b	1799	5309	630	6134	81,23
R7b	478	1299	102	990	8,29
R9b	1372	3679	147	5362	44,66
R10b	1740	4733	113	7969	66,41
Moyenne	1075,15	3121,85	292,92	3669,38	36,94

TABLE 4.14 – Résultats obtenus par l'algorithme génétique ([Jorgensen *et al.*, 2007])

de la durée des tournées, du temps d'attente des véhicules et du temps passé à bord l'algorithme de recherche à voisinage variable proposé dans ([Parragh *et al.*, 2010]) produit des solutions nettement meilleures. Les résultats obtenus par l'algorithme NSGA-II sont prometteurs, l'avantage de cet algorithme est qu'il permet de fournir aux Décideurs un ensemble d'optima de Pareto et pas seulement une solution unique, comme c'est le cas pour les méthodes proposées dans ([Cordeau et Laporte, 2003]), ([Parragh *et al.*, 2010]) et ([Jorgensen *et al.*, 2007]).

Instance	Cost	Route duration	Vehi,wait,time	Ride time	CPU (min)
R1a	273,70	863,65	109,95	480	2,70
R2a	431,15	1774,19	383,04	960,00	5,16
R3a	777,92	2425,92	208,00	1440	6,38
R5a	1023,77	3712,18	288,42	1399,99	13,93
R9a	1045,34	3232,18	26,84	2160,00	33,53
R10a	1389,89	4499,45	229,72	2879,84	40,27
R1b	238,51	737,27	18,76	480,00	3,78
R2b	431,18	1403,74	12,56	960	8,29
R5b	951,74	3417,62	65,88	2400,00	23,19
R6b	1241,22	4182,99	61,77	2880,00	26,39
R7b	359,19	1123,56	44,37	720	4,49
R9b	977,59	3211,71	74,12	2160,00	30,32
R10b	1310,00	4218,37	28,38	2879,99	51,81
Moyenne	803,94	2677,14	119,37	1753,83	19,25

TABLE 4.15 – Résultats obtenus par la méthode de recherche à voisinage variable ([Parragh *et al.*, 2010])

4.7 Conclusion

Les résultats obtenus précédemment montrent que l'algorithme NSGA-II semble faire converger l'ensemble des solutions non-dominées de la population initiale. Puisque le nombre d'optima de Pareto augmente d'une manière "exponentielle" avec la taille du problème, une petite taille de population ne peut pas garder l'ensemble des solutions non-dominées qui peuvent être générées le long de l'exécution. Pour cela une méthode d'échantillonnage est utilisée dans NSGA-II pour choisir à partir d'un ensemble de solutions non-dominées un sous ensemble de solutions bien diversifiées.

Mais la méthode de la distance de *Crowding* utilisée dans NSGA-II rend difficile l'obtention d'une convergence rapide vers un ensemble de solutions diversifiées et de meilleurs compromis. Le long du processus de génération des solutions, il arrive que quelques solutions non-dominées ne soient pas archivées faute d'une taille de population suffisante. Ce comportement de l'algorithme de NSGA-II rend difficile le contrôle de la convergence de l'algorithme et empêche de garantir une stabilité des solutions générées d'une exécution à l'autre. Néanmoins, ce n'est pas tant la structure de l'algorithme génétique qui est en cause, mais plutôt la structure du problème et le nombre très important d'optima de Pareto existants.

On peut observer que l'avantage de NSGA-II, par rapport à d'autres métaheuristiques multiobjectifs permettant l'énumération des optimas de Pareto, et de pouvoir être utilisé comme une heuristique permettant d'obtenir un *sous-ensemble représentatif*. Ainsi, on pourrait même imaginer qu'un Décideur ne voulant voir qu'un nombre limité de solutions (par exemple 7) initialise la taille de la population à ce nombre.

Dans cette partie de la thèse nous avons proposé un algorithme NSGA-II pour résoudre un problème de DARP statique tricritère avec une flotte de véhicules hétérogène. La parti-

cularité de ce problème est l'intégration des trois aspects du système de transport : son coté financier, sa perception de point de vu des usagers et son impact environnemental. Ceci nous a permis d'introduire trois critères pour évaluer le DARP : la minimisation du coût de transport, la maximisation de la qualité de service rendue aux passagers et la minimisation de la quantité de CO_2 émise par l'ensemble des véhicules.

Pour résoudre ce problème nous avons proposé un algorithme NSGA-II pour trouver une approximation de l'ensemble des optima de Pareto stricts. Les résultats préliminaires obtenus sont prometteurs pour notre problème de DARP tricritère. Quelques heuristiques pour initialiser la population ont été développées afin d'accélérer la convergence de NSGA-II. Un algorithme d'optimal Timing de complexité $O(nlogn)$ est utilisé pour calculer les dates de début de service à chaque sommet dans une tournée donnée afin de minimiser l'insatisfaction des passagers (maximiser la qualité de service rendue à l'ensemble des passagers).

L'adaptation de l'algorithme NSGA-II pour résoudre le problème de DARP statique tricritère a été présenté dans [Atahran *et al.*, 2011a] et [Atahran *et al.*, 2011c] ainsi que dans ([Atahran *et al.*, 2012]).

Chapitre 5

Le problème de Transport à la Demande dynamique

5.1 Introduction

Les grandes avancées technologiques au cours des dernières années permettent la gestion de systèmes de transport dynamiques : ceux-ci deviennent plus faciles à concevoir parce que la majorité des nouveaux véhicules sont équipés de matériels embarqués GPS (*Global Position System*) et de Systèmes d'Information Géographiques (SIG). Ainsi, les sociétés de distribution sont désormais capables de surveiller la position des véhicules et leur statut à tout moment. Dans ce chapitre nous nous intéressons à la résolution de la version dynamique du Transport à la Demande, qui permet aux passagers de demander le service de transport quasiment en temps réel. Pour cela le système mis en place pour la gestion du service doit être capable de prendre une décision d'accepter ou de rejeter une ou plusieurs nouvelles requêtes lorsqu'elles se présentent. Cela revient à résoudre un nouveau problème de transport qui prend en considération les nouvelles requêtes qui arrivent au fil du temps. Souvent, ce problème est résolu en deux phases : dans la première phase il s'agit seulement de chercher une solution faisable en utilisant des méthodes heuristiques rapides, tandis que la deuxième phase (si une solution faisable est trouvée dans la première phase) consiste à améliorer la solution réalisable trouvée en s'autorisant un temps de calcul plus important. La première phase survient lorsque l'opérateur de transport doit fournir une réponse rapidement à un usager (par exemple au téléphone) tandis que la seconde phase se déroule en dehors de tout contact avec l'usager.

Dans un problème dynamique le nombre de véhicules peut évoluer durant le temps. Pour cela la mise en place d'un véhicule de plus sur la route est une décision à prendre en considération lors de la planification des tournées.

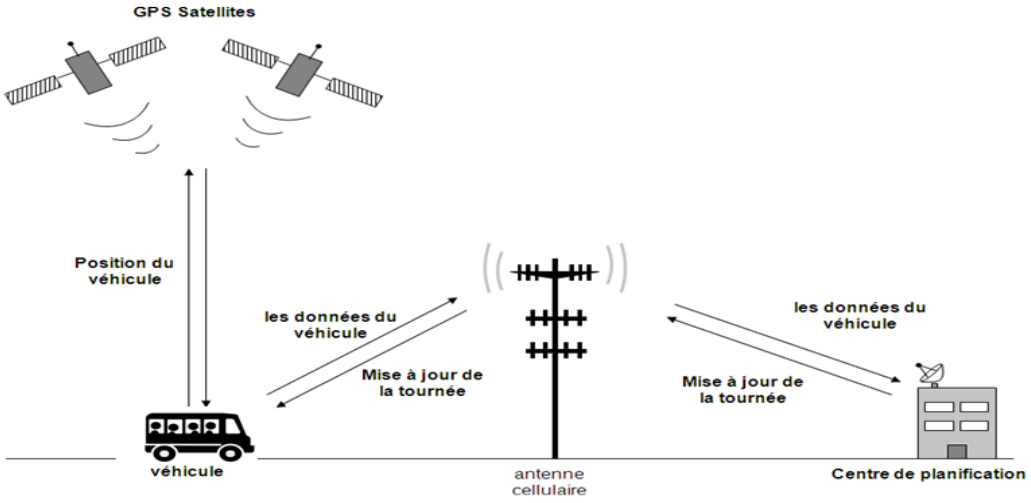


FIGURE 5.1 – Scénario de planification d'une tournée en temps réel

5.2 Définition et modélisation du problème de Transport à la Demande dynamique

Dans le problème du Transport à la Demande dynamique on permet à des clients d'appeler dans la journée pour demander une prise en charge le jour même. Les requêtes entrantes sont alors (si possible) ajoutées aux tournées déjà programmées avec les réservations précédentes. Il est possible qu'à l'heure de l'appel, les véhicules soient déjà en service, ce qui implique que la planification de la nouvelle requête peut se faire de différentes manières :

- la requête est insérée directement dans une tournée existante entre deux arrêts existants. Se posent alors plusieurs questions : Quelle tournée choisir ? Quelle position choisir ? Selon quel critère ?
- une tournée est réorganisée pour accueillir la nouvelle requête, il faut alors choisir un critère pour choisir la tournée,
- mettre en service un nouveau véhicule (qui est au dépôt) pour prendre en charge cette requête, mais cette décision est prise sur quelle base ? A quel moment peut-on décider d'introduire un nouveau véhicule dans le système ?
- les tournées sont toutes réorganisées pour insérer la requête.

Une phase de mise à jour des données est nécessaire avant de procéder à la résolution du problème. Pour les requêtes déjà enregistrées (statiques), nous nous sommes engagés sur une fenêtre de temps de prise en charge et une fenêtre de temps de dépose : elles doivent donc être respectées.

Il faut noter que les fenêtres de temps d'origine (souhaitées par les passagers) peuvent être modifiées une fois qu'elles sont planifiées en tenant compte de la valeur $\overline{QoS}$ obtenue. Ces fenêtres d'origine sont ajustées au plus une seule fois durant le processus de planifications des requêtes. Afin de simplifier la description du problème, on appelle la fenêtre de temps associée à une requête déjà planifiée "*une fenêtre ajustée*" même si elle n'a pas été

modifiée.

Soit t l'heure de soumission d'une nouvelle requête. On peut diviser les sommets qui composent notre problème en quatre catégories :

- le sommet correspond à des requêtes déjà effectuées (heure de départ et d'arrivée $< t$) : le sommet n'est plus pris en compte dans la suite du problème,
- le sommet correspond à une requête en cours (heure de départ $< t$ et heure d'arrivée $> t$) : le sommet est pré-affecté à un véhicule et on ne peut pas modifier la fenêtre de temps de passage ajustée qui lui est associée.
- le sommet correspond à une requête programmée mais non démarrée (heure de départ et heure d'arrivée $> t$) : le sommet n'est pas affecté à un véhicule mais la fenêtre de temps de passage ajustée ne peut pas être modifiée,
- le sommet correspond à une nouvelle requête : l'affectation à un véhicule est libre et la fenêtre de temps souhaitée par les passagers peut éventuellement être modifiée.

5.2.1 Les données du problème

Les données du problème fournies en entrée ne contiennent pas seulement les données liées aux requêtes et aux véhicules, mais aussi les données liées à la solution courante du problème. On distingue deux types de requêtes : le premier type concerne les requêtes statiques qui ont été acceptées et validées et pour lesquelles une solution a déjà été trouvée lors de la résolution du problème statique. Ces requêtes ont des horaires de service fermes, c'est-à-dire que les fenêtres de date de départ et d'arrivée doivent être respectées. Plus précisément, après la planification des requêtes statiques nous obtenons un ordonnancement pour l'ensemble des sommets de départ et d'arrivée relatifs à ces requêtes. Si, dans cet ordonnancement, l'heure de service A_i d'un sommet v_i donné dépasse l_i alors on redéfinit l_i par $l_i = A_i$, tandis que si $A_i < e_i$ alors on redéfinit $e_i = A_i$.

Le second type de requêtes concernent les requêtes dynamiques qui arrivent au fil du temps, pour lesquelles les fenêtres de temps sont souples, i.e. peuvent ne pas être respectées au détriment de la qualité de service rendue aux passagers. Pour les données liées aux véhicules nous aurons besoin, en plus des données liées à la capacité, au coût de transport et à l'émission de CO_2 et de connaitre à chaque instant t la position de chaque véhicule.

Il faut noter que dans le problème dynamique, chaque nouvelle requête est notée par r_i et correspond à un sommet de départ v_{2i-1} et un sommet d'arrivée v_{2i} avec $i \in \{1, 2, ..., n\}$ où n est le nombre de requêtes à traiter à l'instant considéré et v_0 le sommet correspondant au dépôt.

5.2.2 Les contraintes

La version dynamique du problème de transport à la demande que nous avons considéré dans cette étude contient plus de contraintes que le problème statique, car nous considérons qu'une fois les requêtes statiques planifiées, les plages horaires de desserte sont confirmées aux passagers et devront impérativement être respectées. Donc il est interdit de violer les fenêtres de temps liées aux requêtes statiques. De plus, nous avons des requêtes qui sont pré-affectées à certains véhicules, dans le cas des requêtes dont les passagers sont à bord

des véhicules lors de la soumission d'une nouvelle requête. Une autre contrainte est que les véhicules ne sont pas tous au dépôt quand on procède à l'insertion de nouvelles requêtes.

5.2.3 Les critères

Nous considérons les mêmes critères que ceux du problème statique :
- minimisation du coût total de transport,
- minimisation de l'insatisfaction des clients,
- minimisation de la quantité de CO_2 émise par les véhicules.

La première approche de résolution a été consacrée à la spécification et au développement d'une heuristique capable de fournir une solution rapide et de meilleure qualité possible.

5.3 Heuristique par insertion HI

5.3.1 Description générale

Cette méthode consiste à insérer rapidement un ensemble de requêtes dynamiques dans une solution statique initiale. Afin de simplifier la recherche des solutions ainsi que de raccourcir le temps de calcul, la structure des tournées est conservée. C'est-à-dire que l'affectation et l'ordonnancement des requêtes statiques sont maintenues et que les requêtes dynamiques sont insérées dans les tournées existantes. La sélection des meilleures solutions s'effectue selon les trois fonctions objectif. L'insertion doit respecter les contraintes de fenêtres de temps des passagers déjà planifiés et respecter bien évidemment la capacité des véhicules.

La première question qui se pose est l'ordre de priorité d'insertion des requêtes. La deuxième question est l'ordre des tournées, c'est à dire par quelle tournée commencer l'insertion des nouvelles requêtes ? En effet, il parait logique que plus une tournée est chargée, plus il est difficile d'insérer une nouvelle requête. Partant de ce constat, il est alors clair que l'ordre d'insertion des nouvelles requêtes peut avoir une importante influence sur la qualité de la solution finale. En outre, il faut remarquer que toutes les requêtes entrantes n'ont pas les mêmes caractéristiques. Certaines concernent beaucoup de passagers, d'autres ont des fenêtres de temps très étroites, etc. Pour cela nous avons défini un indicateur appelé « indice de difficulté » pour pouvoir comparer les requêtes et les trier : cet indice représente les caractéristiques intrinsèques de la requête et il est indépendant des tournées existantes. Nous partons du principe que le but est de réussir à insérer toutes les requêtes. Les requêtes seront donc insérées de la plus difficile à la plus facile, c'est-à-dire de l'indice de difficulté le plus grand au plus petit.

Afin de diminuer encore le temps de calcul, il peut être utile de tenter de deviner dans quelle tournée une requête peut facilement être insérée. Pour cela, nous avons introduit un nouveau coefficient que l'on nomme « degré d'attraction ». Ce coefficient va nous permettre de ranger les tournées selon qu'il semble facile ou difficile d'insérer la requête dans cette tournée.

5.3.2 Indice de difficulté d'une requête I_{Dif}

Cet indice est calculé pour chaque requête à partir de trois composantes :

1. La largeur des fenêtres de temps de la requête,
2. La marge maximale de parcours,
3. Le nombre de clients pour cette requête.

La largeur des fenêtres de temps

Cette composante permet de prendre en considération la largeur des fenêtres de temps au départ et à l'arrivée de chaque requête. Plus les fenêtres de temps sont étroites et plus la chance d'insérer la requête est faible. Pour une requête r_i de point de départ v_j et de point d'arrivée v_l cette composante est représentée par la formule suivante :

$$C_{fen} = \frac{1}{\Delta_j} + \frac{1}{\Delta_l}$$

avec :
- $\Delta_1 = l_j - e_j$ et $\Delta_2 = l_l - e_l$,
- $[e_j, l_j]$ la fenêtre de temps liée au sommet de départ v_j,
- $[e_l, l_l]$ la fenêtre de temps liée au sommet d'arrivée v_l.

Plus les fenêtres de temps des deux arrêts de la requête sont étroites, plus cette composante sera grande.

La marge maximale de parcours

Il s'agit de la différence entre le temps de parcours idéal pour aller du sommet de départ au sommet d'arrivée d'une requête, et le temps maximum alloué par le passager. Pour une requête r_i de point de départ v_j et de point d'arrivée v_l :

$$C_{ride} = \frac{1}{\Delta Ride}$$

Où $\Delta Ride = (l_l - e_j) - t_{j,l}$ et $t_{j,l}$ est le temps minimal de trajet entre les deux sommets v_j et v_l.

Plus la marge fournie par le client est petite, plus cette composante est grande et moins il y a de chance d'insérer facilement la nouvelle requête.

Le nombre de clients

Plus il y a de clients à prendre en charge pour une requête, plus elle est difficile à satisfaire, car elle risque de violer les contraintes de capacité des véhicules. Pour une requête r_i nous avons :

$$C_{cap} = q_i$$

Où q_i est le nombre de personnes associées à la requête r_i.

Calcul de l'indice de difficulté I_{Dif}

On effectue une somme pondérée des trois composantes précédemment définies pour définir l'indice globale I_{Dif} par la formule suivante :

$$I_{Dif} = \alpha C_{fen} + \beta C_{ride} + \gamma C_{cap}$$

Avec α, β et γ trois constantes à définir pour favoriser une composante par rapport à l'autre.

Si I_{Dif} est grand, cela signifie que la requête est difficile à insérer.

Dans la suite nous définissons un autre indice pour trier les tournées selon la difficulté d'insertion d'une nouvelle requête.

5.3.3 Degré d'attraction d'une requête par une tournée

Cet indice permet, pour chaque requête, de ranger les tournées selon leur degré d'attraction de cette requête. Donc pour chaque requête, on lui associe une liste de tournées triée selon le degré d'attraction. En utilisant cet indice nous pourrons alors nous contenter de ne tester que les premières tournées dans la liste pour insérer une nouvelle requête et non l'ensemble des tournées disponibles, ce qui permet de diminuer le temps de calcul.

5.3.3.1 Le degré d'attraction

Le degré d'attraction est basé sur le modèle de Huff ([Huff, 1964]) qui a été introduit en 1964. Il s'agit d'un modèle de la classe des modèles d'Interaction Concurrentielle Spatiale, qui a été conçu pour être utilisé dans le domaine du géomarketing. L'objectif de ce modèle est d'estimer la probabilité P_{ik} qu'un consommateur au point i fréquente un magasin particulier au point k en faisant le rapport entre l'utilité de ce point de vente pour le consommateur et la somme des utilités de tous les points de vente concurrents pour ce même consommateur. Cette notion d'attractivité a été utilisé dans ([Erdoğan *et al.*, 2010]) pour résoudre un problème de voyageur de commerce avec profit et elle est largement utilisée pour résoudre des problèmes de localisation-allocation ([Drezner et Drezner, 2006]).

Définition 3

$$P_{ik} = \frac{U_{ik}}{\sum_{j=1}^{M} U_{ij}}$$

avec :

- U_{ik} : l'utilité du magasin k pour le consommateur i avec $U_{ik} = \frac{S_k}{D_{ik}^{\beta}}$,
- S_k : la taille du point de vente k (la surface du parking, le nombre de caisses disponibles,...),
- D_{ik} : la distance (ou le temps) minimale entre le consommateur i et le magasin k,
- β : paramètre mesurant l'importance de la distance dans la décision d'un consommateur de fréquenter tel ou tel magasin,
- M : le nombre de magasins en concurrence dans le secteur considéré.

Plus l'utilité U_{ik} est grande, plus le consommateur i aura tendance à être attiré par le point de vente k, il est à noter que dans la formule, la taille S_k du point de vente k peut être remplacée par une quelconque autre mesure de l'attractivité du magasin.

Par analogie on définit les différents paramètres que nous utilisons pour définir l'attraction d'une tournée par une nouvelle requête comme suit :

- U_{ik} : représente l'utilité de la tournée k pour la nouvelle requête r_i,
- S_k : un paramètre qui représente le potentiel de la tournée k à satisfaire une ou plusieurs requêtes supplémentaires. Plus ce paramètre est élevé, plus il est facile d'insérer une nouvelle requête dans cette tournée,
- D_{ik} : représente la distance entre la requête r_i et la tournée k, (le calcul de cette distance sera défini ultérieurement). Plus cette distance est faible, plus la requête r_i sera "attirée" par la tournée k.

Calcul du potentiel d'attractivité d'une tournée S_k :

Le paramètre S_k représente le potentiel de la tournée k à accepter de nouvelles requêtes. Il est défini par la somme pondérée de deux composants : le temps d'attente et la capacité résiduelle des véhicules :

$$S_k = aCap_k^{r_j} + bWait_k^{r_j}.$$

Nous nous intéressons maintenant à la définition du temps d'attente. On note W_u^k le temps d'attente du véhicule k lorsqu'il dessert le sommet v_u. Il s'agit donc d'un temps mort au sein de l'ordonnancement de la tournée durant lequel le véhicule reste immobilisé. Pour une nouvelle requête r_i de sommet de départ v_j et de sommet d'arrivée v_l, nous ne prenons en compte que les temps d'attente W_u^k, pour un véhicule k, qui ont lieu dans l'intervalle $[e_j, l_l]$. On a donc

$$Wait_k^{r_i} = \sum_{u=1}^{n} \alpha_u W_u^k$$

avec :

$$\alpha_i = \begin{cases} 1 & \text{si } A_u^k \in [e_j, l_l], \\ \frac{1}{e_j - A_u^k} & \text{si } A_u^k < e_j \\ \frac{1}{A_u^k - l_l} & \text{si } A_u^k > l_l \end{cases}$$

- $[e_j, l_j]$: la fenêtre de temps associée au sommet de départ v_j,
- $[e_l, l_l]$: la fenêtre de temps associée au sommet de destination v_l,
- A_u^k : l'heure de début de service au sommet situé à la position u dans la tournée k.

Plus $Wait_k^{r_i}$ est grand, plus le potentiel d'attraction de la tournée est grand.

La capacité résiduelle $Cap_k^{r_i}$ est le nombre de places disponibles dans le véhicule après l'embarquement ou le débarquement des passagers à un sommet de la tournée.

Comme pour les temps d'attente, on compte les places libres sur la partie de la tournée qui coïncide avec les plages horaires associées à la nouvelle requête. on a :

$$Cap_k^{r_i} = \sum_{u/A_u^k \in [e_j, l_l]}^{n} (Q_k - E_u^k)$$

Avec E_u^k le nombre de personnes à bord du véhicule après avoir servi le sommet qui se trouve à la position u de la tournée.

Calcul de la distance D_{ik} entre une requête r_i et une tournée k

D_{ik} représente la distance spatio-temporelle entre la tournée k et une requête r_i. Cette distance reflète la proximité spatiale mais aussi temporelle entre une requête et une tournée. En effet, on peut imaginer qu'il est plus facile et plus intéressant d'insérer une requête dans une tournée qui se trouve à proximité et qui tourne sur les mêmes plages horaires. Pour cela une distance euclidienne en 3 dimensions est utilisée : les deux premières dimensions représentent la position géographique (x, y) des sommets et la $3^{\text{ème}}$ dimension représente l'axe du temps que nous définissons un peu plus loin dans cette section. Calculer la distance entre une requête r_i et une tournée k revient à calculer une distance entre un segment qui relie les deux sommets de la requête et un polygone qui relie tous les sommets de la tournée. On définit quatre sommets particuliers :

- le sommet de départ v_j de la requête entrante r_i de coordonnées $\left(X_j^1, X_j^2, X_j^3\right)$,
- le sommet de destination V_l de la requête entrante r_i de coordonnées $\left(X_l^1, X_l^2, X_l^3\right)$,
- le sommet appartenant à la tournée k, noté v_p, tel que :
  $d(v_p, v_j) = \min_{u \in \{1,\ldots,n\}} \left(\sum_{v=1}^{3} \left(X_{[u]}^v - X_j^v \right)^2 \right)$. Il s'agit du sommet de la tournée le plus proche de V_j. Avec $X_{[u]}^v$, $\forall u \in 1, 2, 3$, sont les coordonnées du sommet qui se trouve à la position u dans la tournée k,
- le sommet appartenant à la tournée k, noté v_d, tel que :
  $d(v_d, v_l) = \min_{u \in \{1,\ldots,n\}} \left(\sum_{v=1}^{3} \left(X_{[u]}^v - X_l^v \right)^2 \right)$. Il s'agit du sommet le plus proche de v_l.

Notons que X_u^1 et X_u^2 d'un sommet v_u représentent les coordonnées géographiques (x, y) associées à ce sommet et la troisième dimension temporelle X_u^3 est obtenue de la manière suivante :

$$X_u^3 = \begin{cases} A_u^k & \text{si } v_u \text{ est un sommet de la tournée (les sommets des requêtes statiques),} \\ \frac{e_u+l_u}{2} & \text{si } v_u \text{ est un sommet d'une nouvelle requête entrante.} \end{cases}$$

Donc la distance D_{ik} entre une requête r_i et une tournée k est donnée par la formule suivante :

$$D_{ik} = \sqrt{\sum_{v=1}^{3} (X_p^v - X_j^v)^2} + \sqrt{\sum_{v=1}^{3} (X_d^v - X_l^v)^2}$$

Les données temporelles et spatiales n'ont pas le même ordre de grandeur, et pour éviter l'effet des grandes valeurs sur le calcul de distance, une normalisation à 1 des coordonnées est réalisée. Cette normalisation consiste à diviser les coordonnées de l'ensemble des sommets par la valeur maximale de chaque dimension : on divise les abscisses par l'abscisse maximale, les ordonnées par l'ordonnée maximale,...

5.3.3.2 Insertion d'une nouvelle requête dans une tournée

Une fois qu'une requête et une tournée sont sélectionnées, il faut vérifier la faisabilité de l'insertion de la requête sélectionnée dans la tournée. Chaque requête est composée de deux sommets à insérer : lors de l'insertion il faut donc vérifier la contrainte de capacité du véhicule ainsi que les contraintes de fenêtres de temps associées aux sommets des requêtes statiques.

Vérification de la capacité

Soit r_i une requête de q_i personnes à insérer dans une tournée k planifiée. Q_k est la capacité du véhicule et $E_{[u]}^k$ le nombre de personnes à bord du véhicule directement après avoir visiter le sommet situé à la position u de la tournée k.

Soit d (resp. f) la position d'insertion du sommet de départ v_j (resp. d'arrivée v_l) de la requête r_i dans la tournée k. Pour vérifier que la tournée reste réalisable du point de vu de la capacité, il suffit de vérifier que $\forall u \in \{d+1, ..., f-1\}$, $E_{[u]}^k + q_i \leq Q_k$

Vérification des fenêtres de temps

Les tournées sur lesquelles nous essayons d'insérer de nouvelles requêtes ont été calculées à partir des réservations (requêtes statiques) enregistrées au préalable et validées. C'est-à-dire que les horaires de la desserte sont fixés et confirmés pour les passagers. Donc lors de l'insertion des nouvelles requêtes dynamiques, nous ne pouvons pas nous permettre de violer la contrainte des fenêtres de temps associées aux sommets d'une requête statique.

Notons p la position d'insertion du sommet de départ de la requête dynamique et d la position d'insertion du sommet d'arrivée. Les sommets situés avant la position p de la tournée ne nécessitent aucune mise-à-jour ou vérification. Pour les sommets situés entre p

(inclus) et d (exclus) il faut mettre à jour l'heure d'arrivée et le temps d'attente à chaque sommet et éventuellement vérifier les contraintes de fenêtres de temps.

On commence par calculer l'heure d'arrivée au sommet situé à la position p dans la tournée qui est égale à $A^k_{[p]} = A^k_{[p-1]} + s_{[p-1]} + t_{[p-1,p]}$, avec $s_{[p-1]}$ le temps de service au sommet situé à la position $p-1$ dans la tournée et $t_{[d-1,d]}$ est le temps nécessaire pour aller du sommet situé à la position $p-1$ vers son successeur dans la tournée :

- si $A^k_{[p]} < e_{[p]}$, on est en avance pour un sommet de départ, donc le véhicule doit attendre et le temps d'attente est $w^k_{[p]} = e_{[p]} - A^k_{[p]}$ et la date de début de service devient $A^k_{[p]} = e_{[p]}$,
- si $A^k_{[p]} > l_{[p]}$, l'heure d'arrivée est maintenue même si on dépasse la fenêtre de temps (c'est une requête dynamique). Cette violation se répercutera sur la qualité du service.

Pour la vérification des sommets suivants (situés après la position d) dans la tournée, on utilise le même processus, on met à jour l'heure d'arrivée, éventuellement les temps d'attente. Par contre, puisque entre la position p et d (exclus) il n'y a que les sommets des requêtes statiques, s'il existe un sommet v_i pour lequel $A^k_i > l_i$, cela signifie que l'insertion est impossible.

On remarque que le fait d'insérer un sommet à la position p aura toujours pour effet de déplacer le reste de l'ordonnancement vers la droite. Les sommets situés à droite de p ne pourront pas être servis plus tôt qu'avant l'insertion, dans le meilleur des cas ils seront servis à la même heure qu'avant si le temps de service du sommet inséré est nul.

Nous introduisons une valeur « gap » qui représente la quantité de décalage vers la droite effectuée pour insérer un sommet à la position p. On peut noter que ce gap ne décale pas directement les heures de début de service des autres sommets de la tournée, car ce gap peut être absorbé par des éventuelles temps d'attente qui peuvent avoir lieux aux sommets situés à droite de la position p. Il faut noter que si la valeur du gap devient nulle à un sommet donné alors la suite de la tournée sera inchangé par rapport à la tournée initiale.

Cette propriété sera utilisée pour tenter de gagner un peu de temps dans la mise-à-jour de l'ordonnancement et la vérification de la tournée.

On considère l'exemple de la Figure 5.2 dans lequel un sommet v_i est inséré à la position i dans une tournée k. Le gap égale à :

$gap = t_{[i-1,i]} + s_{[i]} + t_{[i,i+1]} - t_{[i,i+1]}$.

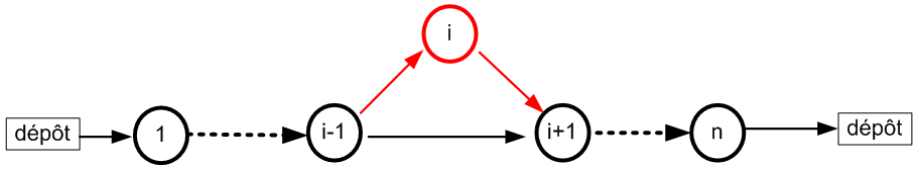


FIGURE 5.2 – Insertion de deux sommet dans une tournée

Pour chaque sommet v_j situé après v_i dans une tournée k de n sommets, $j \in \{i+1, ..., n\}$, si $W^k_j > 0$ alors :

$$\begin{cases} gap = \max\left(0, gap - W_j^k\right), \\ W_j^k = \max\left(0, W_j^k - gap\right) \end{cases}$$

Si on arrive à un sommet pour lequel $gap = 0$, on sait que la suite de la tournée ne sera pas modifiée et donc on peut passer directement sans vérifier le reste des sommets.

Lors de l'insertion d'une requête on peut en théorie être amené à tester toutes les possibilités d'insertion des deux sommets de départ et d'arrivée ce qui représente $(n + 1)(n + 2)/2$ possibilités, avec n le nombre de sommets dans une tournée. Ce nombre peut être réduit par l'utilisation de règles d'élimination simples.

On commence l'insertion d'une nouvelle requête dans une tournée après le sommet qui suit la position du véhicule lors de la soumission de cette requête. Supposons que nous insérons le sommet de départ de la nouvelle requête à la position i et le sommet d'arrivée à la position j de la tournée, si nous observons un dépassement de capacité après l'insertion de la requête sur la portion de tournée entre i (inclus) et j, alors il ne sert à rien de tester les autres configurations qui placent j plus tard dans la tournée ou qui placent i avant le sommet qui pose problème (sommet auquel la capacité est violée), il faut directement placer i et j après ce sommet qui pose problème. La Figure 5.3 montre une tournée de 2 requêtes (r_1 de 2 personnes, $1 \rightarrow 2$ et r_2 de 3 personnes, $3 \rightarrow 4$), un véhicule de capacité 6 et une requête dynamique r_3 de 2 personnes à insérer $(5 \rightarrow 6)$.

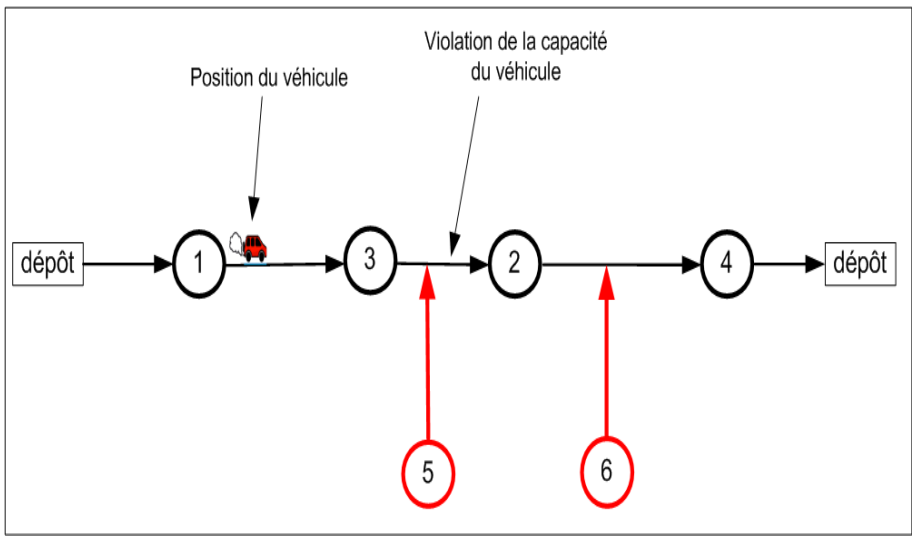


FIGURE 5.3 – Violation de la contrainte de capacité lors de l'insertion d'une requête

On voit sur l'exemple que le point de départ de la requête ne peut pas être inséré directement après le sommet 3, sinon la capacité du véhicule sera violée. Donc il faut chercher une position après le sommet 2 (voir Figure 5.4)

Après avoir testé toutes les insertions possibles pour une requête, nous obtenons un ensemble de solutions qui sont triées selon la règle de dominance au sens de Pareto relativement aux trois critères. Seules les solutions non-dominées sont conservées. Or, pour

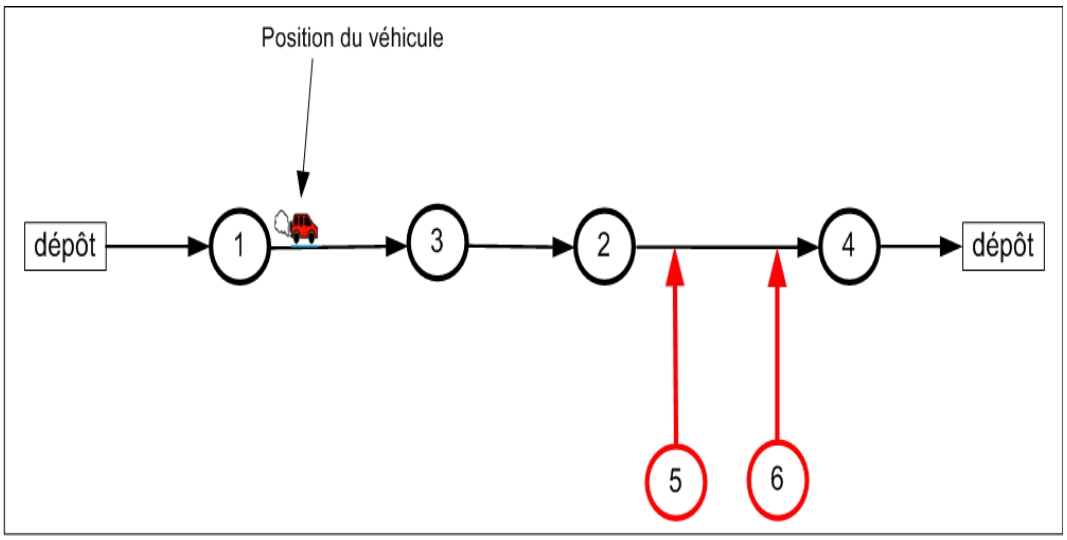


FIGURE 5.4 – Insertion d'une requête sans violation de contrainte de capacité du véhicule

continuer à insérer les autres requêtes nous avons besoin de choisir une seule solution. Puisqu'on suppose qu'on ne connait pas les préférences du décideur qui vont nous permettre de trier les solutions non-dominées, nous avons utilisé une distance de Tchebychev calculée pour chaque solution par rapport à une solution de référence qui peut être le vecteur *Idéal* calculé à partir de l'ensemble des solutions non-dominées. La solution qui a la plus petite valeur de la distance de Tchebychev par rapport au vecteur idéal, sera sélectionnée pour être utilisée pour les futures insertions. L'utilisation de la distance de Tchebycheff comme méthode d'agrégation s'explique par notre désir d'obtenir des solutions équilibrées. La formulation pour l'évaluation d'une solution $Z = (Z_1, ..., Z_K)$ à K critères, est la suivante :

$$\left\|Z - Z^{\texttt{idéal}}\right\| = \max_{1 \leq i \leq k} \lambda_i \left|Z_i - Z_i^{\texttt{idéal}}\right| + \epsilon \sum_{i=1}^{k} \left|Z_i - Z_i^{\texttt{idéal}}\right|$$

avec :

- $Z^{\texttt{idéal}}$: le vecteur idéal calculé à partir de l'ensemble des solutions non-dominées,
- ϵ une constante, $\epsilon = 10^{-3}$,
- $\lambda_i = 1/Max(Z_i)$ permet de normaliser les trois critères.

L'algorithme complet de l'heuristique d'insertion est donné dans la Figure (5.5).

5.4 Méthode arborescente tronquée

5.4.1 Description générale

Dans cette section nous présentons la deuxième méthode qui a été développée pour résoudre la version dynamique du problème de Transport à la Demande. Contrairement à la première méthode développée, l'idée ici est de remettre en cause la structure des tournées, pour tenter de trouver des solutions de meilleure qualité. Pour cela nous avons opté pour

```
Algorithme d'insertion
Entrée :
    S1 :sommet de départ de la requête dynamique ;
    S2 :sommet d'arrivée de la requête dynamique ;
    Requêtes[] : liste des requêtes à insérer ;
    Tournees[] : liste des tournées disponibles ;
    MAX-TOURNEE : le nombre maximal de tournées dans lesquelles l'insertion de la nouvelle requête sera testée ;

Trier les requêtes par indice de difficulté d'insertion, du plus difficile au moins difficile
Pour chaque Requête r de Requêtes[] Faire ;
    S1 = le sommet de départ de r ;
    S2 = le sommet d'arrivée de r ;
    Trier les tournées par degré d'attraction par rapport à la requête r ;
    Pour i de 0 à MAX-TOURNEE Faire // on ne teste que les MAX-TOURNEE premières tournées ;
        Initialiser la position S1 et S2 ;
        Tant qu'il reste des insertions possibles Faire ;
            Vérifier la faisabilité de l'insertion ;
            Si l'insertion est faisable Alors ;
                Evaluer la solution selon les trois fonctions objectif ;
                Insérer dans le front de Pareto qui contient les solutions non-dominées générées par l'insertion de r ;
                Placer S1 et S2 à une position suivante dans la tournée ;
            Sinon // On a violé une contrainte
                Si violation de capacité sur sommet X Alors ;
                    Placer S1 et S2 juste après X ;
                Fin Si ;
            Fin Si ;
        Fin Tant Que ;
    Fin Pour ;
    Choisir la meilleure solution à l'aide de la distance de Tchebychev ;
Fin Pour ;
Appliquer l'algorithme 'Optimal Timing ;
```

FIGURE 5.5 – Heuristique d'insertion pour résoudre le problème du Transport à la Demande dynamique

l'utilisation d'une méthode arborescente tronquée, plus précisément d'une recherche par faisceaux améliorée par une phase de récupération des noeuds, cette méthode est appelée « *Recovering Beam Search* ». Dans la suite nous décrivons les principes généraux de cette heuristique, puis nous expliquons en détail comment cette heuristique a été adaptée pour résoudre notre problème.

La méthode *Beam Search* est une recherche arborescente tronquée où seule une partie des noeuds, de chaque niveau, est explorée. L'idée est d'explorer uniquement la descendance des meilleurs noeuds de chaque niveau. Ces derniers forment le faisceau et leur nombre représente la largeur de celui-ci. Le critère de qualité qui permet à un noeud de faire partie du faisceau est défini par l'expert. Il s'agit généralement d'une borne inférieure, d'une borne supérieure ou d'une combinaison linéaire des deux.

Récemment, plusieurs plusieurs techniques ont été développées pour améliorer la performance de la BS. Par exemple, ([Honda *et al.*, 2003]) proposent un Backtracking Beam Search (BBS) pour résoudre un problème de flowshop multi-objectif et une extension de cette méthode a été proposée aussi dans ([Della Croce *et al.*, 2002]) et ([Della Croce *et al.*, 2004]) et s'appelle *recherche par faisceaux améliorée par récupération des noeuds* (*Recovring Beam Search*), noté RBS dans la suite du texte. L'avantage majeur de ce type d'heuristique est de

considérer un nombre limité d'ensembles de solutions en parallèle. A chaque itération, les noeuds fils sont engendrés et seulement ceux qui contiennent les solutions les plus prometteuses sont retenus. Quand un noeud fils est retenu, la phase de récupération consiste à ce moment là, à appliquer une recherche locale pour essayer de remettre en question les décisions possiblement erronées qui ont conduit à considérer ce noeud. Plusieurs approches RBS ont été aussi proposées pour résoudre d'autres problèmes, notamment dans le domaine d'ordonnancement ([Valente et Alves, 2005], [Ghirardi et Potts, 2005], [Lim *et al.*, 2006] parmi d'autres).

Dans le cas de notre problème un noeud représente un ensemble d'ordonnancements (tournées partielles) et il y a autant de tournées que de véhicules. Le noeud fils est construit à partir du noeud courant (son noeud père), en ajoutant à la fin (ou à une position appropriée) d'un ordonnancement un sommet parmi ceux restant à placer. Seules les solutions partielles faisables vont être gardées et parmi tous les fils retenus à un niveau de profondeur de l'arbre, seulement un nombre limité d'entre eux est sélectionné. Ce nombre est appelé la *largeur du faisceau* est noté w. Ceci veut dire que les noeuds d'un niveau donné vont être triés et sélectionnés en utilisant une fonction d'évaluation. Après la sélection de w meilleures solutions, on appliquera la phase de récupération. Ensuite, les noeuds fils de ces w noeuds sélectionnés seront crées et le processus est itéré jusqu'aux noeuds feuilles.

Lorsqu'on cherche à implémenter un algorithme de *Recovring Beam Search* les cinq mécanismes suivants doivent être définis :

- le schéma de branchement,
- le filtre,
- l'évaluation,
- la sélection,
- la récupération.

Dans la suite nous décrivons chacun de ces mécanismes, ainsi que leur mise en place dans le cas de notre problème.

5.4.2 Etat initial

La méthode que nous proposons sera utilisée pour résoudre un problème dynamique, c'est-à-dire qu'il y a une solution courante qui est en train d'être exécutée et que nous avons un ensemble de nouvelles requêtes à insérer. Pour cette méthode nous avons décidé de remettre en cause tout ou partie des tournées déjà planifiées. L'état initial est constitué alors d'une liste de sommets non desservis que nous allons insérer grâce au schéma de branchement. Il faut bien garder à l'esprit pour la suite que même si tous ces sommets sont mélangés (statiques et dynamiques) dans la même liste, ils ont des caractéristiques différentes qui vont influencer sur la méthode lors de leur insertion. La liste des sommets à insérer est composée :

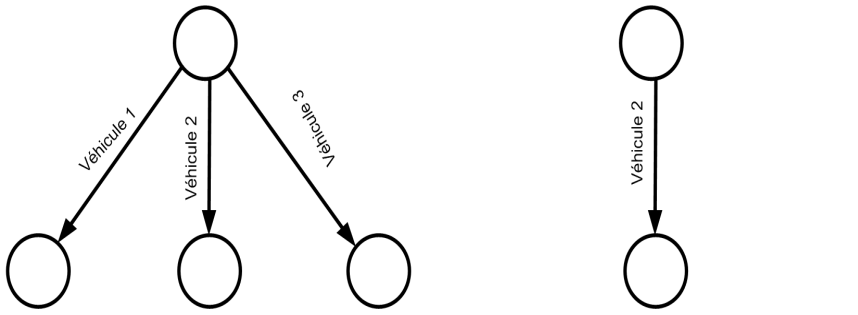
- des sommets issus des requêtes dynamiques. Ces sommets sont libres d'être affectés à n'importe quel véhicule et leurs fenêtres de temps peuvent être dépassées, si besoin, en dégradant la qualité de service,
- des sommets issus des requêtes statiques non desservies. Ce sont les requêtes qui étaient planifiées mais dont les passagers n'ont pas encore embarqués dans un véhicule. Ces sommets sont aussi libres d'être affectés à n'importe quel véhicule, mais

leurs fenêtres de temps sont inviolables,
- des sommets issus des requêtes en cours de traitement. Ce sont des sommets de dépose, liés à des sommets de montée déjà servis. Cela signifie que les clients sont dans un véhicule précis au moment du réordonnancement. Ces sommets seront donc obligatoirement affectés au même véhicule et leurs fenêtres de temps sont inviolables.

Cette liste de sommets est triée selon un ordre de priorité et les sommets seront insérées par la RBS dans cet ordre. Nous avons choisi de trier les sommets par la valeur de la borne supérieure de la fenêtre du temps l_i. Il faut noter que les intervalles de temps doivent être spécifiées de telle façon que la borne supérieure de la fenêtre du temps d'un sommet de départ soit inférieure strictement à la borne supérieure de la fenêtre du temps du sommet de destination de la même requête.

5.4.3 Le schéma de branchement

Le schéma de branchement permet, à partir d'un niveau de l'arbre donné, de créer le niveau suivant. Le branchement s'effectue sur l'affectation des sommets aux véhicules et les sommets sont triés selon la borne supérieure des fenêtres de temps l_i, à chaque niveau de l'arborescence, on insère le prochain sommet issu de la liste triée. A partir d'un noeud père, on créé alors un ensemble de noeuds fils en affectant ce sommet à l'ensemble des véhicules possibles. Si le sommet est lié à un véhicule alors un seul noeud fils sera crée. Si le sommet est libre alors nous aurons autant de noeuds fils que de véhicules (Figure 5.6).



(a) Branchement pour un sommet libre (b) Branchement pour un sommet lié

FIGURE 5.6 – Le branchement selon le type de sommet

Lors du branchement on s'intéresse seulement au choix des véhicules pour leur affecter de nouveaux sommets. Pour cela un mécanisme d'ordonnancement des sommets doit être défini pour mettre en ordre les sommets affectés à chaque véhicule. Pour cela, nous avons utilisé une règle que nous appellerons « insertion au plus tard ». C'est-à-dire que le nouveau sommet est placé à la fin d'ordonnancement, mais en respectant les contraintes des fenêtres de temps qui existent (si le sommet à insérer est de type « statique »). Nous avons donc deux possibilités :
- le sommet est de type « dynamique » associé à une fenêtre de temps souple, le sommet est inséré à la fin de l'ordonnancement,

– le sommet est de type « statique », dans ce cas le sommet est placé à la fin, mais il est remonté, si nécessaire, jusqu'à respecter la contrainte des fenêtres de temps (Figure 5.7).

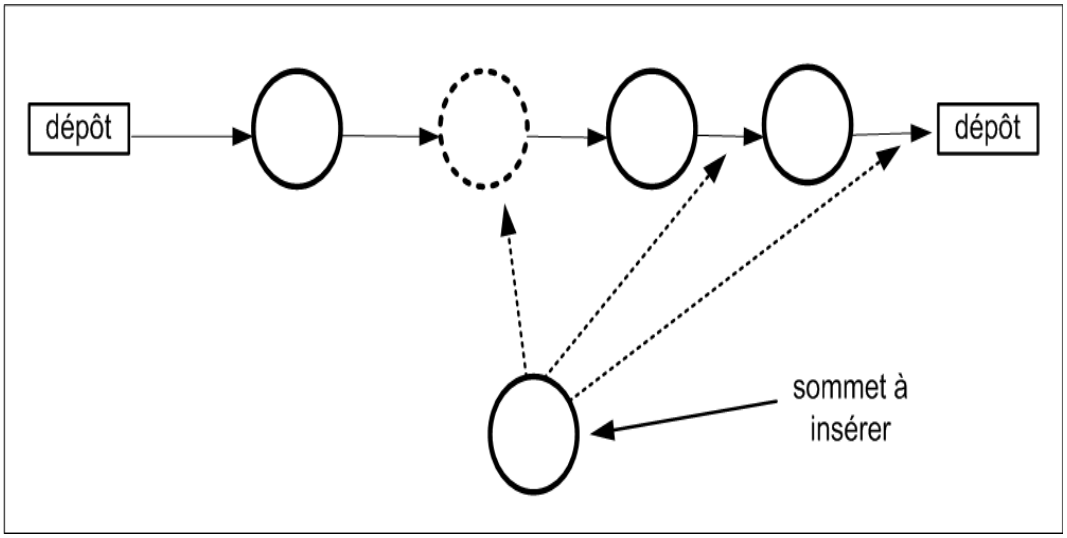


FIGURE 5.7 – Affectation d'un sommet statique

5.4.4 Le filtre

C'est un mécanisme qui s'utilise pour éliminer très rapidement les mauvais noeuds. Cela peut être fait en utilisant des règles de dominance entre noeuds et un noeud dominé sera alors coupé. Dans le cas de notre problème, nous avons mis en place deux filtres :

– lors du placement d'un sommet, affecté à un véhicule, il faut respecter la contrainte du capacité du véhicule,
– il faut aussi vérifier que les fenêtres de temps des requêtes statiques sont respectées.

Ces deux aspect ont été détaillés dans la section (5.3.3.2).

Pour plus d'informations sur la manière de vérifier efficacement ces deux contraintes, voir la partie « faisabilité d'une insertion » dans la section précédente consacrée à « l'heuristique par insertion ».

5.4.5 L'évaluation

Dans une méthode RBS, chaque noeud de l'arbre de recherche s est évalué grâce à une fonction $f(s) = \gamma LB_s + (1-\gamma)UB_s$ où γ est un poids donné et LB_s et UB_s font référence respectivement à une borne inférieure et à une borne supérieure au noeud s. Dans la procédure classique par séparation et évaluation (PSE), l'évaluation du noeud s est généralement calculée en utilisant une borne inférieure et en prenant $\gamma = 1$. Cette évaluation est alors utilisée pour détruire tous les noeuds indésirables. Mais dans une RBS cette fonction d'évaluation a seulement pour but de guider l'heuristique en choisissant des noeuds prometteurs et en écartant les autres.

Dans notre problème, il est très difficile de calculer une borne inférieure. Pour cela nous avons décidé de nous passer d'une borne inférieure et de ne travailler qu'avec une borne supérieure. Cette borne supérieure doit être la plus précise possible, mais très rapide en temps de calcul, car elle sera appelée à chaque nouveau noeud généré. Pour calculer la borne supérieure nous utilisons l'heuristique par insertion (HI) que nous avons développée dans la première partie de ce chapitre. L'heuristique prend en entrée :

- la solution partielle (s_N) du noeud courant comme solution initiale,
- la liste des sommets restants à insérer,
- l'heure de soumission.

Dans la version originelle de l'heuristique par insertion, nous n'insérions que des requêtes dites « dynamiques ». Elles ont la particularité d'avoir des fenêtres de temps souples, c'est-à-dire que si besoin on pouvait servir le client en dehors des fenêtres de temps souhaitées. Cette caractéristique permet d'affirmer que l'heuristique donnera toujours une solution réalisable, car dans le pire des cas on peut insérer la requête à la fin de la tournée. Par contre ici nous utilisons cette heuristique pour insérer aussi certaines requêtes statiques. Ces requêtes ont des fenêtres de temps dures, dans ce contexte nous ne pouvons plus garantir que l'heuristique mènera vers une solution complète et réalisable. Il se peut que dans le déroulement de l'algorihme, l'heuristique HI ne réussisse pas à trouver de place pour un sommet à fenêtre dure. Dans ce cas l'heuristique s'arrête et renvoie une solution partielle s_{HI} et une liste de sommets non placés L_{snp}.

L'évaluation de la solution courante s_N s'effectue alors de la manière suivante :

- si la solution s_{HI} donnée par l'heuristique est complète (c'est-à-dire si $L_{snp} = \emptyset$) alors on calcule la valeur des trois critères (coût de transport, insatisfaction des passagers et quantité de CO_2),
- si la solution s_{HI} est incomplète (c'est-à-dire si L_{snp} n'est pas vide), on calcule aussi la valeur des trois critères de la solution partielle s_{HI} et on ajoute à chacun une pénalité. Cette pénalité est égale aux nombres de sommets non placés multiplié par une très grande constante.

On part du principe que même si l'heuristique par insertion HI n'a pas trouvé une solution complète, il ne s'agit que d'une borne supérieure. On peut donc espérer qu'il existe une solution complète et réalisable que RBS pourrait retrouver en poursuivant les branchement et que l'heuristique HI n'a pas pu retrouver.

5.4.6 La sélection

L'objectif est de sélectionner les w meilleurs noeuds que l'on explorera ensuite. Dans notre cas, nous disposons d'un ensemble de noeuds contenant des solutions partielles qui ont été évaluées selon les trois critères. Mais nous ne disposons pas de borne inférieure. De plus notre borne supérieure n'est pas une valeur scalaire, mais un triplet de valeurs.

Nous pourrions utiliser une combinaison linéaire de ces critères, mais nous avons préféré mettre en place un algorithme de sélection différent. Nous construisons le front constitué des optima de Pareto des solutions partielles courantes. La sélection des w meilleurs noeuds est effectuée en utilisant la distance de Crowding. Il faut noter que la valeur de pénalité utilisée lors de l'évaluation des solutions incomplètes s_HI, fera qu'on sélectionnerera pas

de noeuds menant à des solutions incomplètes $s_H I$ s'il en existe qui mènent à des solutions complètes.

5.4.7 La récupération

Il s'agit de la dernière phase de la méthode. Cette phase a pour but de revenir sur les mauvaises décisions qui auraient pu être prises pour conduire à ce noeud. Cette phase est appliquée aux w noeuds sélectionnés précédemment et elle consiste à appliquer une recherche locale sur l'ordonnancement partiel construit à chaque noeud. L'objectif est de faire des modifications au niveau de l'ordonnancement des sommets affectés à chaque véhicule pour tenter d'améliorer la solution. Dans la suite nous décrivons le mécanisme de récupération pour un noeud :

L'opérateur de voisinage utilisé consiste à prendre le sommet qui se trouve à la fin de chaque tournée partielle et à tenter de l'insérer à toutes les positions possibles dans la même tournée (opérateur INSERT). Une liste de voisins est générée mais ne sont conservés que les voisins « faisables » qui respectent les contraintes suivantes :

- le sommet de destination d'une requête doit être toujours placé après son sommet de départ dans la même tournée partielle,
- il faut respecter les contraintes de capacité du véhicule,
- il faut respecter les contraintes de fenêtres de temps relatives aux requêtes statiques.

Chaque voisin généré, pour une tournée partielle est évalué à l'aide d'une combinaison linéaire des critères de coût et de qualité de service (on ne prend pas en compte les émissions de CO_2, car pour un même véhicule, le coût de transport et l'émission de CO_2 sont corrélés). Les valeurs des deux critères sont calculées sur les solutions partielles uniquement. Si le meilleur voisin est de meilleur qualité que la solution initiale, il prend sa place et on réitère l'opération jusqu'à ne plus trouver de voisins améliorants la solution.

Ce mécanisme implique que l'on peut "sauter" d'un noeud de l'arbre à un autre noeud de même niveau.

La Figure (5.8) montre un exemple de séparation des noeuds dans une RBS sur un exemple à 2 requêtes r_1 $(1 \rightarrow 2)$ et $(3 \rightarrow 4)$ et 3 véhicules hétérogènes. A l'état initial, les quatre sommets sont triés selon leur valeur de l_i croissant et aucun sommet n'est affecté à un véhicule. Le sommet v_1 est libre : on peut donc l'affecter à n'importe quel véhicule qui a une capacité supérieure ou égale au nombre de personnes de la requête r_1. On construit alors trois solutions partielles qui sont évaluées et triées selon la dominance de Pareto relative aux trois critères. La solution partielle du noeud 2 est sélectionnée parce qu'on suppose, par exemple, qu'elle domine les deux autres solutions.

Dans le niveau deux de l'arbre nous affectons de la même façon le sommet v_3 qui est un sommet de départ libre. A partir de la solution sélectionnée précédemment on peut donc construire trois autres solutions. Les trois solutions sont maintenues parce qu'on considère qu'elles ne sont pas dominées. Dans le troisième niveau de l'arbre, nous affectons le sommet v_2 qui est le sommet de destination lié au sommet de départ v_1 et puisque ce dernier est affecté au véhicule 2 alors le sommet v_2 doit être obligatoirement affecté au véhicule 2. Pour cela seulement un noeud-fils est généré à partir de chacun des trois noeuds générés

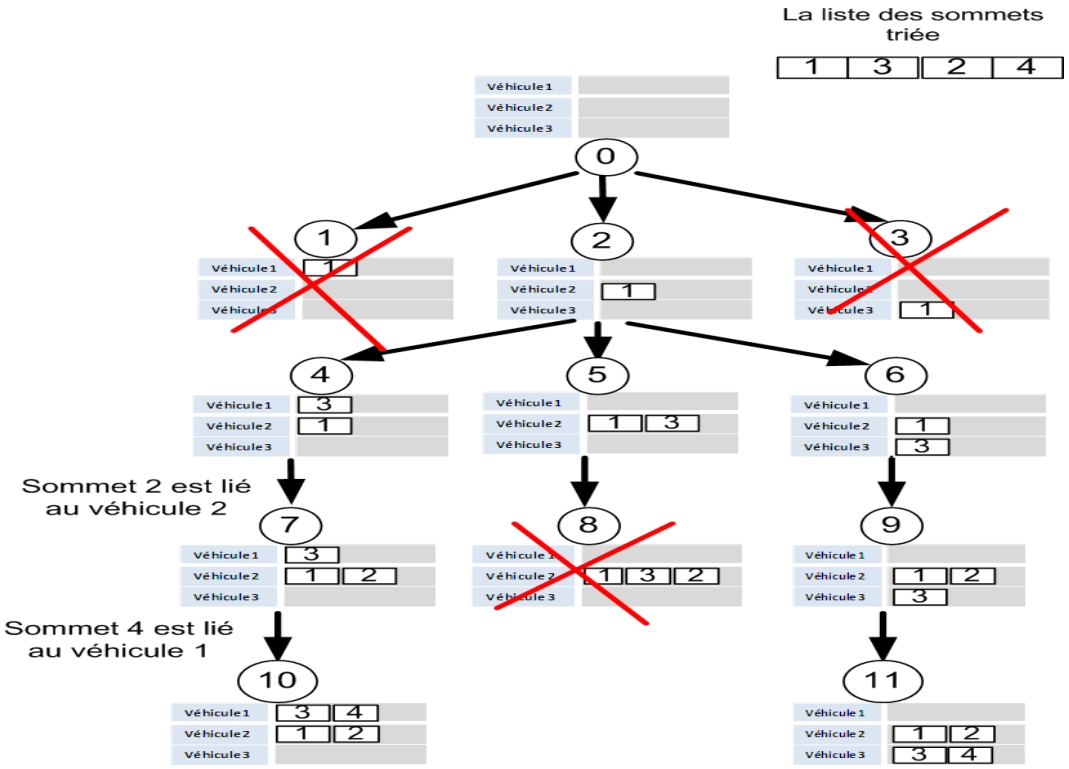


FIGURE 5.8 – Exemple de séparation des noeuds de la RBS

précédemment. Dans le niveau quatre de l'arbre nous affectons le sommet v_4 de la même façon que le sommet v_3.

5.5 Tests expérimentaux

Dans cette section, nous présentons des tests expérimentaux qui ont été menés sur les deux heuristiques en utilisant des instances générées aléatoirement. Dans un premier temps, nous présentons quelques statistiques sur l'heuristique par insertion, puis sur la méthode RBS. Dans un deuxième temps, nous comparons les deux méthodes entre elles, ainsi qu'avec des résultats calculés sur les mêmes jeux de données à l'aide de la méthode NSGA-II.

5.5.1 Les données

Les données liées aux requêtes et aux véhicules sont générées aléatoirement, comme expliqué dans le Chapitre 4 (section **??**). Nous avons considéré 30 catégories d'instances. Chaque catégorie est définie par un nombre de véhicules M, avec $M \in \{2, 4, 6\}$, un nombre de requêtes statiques n, avec $n \in \{10, 20, 30, ..., 100\}$ et 10 requêtes dynamiques à insérer pour toutes les catégories. Ces requêtes ont été générées uniformément sur l'horizon de planification, c'est-à-dire dans l'intervalle $[0, 1440]$. Pour chaque catégorie 10 instances sont

générées aléatoirement. Donc, au total, nous avons 300 instances à tester.

Lors des expérimentations, il a fallu simuler le comportement de l'opérateur de transport lorsque celui-ci, confronté à un ensemble d'optima de Pareto, doit choisir "l'optima qu'il préfère". Nous avons fait le choix, arbitraire, d'utiliser la distance pondérée augmenté de Tchebychev ([T'kindt et Billaut, 2006]) pour modéliser les préférences de l'opérateur de transport.

Nous supposons que cet opérateur va choisir un optimum de Pareto au plus proche, au sens de cette distance, de la solution idéale $Z^{\texttt{idéal}}$. Celle-ci calculée à partir de l'ensemble des optima de Pareto au sein duquel l'opérateur doit choisir.

La formulation pour l'évaluation d'une solution $Z = (Z_1, Z_2, Z_3)$ avec la distance de Tchebychev pondérée augmentée est la suivante :

$$\left\|Z - Z^{\texttt{idéal}}\right\| = \max_{1 \leq i \leq 3} \lambda \left|Z_i - Z_i^{\texttt{idéal}}\right| + \epsilon \sum_{i=1}^{3} \left|Z_i - Z_i^{\texttt{idéal}}\right|$$

avec :

- Z_1 : le coût de transport,
- Z_2 : l'insatisfaction des passagers,
- Z_3 : la quantité de CO_2 émise par les véhicules,
- $Z^{idéal}$: le vecteur idéal calculé à partir de l'ensemble des solutions non-dominées,
- ϵ une constante, $\epsilon = 10^{-3}$,
- $\lambda = 1/Max(Z_i),\ \forall i \in \{1, 2, 3\}$ permet de normaliser les trois critères.

Pour lancer des tests avec l'heuristique par insertion nous avons utilisé comme solution de départs, l'une des solutions du front de Pareto généré par NSGA-II pour les mêmes instances avec seulement les requêtes statiques.

De plus, afin de tester la dégradation de la qualité des algorithmes en fonction de l'avancement de l'heure de soumission, nous avons décidé de tester ce jeu de données sur 5 heures différentes. Ces 5 heures de soumission sont choisies de manière à être espacées régulièrement dans la journée, entre le temps 0 et le temps où toutes les tournées programmées sont finies. Pour une heure donnée t, un nombre de requêtes dynamiques sont insérées, ce nombre de requêtes vari d'une instance à l'autre. En ajoutant cette variable nous avons donc 300*5 =1500 instances par algorithme à insérer. Les tableaux de résultats de ces tests seront rapportés dans l'annexe de ce document.

Les heuristiques ont été implémentées en C++ et exécutées sur une machine Intel Core Duo 3GHZ et 2GO de RAM.

5.5.2 Les résultats expérimentaux de l'heuristique HI

Dans la suite nous présentons quelques résultats liés à l'heuristique par insertion HI. Nous présentons l'évolution du temps de calcul en fonction de la taille du problème ainsi que la performance de l'heuristique en terme de pertinence de l'indice d'attraction.

Tout d'abord il faut fixer les paramètres qui sont utilisés dans cette heuristique, notamment pour le calcul de l'indice de difficulté des requêtes et le degré d'attraction des tournées. La Table 5.1 montre les valeurs des différents paramètres qui ont été retenues.

α	β	γ	a	b
1	1	1	1	1

TABLE 5.1 – Les paramètres de l'heuristique HI

La Figure 5.9 montre l'évolution du temps d'exécution de l'heuristique HI en fonction du nombre de requêtes statiques. Le test est effectué sur l'insertion de 10 requêtes dynamiques à $t = 0$ (tous les véhicules sont au dépôt et aucune requête statique n'est encore servie). Les résultats obtenus ont été produits en explorant toutes les tournées pour insérer une nouvelle requête. L'exploration de tournées étant l'un des calculs les plus couteux dans notre algorithme, on peut estimer qu'un gain important pourrait être pris si on décide de n'explorer qu'un ensemble réduit de ces tournées, c'est-à-dire on ne teste que les *MAX-TOURNEE* premières tournées dans l'algorithme HI.

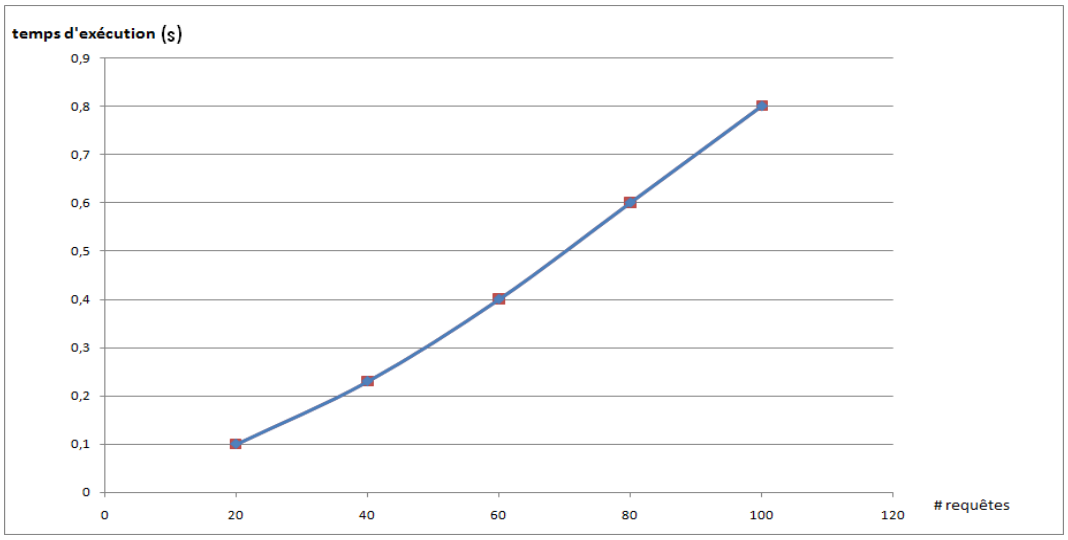


FIGURE 5.9 – Evolution du temps d'exécution en fonction de la taille du problème

Evaluation de l'efficacité du degré d'attraction.

Ce test a pour but d'évaluer l'efficacité du degré d'attraction que nous avons défini pour trier les tournées selon le potentiel de chaque tournée à accueillir facilement une requête donnée.

rtion au début

Le Tableau 5.2 donne les résultats obtenus avec l'heuristique HI sur les 30 catégories d'instances. Chaque ligne correspond aux résultats d'une catégorie d'instance : la colonne « Instance » représente le nom de la catégorie d'instance, la colonne « Req.Stat » contient le nombre de requêtes statiques, la colonne « Req.Dyn » le nombre de requêtes dynamiques, la colonne « Véhicule » contient le nombre de véhicules (les véhicules sont hétérogènes, voir le tableau des véhicules dans l'annexe ? ? ? ? ? ? ?), la colonne « CPU » donne le temps d'exécution moyen en secondes, la colonne « $\#Front$ » donne le nombre moyen de solutions

Instance	#Req.Stat.	Vehicule	CPU	#Front	Choix
r-1	10	2	0,0797	2,8	1
r-2	10	4	0,0637	3,9	1,1
r-3	10	6	0,0574	2,9	1
r-4	20	2	0,1161	1,5	1
r-5	20	4	0,1108	4,7	1,9
r-6	20	6	0,1062	4,6	2
r-7	30	2	0,1963	1,9	1
r-8	30	4	0,1554	4,8	2,1
r-9	30	6	0,1474	4,7	2,5
r-10	40	2	0,298	1,7	1
r-11	40	4	0,2182	5,4	2,2
r-12	40	6	0,2081	5,9	3
r-13	50	2	0,4153	2,3	1
r-14	50	4	0,2855	4,6	2,1
r-15	50	6	0,2503	4,8	3,1
r-16	60	2	0,5868	2,6	1
r-17	60	4	0,3513	2,9	2,1
r-18	60	6	0,3219	3,4	2,8
r-19	70	2	0,7286	2,4	1
r-20	70	4	0,4778	3,8	2,2
r-21	70	6	0,3643	5,4	3,3
r-22	80	2	0,8121	1,6	1
r-23	80	4	0,578	2,5	2,2
r-24	80	6	0,4622	3,5	3
r-25	90	2	1,1942	1,4	1
r-26	90	4	0,612	3,3	1,9
r-27	90	6	0,5048	4,1	2,5
r-28	100	2	1,0826	2,2	1
r-29	100	4	0,7647	2,2	2,1
r-30	100	6	0,6241	3,5	2,8

TABLE 5.2 – Les résultats de l'insertion à t=0

non-dominées générées par l'heuristique et la colonne « Choix » donne le classement moyen (selon le degré d'attraction) des tournées qui ont été choisies pour accueillir les nouvelles tournées.

De l'observation du Tableau 5.2, il ressort que le temps d'exécution moyen ne dépasse pas une seconde même pour les instances à 100 requêtes. La rapidité de cette heuristique permet qu'elle soit utilisée pour répondre très rapidement à la question : existe-t-il une insertion possible, voire même intéressante, d'une requête dynamique ? Elle est donc utilisable dans un contexte "temps réel" où l'opérateur de transport doit fournir une réponse à l'usager, par exemple au téléphone. Pour la taille du front de Pareto généré par cette heuristique, les résultats montrent que ce front ne contient pas assez de solutions : sur toutes les catégories d'instances en moyenne il ne dépasse pas 4 solutions. Ceci peut s'expliquer par l'utilisation de la distance de Tchebychev pour sélectionner une solution d'une insertion à l'autre et que c'est seulement lors de la dernière insertion que nous conservons toutes les solutions non-dominées qui figurent dans le dernier front.

Une autre explication est liée au fait que les tournées de véhicules ne sont jamais remises en cause : les nouvelles requêtes sont insérées dans les tournées. On se focalise donc ainsi sur un sous-ensemble de l'espace de recherche.

Les résultats figurant dans la colonne "choix" montrent que l'indice calculé pour estimer le degré d'attraction des véhicules est pertinent. Ainsi très souvent il suffit de reconsidérer que les 2 ou 3 véhicules les plus attractifs pour une requête dynamique pour pouvoir l'insérer et réduire les valeurs des critères.

Cela est confirmé par le Tableau 5.3 qui présente la position moyenne selon le degré d'attraction de la tournée ayant accueillie une requête à insérer. Cette moyenne est calculée sur l'ensemble des instances ayant le même nombre de véhicules, sans distinction sur la

Nombre de véhicules	classement de la tournée choisie (moyenne)
2	1,064
4	2,048
6	2,674

TABLE 5.3 – Le classement moyen de la tournée choisie pour accueillir une requête selon le degré d'attraction

taille du problème. On voit que sur l'ensemble des instances possédant 2 véhicules, notre « degré d'attraction » a bien fonctionné car il a placé quasiment à chaque fois la bonne tournée en première position. Sur les instances à 4 véhicules, cela nous indique qu'il aurait suffi de parcourir les deux premières tournées pour insérer facilement la requête et obtenir la meilleure solution que nous avons trouvée en parcourant l'ensemble des tournées. Des conclusions similaires peuvent être tirées des instances avec 6 véhicules. Par conséquent, nous pouvons nous permettre de ne parcourir que 50% des tournées pour obtenir quasiment le même résultat qu'en parcourant 100% des tournées, ce qui permet de réduire le temps d'exécution de l'heuristique.

5.5.3 Comparaison de l'heuristique d'insertion HI avec NSGA-II

Nous allons maintenant comparer la qualité des solutions obtenues par l'heuristique HI à celles de la méthode NSGA-II développée dans le Chapitre 4 pour résoudre le problème de DARP statique. Ce test consiste à prendre les requêtes statiques et dynamiques à la fin de la journée (ou de l'horizon de planification) et à exécuter l'algorithme NSGA-II sur l'ensemble des requêtes pour générer un ensemble de solutions non-dominées. L'objectif est de voir de combien la solution proposée par l'heuristique HI est proche des solutions générées par NSGA-II. Donc pour NSGA-II les requêtes statiques et dynamiques sont traitées de la même manière et les violations des fenêtres de temps sont autorisées, ces violations entrainant une répercussion sur la qualité de service. En quelque sorte, le front calculé par NSGA-II est utilisé comme étant l'ensemble des optima de Pareto de référence si nous avions connu toutes les requêtes par avance.

Lors du calcul de la qualité de service, NSGA-II utilise les fenêtres de temps fournies par les passagers. Tandis que l'heuristique HI utilise, pour les requêtes statiques, des fenêtres corrigées. La correction de ces fenêtres est faite sur la base de la solution initiale sélectionnée pour servir les requêtes statiques.

Donc toute comparaison entre ces deux méthodes est à prendre avec précaution et pour la qualité de service cette comparaison est même impossible, étant donné que les fenêtres de temps conduisant aux pénalités ne sont pas identiques. Nous nous contentons donc de comparer les deux méthodes sur les deux autres critères restants.

Le Tableau 5.4 donne les résultats obtenus de la comparaison de l'heuristique HI et de l'heuristique NSGA-II sur les 30 catégories d'instances. Chaque ligne correspond aux résultats d'une catégorie d'instance. La colonne « Instance » représente le nom de la catégorie d'instance, la colonne « Véhicule » représente le nombre de véhicules, la colonne « %Reussi » représente le pourcentage moyen de réussite d'insertion des 10 requêtes dy-

namiques, la colonne « Temps NSGA-II » donne le temps d'exécution (en secondes) de la méthode de NSGA-II et la colonne « Temps HI » donne le temps d'exécution de l'heuristique d'insertion HI.

Les algorithmes NSGA-II et HI retournant un ensemble de solutions, nous avons simulé le comportement de l'opérateur de transport (comme indiqué au début de la section 5.5.1) pour n'en retenir à chaque fois qu'une. Les deux solutions retenues sont alors comparées en termes de coût total de transport et de CO_2 émis.

Les colonnes « Coût NSGA-II » et « Coût HI » représentent le coût de transport moyen des solutions retenues respectivement pour la méthode NSGA-II et l'heuristique HI et la colonne « %Coût Diff » représente le pourcentage de différence du coût moyen de transport obtenu par l'heuristique d'insertion par rapport à la méthode NSGA-II. Les colonnes « CO_2 NSGA-II » et « CO_2 RBS » représentent la quantité de CO_2 moyenne des solutions retenues respectivement pour NSGA-II et l'heuristique HI et la dernière colonne « $\%CO_2$ Diff. » représente le pourcentage de différence de la quantité de CO_2 moyenne obtenue par l'heuristique HI par rapport à la méthode NSGA-II.

Selon les résultats obtenus dans le Tableau 5.4, on voit que les valeurs des critères des solutions générées par NSGA-II sont en moyenne inférieure d'environ 10 % à celles calculées par l'heuristique d'insertion HI, sur les deux critères considérés.

5.5.4 Les résultats expérimentaux de la méthode arborescente

Les mêmes instances qui ont été utilisées pour tester l'heuristique par insertion HI vont être utilisées pour tester la méthode arborescente. La Figure 5.10 montre l'évolution du temps d'exécution en seconde de la méthode RBS en fonction de la taille du problème (nombre de requêtes statiques). Les temps d'exécution sont calculés sur la base de 10 requêtes dynamiques à insérer.

Cela s'applique par la règle de sélection des noeuds fils à explorer dans la RBS (cf section ? ? ? ? ?) : nous conservons tous les noeuds non dominés au sens de Pareto. Leur nombre devient important avec l'augmentation de la taille du problème.

De l'observation de la Figure 5.10 on constate que le temps d'exécution évolue de manière importante avec le nombre de requêtes statiques dans le problème. Par exemple le temps d'exécution dépasse 250 secondes pour les instances de plus 90 requêtes statiques et 10 requêtes dynamiques à insérer.

Taux de réussite et d'insertion

Les tests sur la méthode arborescente ont rapidement révélé que sur certaines instances, la RBS ne permet pas de trouver une solution réalisable. Ceci peut se produire à un niveau de l'arbre donné, lorsque tous les noeuds fils possibles sont coupés lors de la phase de filtrage. N'ayant plus de noeuds père pour générer le niveau suivant, l'algorithme s'arrête avec des solutions incomplètes. Nous rappelons que le filtre coupe un noeud dès lors que l'insertion d'un sommet dans une tournée viole les contraintes de capacité de véhicule ou conduit à ne plus respecter les fenêtres de temps des requêtes statiques.

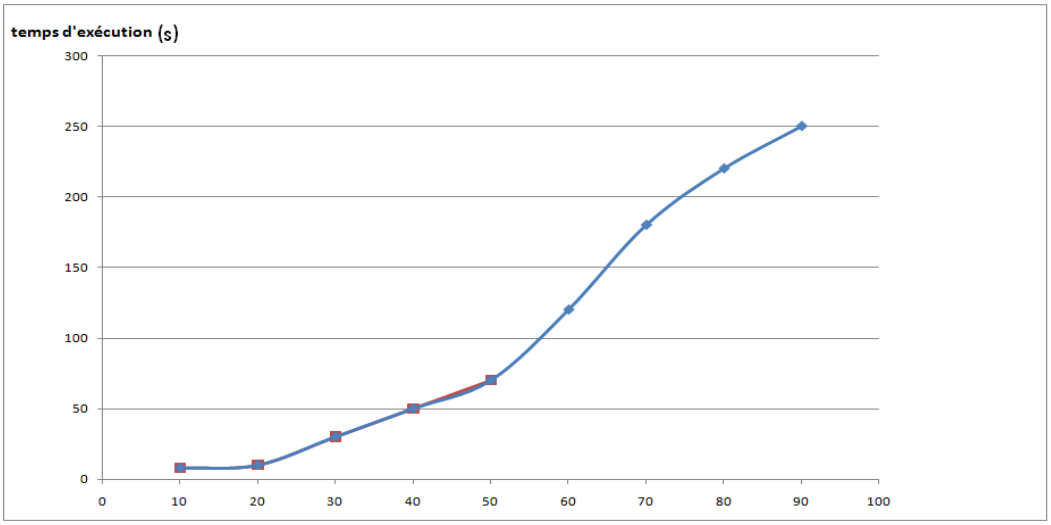


FIGURE 5.10 – Evolution du temps d'exécution de la RBS en fonction de la taille du problème

Le Tableau 5.5 présente le taux de réussite de l'algorithme pour chaque catégorie d'instance.

Les résultats présentés sont assez décevant car pour les instances de plus de 70 requêtes, la RBS s'avère inefficace pour résoudre le problème dynamique. Cela s'explique par les décisions qui sont prises en haut de l'arbre de recherche : des noeuds qui peuvent paraitre prometteurs ne conduisent finalement à aucune solution réalisable.

Pour remédier à cela, il faut envisager d'appliquer, sur des noeuds ne passant pas le filtre, une procédure de réparation. Ce travail fait partie des perspectives de recherche à court terme à l'issue de cette thèse.

Taille du front de Pareto

Pour avoir une idée sur le nombre de solutions non-dominées générées par RBS, nous calculons la taille du front de Pareto au dernier niveau de l'arbre (avec des solutions complètes) pour les instances qui pu être résolues. Les résultats obtenus dans le Tableau 5.6 montrent que la taille du front de Pareto est très faible. Pour cela il est préférable que lors de la sélection de garder l'ensemble des d'optima de Pareto générés à chaque niveau. Notons qu'il faut prendre ces résultats avec précautions, car certaines moyennes ne sont calculées qu'à partir d'une ou deux instances résolues sur les dix prévues initialement.

5.6 Conclusions

Dans ce chapitre nous avons abordé le problème de Transport à la Demande dynamique, deux méthodes ont été proposées : la première est une heuristique rapide qui permet de trouver des solutions réalisables, cette heuristique est basée sur un indicateur de difficulté

des requêtes et un indicateur de degré d'attraction. La structure des tournées courantes est conservée et l'heuristique tente seulement d'insérer les nouvelles requêtes tout en respectant les fenêtres de temps des requêtes déjà planifiées et la capacité des véhicules . Un ensemble de solutions non-dominées sont retenus à la fin du processus d'insertion. Les résultats obtenus avec cette heuristique sont prometteurs, notamment que le temps d'exécution est petit et que les solutions sont de bonne qualité.

La deuxième méthode proposée est une méthode arborescente tronquée RBS. Dans cette méthode la structure des tournées courantes est mise en cause lors de la planification de nouvelles requêtes. Les données du problème sont mises à jour en ajoutant les nouvelles requêtes ce qui donne un nouveau problème à résoudre. Ce nouveau problème est résolu en utilisant la RBS. Les résultats expérimentaux montrent que cette méthode n'arrive pas souvent à trouver des solutions réalisables, notamment pour les instances de grande taille. Donc un travail d'amélioration de cette méthode, à court terme, est nécessaire notamment au niveau du mécanisme de recovring, la sélection l'ordre d'insertion des tournées .....

Instance	Req.Stat.	Vehicule	%Reussi	Temps NSGA-II	Temps HI	Coût NSGA-II	Coût HI	%Coût Diff.	CO_2 NSGA-II	CO_2 HI	$\%CO_2$ Diff.
r-1	10	2	100	335,335	0,0797	193204	278645	44,2227	177034	230628	30,2729
r-2	10	4	100	382,327	0,0637	251813	285793	13,4939	218511	207014	-5,26144
r-3	10	6	100	403,808	0,0574	258010	308123	19,4229	179796	171803	-4,4455
r-4	20	2	100	455,058	0,1161	387175	356460	-7,9331	290539	309844	6,64448
r-5	20	4	100	469,429	0,1108	440679	472238	7,16141	340965	376349	10,3778
r-6	20	6	100	487,666	0,1062	432128	442629	2,42994	306820	297081	-3,17427
r-7	30	2	100	574,076	0,1963	452282	549818	21,5651	343055	426788	24,408
r-8	30	4	100	566,562	0,1554	518266	558354	7,73503	396003	444633	12,2801
r-9	30	6	100	579,464	0,1474	561216	551154	-1,79289	420860	411793	-2,15436
r-10	40	2	100	707,884	0,298	538719	609963	13,2248	403224	483097	19,8085
r-11	40	4	100	669,606	0,2182	614304	692832	12,7832	475384	546260	14,9092
r-12	40	6	100	670,284	0,2081	681169	681344	0,0257234	537127	545012	1,46812
r-13	50	2	100	832,235	0,4153	624257	696846	11,628	454110	529893	16,6883
r-14	50	4	100	765,431	0,2855	713604	759148	6,38217	555191	610142	9,89769
r-15	50	6	100	758,034	0,2503	784244	779844	-0,561102	611053	647081	5,89615
r-16	60	2	100	1020,25	0,5868	687698	791048	15,0285	507807	597894	17,7404
r-17	60	4	100	898,298	0,3513	779577	862694	10,6618	606195	697122	14,9996
r-18	60	6	100	864,769	0,3219	963231	934019	-3,03273	763191	758973	-0,552658
r-19	70	2	100	1140,01	0,7286	674563	807255	19,6708	481717	614093	27,4801
r-20	70	4	100	998,288	0,4778	790981	936318	18,3743	615378	749481	21,7918
r-21	70	6	100	933,56	0,3643	939480	913782	-2,73538	738567	758156	2,6523
r-22	80	2	100	1335,19	0,8121	675374	764500	13,1966	484876	571681	17,9026
r-23	80	4	100	1128,58	0,578	820516	866474	5,60118	633781	690406	8,93458
r-24	80	6	100	1033,51	0,4622	917152	881285	-3,91062	711985	726243	2,00264
r-25	90	2	100	1545,66	1,1942	720586	770589	6,93925	516457	596998	15,5948
r-26	90	4	100	1198,43	0,612	800706	875014	9,2804	626990	693239	10,5662
r-27	90	6	100	1114,56	0,5048	955722	854531	-10,5879	765976	732872	-4,32186
r-28	100	2	100	1749,61	1,0826	736390	845929	14,8751	535605	624765	16,6465
r-29	100	4	100	1323,3	0,7647	791795	890620	12,4811	616885	713350	15,6374
r-30	100	6	100	1193,41	0,6241	870960	868757	-0,252937	695934	726090	4,33313

TABLE 5.4 – Comparaison des algorithmes HI et NSGA-II

Instance	#Req.Stat.	Véhicule	% réussi.	%insert.	CPU	#Front
r-1	10	2	60	90,25	2,40733	2,33333
r-2	10	4	90	96	6,12367	3,44444
r-3	10	6	100	100	7,8458	4,8
r-4	20	2	10	45,7345	6,193	2
r-5	20	4	100	100	17,5968	2
r-6	20	6	100	100	26,5654	4,5
r-7	30	2	0	19,5127	-	0
r-8	30	4	50	77,0712	18,1394	5,4
r-9	30	6	100	100	40,5946	7
r-10	40	2	0	17	-	0
r-11	40	4	60	79,5	39,084	3,16667
r-12	40	6	100	100	59,3238	6,9
r-13	50	2	0	10,2654	-	0
r-14	50	4	30	68,6821	56,304	3,33333
r-15	50	6	90	96,4706	96,0646	3,77778
r-16	60	2	0	10,2271	-	0
r-17	60	4	10	50,3268	101,859	2
r-18	60	6	90	95,5	160,676	5,88889
r-19	70	2	0	10,25	-	0
r-20	70	4	10	50,625	72,125	4
r-21	70	6	20	61	164,236	2,5
r-22	80	2	0	8,37182	-	0
r-23	80	4	0	22,0299	-	0
r-24	80	6	30	61,1173	168,101	3,33333
r-25	90	2	0	8,66382	-	0
r-26	90	4	0	17,7319	-	0
r-27	90	6	10	51,1942	709,415	10
r-28	100	2	0	5,29265	-	0
r-29	100	4	0	17,2418	-	0
r-30	100	6	0	39,0361	-	0

TABLE 5.5 – Les résultats de la RBS pour t=0

Nombre de véhicules	nombre de solutions
2	2,02
4	4,16
6	5,31

TABLE 5.6 – Nombre moyen de solutions générées par la RBS

Conclusion

Ce travail de thèse a porté sur l'étude et la résolution d'un problème de Transport à la Demande multicritère. On distingue deux versions de ce problème : une version statique dans laquelle les données du problème sont connues à l'avance, ce problème est abordée souvent dans la littérature sous le nom de *static Dial a ride Problem* (DARP) et une version dynamique dans laquelle les requêtes peuvent arriver au fil du temps.

Le premier travail réalisé dans cette thèse est l'étude d'un problème de DARP statique multicritère. Trois critères qui peuvent être conflictuels ont été définis : le premier consiste à minimiser le coût de transport qui reflète l'intérêt du transporteur, le deuxième critère consiste à minimiser l'insatisfaction des passagers et enfin le troisième critère consiste à minimiser la quantité de CO_2 émise par l'ensemble des véhicules.

Le chapitre 1 présente une introduction au service de Transport à la demande. Nous avons présenté dans ce chapitre les différentes définitions données par le législateur ainsi que les caractéristiques et les enjeux de ce mode de transport. Quelques modes de configuration et de conception de ce mode de transport sont présentés dans ce même chapitre.

Dans le chapitre 2 nous avons présenté un état de l'art sur les problèmes de DARP. Nous avons présenté les travaux selon qu'ils abordent un problème statique ou un problème dynamique et selon le type d'objectif considéré : monocritère, multicritère avec une fonction d'agrégation ou multicritère avec optimisation simultanée de l'ensemble des critères. Cet état de l'art fait ressortir les différents travaux qui ont été consacrés aux problèmes de DARP multicritère et nous n'avons identifié que quatre travaux ayant abordé le problème en multicritère avec une optimisation simultanée de l'ensemble des critères. A notre connaissance aucun travail n'a considéré les trois critères que nous avons considéré dans le cadre de cette thèse.

Dans ce même chapitre nous avons présenté les deux problèmes abordés dans cette thèse. Le problème de DARP statique multicritère qui a été résolu dans le chapitre 4 et le problème de Transport à la Demande dynamique dont la méthode de résolution est présentée dans le chapitre 5.

Pour minimiser l'insatisfaction des passagers (ce qui revient à maximiser la qualité de service) un problème d'Optimal Timing a été étudié dans le chapitre 3. Une synthèse de l'état de l'art sur les problèmes de Timing est présentée dans ce chapitre. Le problème de Timing que nous avons défini dans cette thèse présente une similarité avec les problèmes

d'ordonnancement à une machine pour une séquence de travaux connue et une fonction de coût convexe. La fonction de coût (l'insatisfaction des passagers) que nous avons proposée de minimiser, pour chaque tournée, est constituée de la somme des retards et des avances aux sommets d'une tournées plus une grandeur liée au temps passé à bord du véhicule entre chaque paire de sommets successifs dans une tournée. Pour résoudre ce problème d'Optimal Timing nous avons adapté l'algorithme général proposé par Chrétienne et Sourd [Chrétienne et Sourd, 2003].

Dans le chapitre 4, nous avons proposé une méthode évolutionnaire pour chercher un ensemble d'optima de Pareto. Quelques tests préliminaires ont été effectués pour faire une énumération exhaustive des optima de Pareto sur des instances de petite taille en utilisant une méthode *force brute*, les résultats obtenus montrent que la taille du front de Pareto augmente considérablement avec la taille du problème. Il s'avère donc utopique de vouloir calculer l'intégralité du front de Pareto. C'est pour cela que nous avons choisi d'utiliser l'algorithme NSGA-II qui intègre naturellement une méthode de diversification qui permet de faire un échantillonnage du front de Pareto en se basant sur une distance de *Crowding*.

La mise en oeuvre des différents mécanismes de cet algorithme a été présentée en détail dans ce chapitre 4 ainsi que les trois heuristiques de construction qui ont été utilisées pour la création de la population initiale dans NSGA-II. La première est une *heuristique de liste* qui a vocation à produire de bonnes solutions relativement au critère d'insatisfaction des passagers. La deuxième est une *heuristique de balayage* destinée à minimiser la distance parcourue, ce qui influence directement les critères de coût et de quantité émise de CO_2. La troisième est une *heuristique de Clustering* qui prend en considération la proximité géographique et temporelle des requêtes lors de la construction des tournées. Le but est de produire des solutions "équilibrées" entre le critère de l'insatisfaction des passagers d'une part et le coût de transport et la quantité émise de CO_2 d'autre part.

L'algorithme NSGA-II a été ensuite calibré. Durant ce calibrage des comportements particuliers ont été constatés, en effet durant l'exécution de NSGA-II certaines solutions non-dominées disparaissent. Ceci est dû à la taille de la population qui est limitée et à l'application de la distance de *Crowding* pour choisir le sous-ensemble de solutions qui constituera la population suivante. Ceci peut conduire à éliminer un optima de Pareto pour éventuellement générer plus tard une solution moins bonne. Ce comportement met en cause le principe d'élitisme qui caractérise l'algorithme NSGA-II. Plusieurs tests ont été menés sur des instances aléatoires. Nous nous sommes également comparés à des méthodes issues de la littérature en adaptant les problèmes sur lesquels elles s'appliquent. Même si NSGA-II est dominé au niveau du coût de transport par la méthode Tabou de Cordeau et Laporte, il a l'avantage de proposer dans un temps de calcul équivalent plusieurs dizaines de solutions là où la méthode Tabou n'en propose qu'une. Il est à noter également que le temps moyen passé à bord des véhicules est plus faible dans les solutions proposées par NSGA-II.

Dans le chapitre 5, nous avons étudié le problème de Transport à la Demande dynamique, dans lequel de nouvelles requêtes à traiter arrivent en cours de journée. Globalement, les caractéristiques des données ainsi que les fonctions objectifs considérées dans le pro-

blème statique sont maintenues dans ce problème. La différence majeure avec le problème statique réside dans les fenêtres de temps associées aux requêtes. Il y a en effet deux types de requêtes : les anciennes requêtes, qui ont déjà été planifiées et pour lesquelles un engagement horaire a été pris auprès des usagers et de nouvelles requêtes. Pour ces nouvelles requêtes, aucun engagement n'est pris, on peut donc violer leur fenêtres de temps, ce qui n'est pas le cas pour les anciennes requêtes. Les nouvelles requêtes ont donc des fenêtres souples et les anciennes des fenêtres dures.

Pour résoudre ce problème nous avons proposé deux méthodes. La première est une heuristique rapide qui consiste à insérer les nouvelles requêtes dans l'ordre de leur degré de difficulté. Cette heuristique ne met pas en cause la structure des tournées courantes, elle tente seulement d'insérer chaque nouvelle requête dans les tournées existantes. Les tests expérimentaux qui ont été effectués sur des instances aléatoires ont montré l'efficacité de cette heuristique à trouver en quelques secondes des solutions de bonne qualité. La deuxième méthode qui a été proposée pour résoudre le problème de Transport à la Demande dynamique consiste à remettre en cause les structures des tournées courantes. L'ensemble des nouvelles requêtes et des anciennes requêtes qui restent à servir forment un nouveau problème pour la résolution duquel nous avons proposé une méthode arborescente tronquée. La méthode développée est connue sous le nom de *Recovering Beam Search*. Les résultats obtenus ont montré les limites de cette méthode, dans sa formulation actuelle, à produire des solutions réalisables.

Les travaux menés dans cette thèse sont des travaux qui doivent être poursuivis. Pour le problème de DARP statique multicritère des améliorations doivent être ajoutées afin de remédier au problème de cycle répétitif de production et de disparition des optima de Pareto.

Au niveau du problème de transport à la demande dynamique des améliorations doivent être appliquées sur la méthode RBS pour augmenter son efficacité à produire des solutions réalisables. La première piste à suivre serait de travailler sur l'ordre d'insertion des requêtes en privilégiant les requêtes à fenêtres dures par rapport aux autres. Une autre idée serait d'ajouter un processus de réparation sur les solutions partielles non réalisables. Processus qui pourrait venir en support au mécanisme de *recovering*.

Annexes

Annexe A

Annexes

A.1 Annexes

indicateurs de qualité / Heuristiques	Q1	Q2	Q3	CPU(s)	#Front
H1	12%	50%	0,94	0,070	12
H2	16%	80%	1,00	0,101	10
H3	72%	100%	0,38	0,575	36

TABLE A.1 – Les résultats de la comparaison des trois heuristiques H1, H2 et H3 sur une instance de 40 requêtes et 5 véhicules

indicateurs de qualité / Heuristiques	Q1	Q2	Q3	CPU(s)	#Front
H1	42%	100%	1,00	0,089	10
H2	21%	100%	0,86	0,171	6
H3	38%	75%	0,657	0,915	18

TABLE A.2 – Les résultats de la comparaison des trois heuristiques H1, H2 et H3 sur une instance de 80 requêtes et 5 véhicules

indicateurs de qualité / Heuristiques	Q1	Q2	Q3	CPU(s)	#Front
H1	12%	100%	1,00	0,124	6
H2	24%	100%	0,96	0,329	12
H3	64%	78%	0,38	0,840	72

TABLE A.3 – Les résultats de la comparaison des trois heuristiques H1, H2 et H3 sur une instance de 100 requêtes et 5 véhicules

A.2 les résultats des expérimentations de l'heuristique par insertion

Instance	#Req.Stat.	#Req.Dyn.	Vehicule	CPU	#Front	Choix
r-1	10	9	2	0,0427	2,2	1
r-2	10	9	4	0,0482	2,7	1,1
r-3	10	9	6	0,0446	2,4	1,2
r-4	20	9	2	0,0884444	1,33333	1
r-5	20	9	4	0,0901	4,1	1,8
r-6	20	9	6	0,0864	4,3	2,2
r-7	30	9	2	0,1538	1,7	1
r-8	30	9	4	0,1361	4,5	2,3
r-9	30	9	6	0,1189	5,1	2,6
r-10	40	9	2	0,2386	1,9	1
r-11	40	9	4	0,1864	4,8	2,4
r-12	40	9	6	0,1698	5	3,3
r-13	50	9	2	0,3443	2,5	1
r-14	50	9	4	0,2341	3,6	2,4
r-15	50	9	6	0,2272	4	2,9
r-16	60	9	2	0,4976	2,5	1
r-17	60	9	4	0,2933	2,8	2,2
r-18	60	9	6	0,2737	3,5	3
r-19	70	9	2	0,627	2,6	1
r-20	70	9	4	0,4064	3	2,1
r-21	70	9	6	0,318	4,4	3,3
r-22	80	9	2	0,7257	1,8	1
r-23	80	9	4	0,4896	3	2,3
r-24	80	9	6	0,408	3,9	2,8
r-25	90	9	2	1,0275	1,6	1
r-26	90	9	4	0,525	3,3	2
r-27	90	9	6	0,4448	4,1	2,6
r-28	100	9	2	0,9667	2,1	1
r-29	100	9	4	0,6871	2,4	1,9
r-30	100	9	6	0,523	3,5	2,9

TABLE A.4 – Les résultats de l'insertion pour t=200

Instance	Coût Min.	Coût Moy.,	Coût Max.	QoS Min.	QoS Moy,	QoS Max.	CO_2 Min.	CO_2 Moy.	CO_2 Max.
r-1	155657	289616	362552	201080	1,24E+07	8,83E+07	87503	232513	316591
r-2	209264	282583	392798	13600	5,90E+06	3,19E+07	140835	201683	281466
r-3	266277	294675	330317	6183,18	8,01E+06	2,57E+07	133138	162405	232065
r-4	195572	373995	457163	6,54E+06	4,48E+07	8,85E+07	97786,1	321730	387148
r-5	411830	459907	508285	18419,5	7,76E+06	2,13E+07	316140	360933	432171
r-6	322215	441638	531016	580568	7,25E+06	3,07E+07	241886	291528	351352
r-7	447210	532612	581962	1,57E+07	6,92E+07	1,37E+08	338439	411684	457077
r-8	429408	545522	619965	407499	6,81E+06	3,07E+07	378320	430406	474596
r-9	442986	524934	572111	282009	5,26E+06	2,48E+07	324245	397996	460294
r-10	394021	606644	664783	2,25E+07	9,00E+07	1,74E+08	413722	475022	522901
r-11	552940	665909	727132	1,17E+06	8,46E+06	2,57E+07	421951	521922	568166
r-12	580474	661681	748679	214638	5,67E+06	3,25E+07	421705	525766	622510
r-13	640352	683416	750792	6,82E+07	1,06E+08	1,64E+08	490342	515497	553906
r-14	514417	728986	870690	748318	8,82E+06	2,57E+07	436759	584432	698647
r-15	495821	744202	918427	252505	4,04E+06	1,85E+07	495901	625905	713880
r-16	715854	765751	831337	7,49E+07	1,35E+08	2,28E+08	534778	579214	629462
r-17	661831	832933	1,05E+06	2,00E+06	1,74E+07	5,53E+07	540345	676051	839172
r-18	826636	916062	983403	249429	4,94E+06	2,07E+07	654179	737447	922402
r-19	744447	791524	876273	8,63E+07	1,76E+08	3,00E+08	536950	602151	751284
r-20	841419	917499	1,06E+06	1,59E+06	2,15E+07	6,05E+07	658642	732916	844546
r-21	701964	893736	999338	20694,9	7,01E+06	2,50E+07	663112	723712	813957
r-22	687460	749207	842106	1,09E+08	1,79E+08	2,56E+08	509063	561436	642958
r-23	721646	854803	1,10E+06	1,90E+06	1,80E+07	4,75E+07	585149	681768	872433
r-24	699698	853171	991818	157172	4,26E+06	1,49E+07	596925	701165	784443
r-25	584470	748668	847422	1,41E+08	2,30E+08	2,86E+08	530833	585107	630547
r-26	685268	894837	1,06E+06	1,08E+07	3,62E+07	7,33E+07	558254	704874	815554
r-27	614453	835974	1,10E+06	681125	8,20E+06	2,04E+07	578036	709815	875322
r-28	706141	831748	974385	1,58E+08	2,40E+08	4,12E+08	506797	610469	743855
r-29	657022	870684	1,04E+06	1,16E+07	4,59E+07	1,64E+08	566023	689515	807003
r-30	550054	875941	1,02E+06	333789	6,60E+06	2,85E+07	460662	729329	838638

TABLE A.5 – Les valeurs des trois critères (Min, Moy, Max) (insertion à t=200)

Instance	#Req.Stat.	#Req.Dyn.	Vehicule	CPU	#Front	Choix
r-1	10	7	2	0,0354	2,2	1
r-2	10	7	4	0,0414	2,8	1,1
r-3	10	7	6	0,0436	2	1,2
r-4	20	6,9	2	0,0775556	1,44444	1
r-5	20	6,9	4	0,0733	3,7	1,7
r-6	20	6,9	6	0,0729	4,1	2,2
r-7	30	7	2	0,1349	1,9	1
r-8	30	7	4	0,1137	3,8	2
r-9	30	7	6	0,1112	4,4	2,9
r-10	40	7	2	0,2085	1,5	1,1
r-11	40	7	4	0,1679	3,7	2,3
r-12	40	7	6	0,1521	4,4	3,3
r-13	50	7	2	0,2869	2,6	1
r-14	50	7	4	0,2182	2,9	2,1
r-15	50	7	6	0,1966	3,8	2,9
r-16	60	7	2	0,3909	2,4	1,1
r-17	60	7	4	0,2624	2,9	2,2
r-18	60	7	6	0,2488	2,8	3,3
r-19	70	7	2	0,4972	2,4	1
r-20	70	7	4	0,3423	2,8	2
r-21	70	7	6	0,2798	3,2	3,3
r-22	80	7	2	0,591	1,8	1
r-23	80	7	4	0,4019	2,3	2,3
r-24	80	7	6	0,3501	3,8	3,3
r-25	90	7	2	0,8282	1,8	1
r-26	90	7	4	0,4457	3,2	2
r-27	90	7	6	0,386	3,1	2,6
r-28	100	7	2	0,8288	1,9	1
r-29	100	7	4	0,5698	2,3	2
r-30	100	7	6	0,4554	3,6	2,8

TABLE A.6 – Les résultats de l'insertion pour t=400

Instance	Coût Min.	Coût Moy.,	Coût Max.	QoS Min.	QoS Moy,	QoS Max.	CO_2 Min.	CO_2 Moy.	CO_2 Max.
r-1	108133	251725	313555	103160	1,24E+07	4,07E+07	69409,1	205643	245681
r-2	185231	264552	333432	170961	7,77E+06	1,84E+07	150132	184650	232755
r-3	230722	264814	289319	6246,78	7,17E+06	1,84E+07	115361	143684	195393
r-4	165443	339082	438877	3,51E+06	2,78E+07	7,40E+07	82721,7	297001	363907
r-5	369955	441452	498399	252849	6,33E+06	1,62E+07	303102	347783	397250
r-6	297076	422413	482420	130613	6,98E+06	4,60E+07	221095	280203	336127
r-7	413428	491844	566406	7,09E+06	4,23E+07	6,93E+07	311857	377749	452393
r-8	353720	498082	564567	209582	3,60E+06	2,82E+07	311941	385765	426609
r-9	433077	488355	545133	53270,9	5,73E+06	3,13E+07	312180	363424	448900
r-10	379729	568706	646732	2,26E+07	6,92E+07	1,53E+08	398715	443562	503947
r-11	530880	613218	678583	1,51E+06	8,32E+06	2,24E+07	418178	482455	529321
r-12	568790	634862	715094	159497	5,38E+06	1,88E+07	409264	503794	594289
r-13	602475	649122	696384	3,60E+07	7,02E+07	9,16E+07	460228	489739	529194
r-14	462451	701497	824486	371976	8,52E+06	3,22E+07	387715	557544	652242
r-15	491966	729930	862066	300695	4,14E+06	1,83E+07	487350	603511	680910
r-16	674027	733897	834629	4,15E+07	8,98E+07	1,67E+08	502841	550923	626510
r-17	652049	809341	1,03E+06	1,60E+06	1,74E+07	5,54E+07	512556	645016	823272
r-18	803540	874315	954696	222404	2,82E+06	9,75E+06	620562	703435	889053
r-19	713458	764661	847080	5,09E+07	1,19E+08	2,33E+08	525887	579443	720631
r-20	782377	869814	1,03E+06	576129	1,80E+07	3,67E+07	613339	692241	811794
r-21	656260	874951	975386	412120	7,82E+06	2,47E+07	625183	713474	773103
r-22	675155	725335	825393	8,50E+07	1,29E+08	1,92E+08	487496	539870	625619
r-23	690536	817541	1,08E+06	2,55E+06	1,95E+07	4,75E+07	562576	650697	854177
r-24	664534	831517	950694	167571	5,44E+06	3,80E+07	573205	681276	759429
r-25	541930	741271	814636	1,03E+08	1,73E+08	2,77E+08	512686	563739	612090
r-26	650003	843121	1,02E+06	9,26E+06	3,53E+07	9,90E+07	528113	665575	781140
r-27	583088	743910	1,06E+06	562250	9,41E+06	3,72E+07	548507	644830	818070
r-28	688339	796896	930434	8,25E+07	1,60E+08	2,74E+08	494194	585488	707013
r-29	624397	838330	993520	1,15E+07	3,22E+07	8,53E+07	540939	665908	775360
r-30	522058	830415	963842	392202	6,88E+06	2,08E+07	430555	692727	820598

TABLE A.7 – Les valeurs des trois critères (Min, Moy, Max) (insertion à t=400)

A.3 les résultats des expérimentations de l'heuristique par insertion

Instance	#Req.Stat.	#Req.Dyn.	Vehicule	CPU	#Front	Choix
r-1	10	4,8	2	0,0346	1,8	1
r-2	10	4,8	4	0,0378	2,6	1,2
r-3	10	4,8	6	0,0375	2,1	1,2
r-4	20	4,8	2	0,0735	1,75	1
r-5	20	4,8	4	0,0659	2,7	1,7
r-6	20	4,8	6	0,071	2,4	1,7
r-7	30	5	2	0,1126	1,8	1
r-8	30	5	4	0,1006	3,7	2,3
r-9	30	5	6	0,1039	3,7	2,1
r-10	40	5	2	0,1633	1,5	1
r-11	40	5	4	0,1436	3,4	2,5
r-12	40	5	6	0,1344	3,8	3,2
r-13	50	5	2	0,2387	2,4	1
r-14	50	5	4	0,1859	2,6	2,4
r-15	50	5	6	0,1736	3,4	2,8
r-16	60	5	2	0,3275	2,2	1,2
r-17	60	5	4	0,2271	2,6	2,3
r-18	60	5	6	0,2124	2,1	3
r-19	70	5	2	0,3944	2,3	1,2
r-20	70	5	4	0,3016	2,8	2,3
r-21	70	5	6	0,2555	3,5	3,4
r-22	80	5	2	0,4887	1,9	1,2
r-23	80	5	4	0,3444	2,2	2,2
r-24	80	5	6	0,3069	3,4	3,4
r-25	90	5	2	0,6429	1,8	1,1
r-26	90	5	4	0,3861	3	2
r-27	90	5	6	0,3373	3,3	2,6
r-28	100	5	2	0,7078	2,3	1
r-29	100	5	4	0,4727	2,3	1,9
r-30	100	5	6	0,4106	3,1	3

TABLE A.8 – Les résultats de l'insertion pour t=600

Instance	Coût Min.	Coût Moy.,	Coût Max.	QoS Min.	QoS Moy,	QoS Max.	CO_2 Min.	CO_2 Moy.	CO_2 Max.
r-1	81566,9	222572	273207	16621,4	8,43E+06	2,35E+07	49355,3	181871	210051
r-2	149863	239804	334495	79315,1	5,20E+06	1,42E+07	112813	165184	227250
r-3	193372	238507	270720	117378	5,55E+06	1,38E+07	96686	138441	175061
r-4	219762	337412	412410	1,18E+06	2,53E+07	6,24E+07	230751	297944	336666
r-5	331605	412071	451236	35980,4	8,47E+06	3,39E+07	272315	324308	362013
r-6	257393	390285	456441	502,43	6,49E+06	3,01E+07	189692	249523	308631
r-7	396863	470873	519002	1,77E+06	2,14E+07	5,90E+07	299342	358132	398467
r-8	338525	464520	536270	20464	1,53E+06	1,12E+07	298917	361320	420840
r-9	403078	466377	517863	40542,3	948923	6,36E+06	302929	335972	426420
r-10	356889	540203	598298	9,88E+06	3,48E+07	7,88E+07	374039	421705	472120
r-11	469453	589480	645114	36077,8	4,34E+06	1,93E+07	359405	460224	496153
r-12	537165	620025	699959	22,8759	2,27E+06	1,48E+07	394671	493348	588762
r-13	582390	625184	666346	2,31E+07	4,43E+07	7,91E+07	439030	466808	497929
r-14	439257	680395	784559	369990	7,59E+06	2,56E+07	365324	542236	622095
r-15	442448	704175	851423	200447	2,38E+06	7,13E+06	428017	588862	667891
r-16	656686	713470	792628	2,68E+07	6,65E+07	1,11E+08	489332	531947	594307
r-17	620376	779709	998641	742617	1,09E+07	3,30E+07	497375	623224	795886
r-18	769098	862425	939402	14511,4	1,80E+06	5,99E+06	630412	710766	873855
r-19	678028	724944	806159	2,52E+07	6,96E+07	1,50E+08	497195	539487	677664
r-20	743812	824399	990919	346182	1,21E+07	3,30E+07	577302	649077	769859
r-21	636467	836984	943466	6099,69	4,51E+06	2,12E+07	608912	683269	739678
r-22	640188	694157	788004	4,33E+07	7,91E+07	1,56E+08	463920	514831	597371
r-23	642190	780217	1,05E+06	1,89E+06	1,20E+07	4,59E+07	517022	619468	832242
r-24	647369	817950	930740	128733	3,57E+06	1,47E+07	549780	665505	744778
r-25	517073	703059	761038	7,89E+07	1,08E+08	2,12E+08	488079	532424	567068
r-26	619560	798709	984146	1,91E+06	3,12E+07	6,75E+07	500762	627959	763837
r-27	560392	796309	1,03E+06	121114	1,09E+07	3,52E+07	521272	659986	805606
r-28	656627	758076	897456	3,36E+07	9,74E+07	1,55E+08	471157	555370	673415
r-29	564214	803742	948278	1,57E+06	1,89E+07	5,70E+07	489681	639172	752155
r-30	498380	806788	927675	1381,31	2,88E+06	1,90E+07	412302	671229	779975

TABLE A.9 – Les valeurs des trois critères (Min, Moy, Max) (insertion à t=600)

Instance	#Req.Stat.	#Req.Dyn.	Vehicule	CPU	#Front	Choix
r-1	10	2,7	2	0,0295	1,5	1,1
r-2	10	2,7	4	0,0383	1,8	1,1
r-3	10	2,7	6	0,0418	1,7	1,2
r-4	20	2,6	2	0,0672857	1,14286	1,14286
r-5	20	2,6	4	0,0635	2,3	1,7
r-6	20	2,6	6	0,0653	2,5	1,6
r-7	30	2,9	2	0,118	1,7	1
r-8	30	2,9	4	0,0983	2,8	2,2
r-9	30	2,9	6	0,0931	2,7	2,7
r-10	40	3	2	0,1427	1,8	1,2
r-11	40	3	4	0,1336	3,1	2,3
r-12	40	3	6	0,1328	3,1	2,9
r-13	50	3	2	0,1989	2,6	1,1
r-14	50	3	4	0,173	2,5	2,6
r-15	50	3	6	0,1602	3,2	2,7
r-16	60	3	2	0,2599	2,4	1,6
r-17	60	3	4	0,2154	2,9	2,2
r-18	60	3	6	0,2072	1,9	3,2
r-19	70	3	2	0,3162	2,6	1,3
r-20	70	3	4	0,2521	3,3	2,5
r-21	70	3	6	0,2361	3,7	3,6
r-22	80	3	2	0,3949	1,6	1,4
r-23	80	3	4	0,2973	2,6	2,5
r-24	80	3	6	0,2724	2,9	3,3
r-25	90	3	2	0,5068	1,8	1,2
r-26	90	3	4	0,3388	2,9	2,3
r-27	90	3	6	0,3101	2,7	2,5
r-28	100	3	2	0,5787	2	1,3
r-29	100	3	4	0,4055	2,4	2,1
r-30	100	3	6	0,3621	3,4	3

TABLE A.10 – Les résultats de l'insertion pour t=600

Instance	Coût Min.	Coût Moy.,	Coût Max.	QoS Min.	QoS Moy,	QoS Max.	CO_2 Min.	CO_2 Moy.	CO_2 Max.
r-1	55302,5	187753	264117	0	1,84E+06	6,07E+06	27651,2	155226	227037
r-2	117771	200137	295570	803,386	1,61E+06	5,62E+06	86844,2	140076	207216
r-3	183284	211861	246916	803,386	735396	2,51E+06	91642	124291	160255
r-4	215080	315349	393383	865196	9,40E+06	2,38E+07	225834	273495	296515
r-5	293405	380185	431820	28962,5	4,97E+06	1,47E+07	242411	295657	351996
r-6	232146	374987	425892	0	2,90E+06	1,25E+07	179715	243691	279743
r-7	400328	435038	487252	1,25E+06	6,91E+06	2,10E+07	299954	333172	380342
r-8	332789	451395	512434	19421,9	2,16E+06	9,68E+06	290019	346569	414542
r-9	387065	445505	486113	44478,3	1,81E+06	1,28E+07	270224	317295	389848
r-10	319943	512326	559358	3,77E+06	1,01E+07	1,90E+07	335940	394694	427141
r-11	454402	561171	610279	159,987	1,51E+06	7,81E+06	346844	433594	479678
r-12	493074	592448	694579	12491,5	960750	5,12E+06	372036	468407	583150
r-13	556390	602302	644887	1,23E+07	2,63E+07	5,25E+07	406903	447444	493387
r-14	419872	641735	717230	2274,43	4,59E+06	2,54E+07	346388	509786	578805
r-15	422601	697622	812345	507,682	1,76E+06	7,78E+06	406292	563982	635222
r-16	628752	677697	756145	2,10E+06	2,98E+07	7,44E+07	468366	500861	565912
r-17	589425	753418	975135	4980,7	4,61E+06	3,28E+07	458690	597608	764615
r-18	724013	822539	893306	35,8329	426722	3,59E+06	618832	671832	824784
r-19	638643	694040	762352	1,51E+07	4,22E+07	1,06E+08	459416	508272	618398
r-20	713112	780797	959546	354976	1,03E+07	6,66E+07	562046	612190	742739
r-21	620248	805389	913978	0,00573567	3,88E+06	2,55E+07	583058	651977	720407
r-22	606622	666015	751158	1,59E+07	4,02E+07	8,29E+07	441482	486056	571449
r-23	603121	754126	1,00E+06	132090	4,63E+06	1,38E+07	475999	595803	799462
r-24	623890	783383	911840	186,446	3,50E+06	1,38E+07	531506	631339	692911
r-25	492588	671436	724543	1,59E+07	5,16E+07	7,66E+07	459193	503264	542999
r-26	600201	785471	954907	557515	8,69E+06	2,44E+07	482044	614464	744382
r-27	525477	745769	1,01E+06	88503,2	3,41E+06	8,76E+06	479130	620447	785131
r-28	624503	727666	853088	1,87E+07	5,19E+07	8,47E+07	449520	531138	648588
r-29	533268	761217	886111	178262	8,66E+06	2,37E+07	457187	605447	701231
r-30	475334	771087	873366	137,941	3,11E+06	1,94E+07	384646	642568	761649

TABLE A.11 – Les valeurs des trois critères (Min, Moy, Max) (insertion à t=800)

Instance	#Req.Stat.	Véhicule	% réussi.	%insert.	CPU	#Front
r-1	10	2	60	90,25	2,40733	2,33333
r-2	10	4	90	96	6,12367	3,44444
r-3	10	6	100	100	7,8458	4,8
r-4	20	2	10	45,7345	6,193	2
r-5	20	4	100	100	17,5968	2
r-6	20	6	100	100	26,5654	4,5
r-7	30	2	0	19,5127	-	0
r-8	30	4	50	77,0712	18,1394	5,4
r-9	30	6	100	100	40,5946	7
r-10	40	2	0	17	-	0
r-11	40	4	60	79,5	39,084	3,16667
r-12	40	6	100	100	59,3238	6,9
r-13	50	2	0	10,2654	-	0
r-14	50	4	30	68,6821	56,304	3,33333
r-15	50	6	90	96,4706	96,0646	3,77778
r-16	60	2	0	10,2271	-	0
r-17	60	4	10	50,3268	101,859	2
r-18	60	6	90	95,5	160,676	5,88889
r-19	70	2	0	10,25	-	0
r-20	70	4	10	50,625	72,125	4
r-21	70	6	20	61	164,236	2,5
r-22	80	2	0	8,37182	-	0
r-23	80	4	0	22,0299	-	0
r-24	80	6	30	61,1173	168,101	3,33333
r-25	90	2	0	8,66382	-	0
r-26	90	4	0	17,7319	-	0
r-27	90	6	10	51,1942	709,415	10
r-28	100	2	0	5,29265	-	0
r-29	100	4	0	17,2418	-	0
r-30	100	6	0	39,0361	-	0

TABLE A.12 – Les résultats de RBS pour t=0

Req	Coût Min.	Coût Moy.,	Coût Max.	QoS Min.	QoS Moy,	QoS Max.	CO_2 Min.	CO_2 Moy.	CO_2 Max.
r-1	233396	276105	322689	17920,1	646697	3,61E+06	167059	195047	220541
r-2	218455	262747	307522	90845,4	2,41E+06	1,45E+07	157706	187803	238635
r-3	230808	285382	351655	0	368513	3,62E+06	131174	180737	236211
r-4	356460	366271	376082	1,95E+06	2,07E+06	2,19E+06	278598	283074	287550
r-5	326387	379482	441102	6,84764	1,02E+06	3,95E+06	232973	286617	336004
r-6	338468	402954	472918	129,988	1,87E+06	7,57E+06	211301	279286	331968
r-7	-	-	-	-	-	-	-	-	-
r-8	527999	586302	671914	497005	3,38E+06	9,36E+06	405687	448839	514377
r-9	482484	558444	633699	38660,2	852961	6,76E+06	391022	432377	503652
r-10	-	-	-	-	-	-	-	-	-
r-11	649160	719124	768755	164594	4,50E+06	1,57E+07	516261	573750	615562
r-12	627320	727212	819878	20833,2	1,86E+06	1,32E+07	475937	572499	666713
r-13	-	-	-	-	-	-	-	-	-
r-14	771871	835054	904645	197512	1,87E+06	6,10E+06	614945	669194	738519
r-15	779421	854424	923536	111016	1,32E+06	5,99E+06	591401	680307	736992
r-16	-	-	-	-	-	-	-	-	-
r-17	957612	960533	963454	3,05E+06	3,05E+06	3,05E+06	758529	760188	761847
r-18	904584	1,03E+06	1,11E+06	238497	1,24E+06	2,09E+06	718737	812810	874113
r-19	-	-	-	-	-	-	-	-	-
r-20	996094	1,01E+06	1,02E+06	2,03E+06	2,03E+06	2,04E+06	796715	805677	815753
r-21	1,03E+06	1,07E+06	1,09E+06	1,78E+06	3,06E+06	4,10E+06	837047	846725	853784
r-22	-	-	-	-	-	-	-	-	-
r-23	-	-	-	-	-	-	-	-	-
r-24	1,09E+06	1,16E+06	1,21E+06	1,72E+06	2,95E+06	6,74E+06	863158	917912	946197
r-25	-	-	-	-	-	-	-	-	-
r-26	-	-	-	-	-	-	-	-	-
r-27	1,25E+06	1,26E+06	1,27E+06	9,93E+06	1,01E+07	1,06E+07	993709	1,00E+06	1,01E+06
r-28	-	-	-	-	-	-	-	-	-
r-29	-	-	-	-	-	-	-	-	-
r-30	-	-	-	-	-	-	-	-	-

TABLE A.13 – Les valeurs des trois critères (Min, Moy, Max) obtenues par RBS (insertion à t=0)

Instance	#Req.Stat.	#Req.Dyn.	Véhicule	% réussi.	%insert.	CPU	#Front
r-1	10	9	2	50	74,4543	1,3758	2,2
r-2	10	9	4	100	100	3,0159	4,9
r-3	10	9	6	100	100	4,4073	6,2
r-4	20	9	2	0	30,2116	-	0
r-5	20	9	4	90	90,2041	9,901	3,11111
r-6	20	9	6	100	100	14,2298	4,8
r-7	30	9	2	0	19,5062	-	0
r-8	30	9	4	60	72,9617	15,6573	6,66667
r-9	30	9	6	100	100	20,0452	7
r-10	40	9	2	0	14,1983	-	0
r-11	40	9	4	50	69,8082	25,955	3,4
r-12	40	9	6	100	100	33,2386	5,7
r-13	50	9	2	0	6,62743	-	0
r-14	50	9	4	40	63,0357	30,3483	3,75
r-15	50	9	6	70	77,8846	58,8753	6,14286
r-16	60	9	2	0	4,46598	-	0
r-17	60	9	4	20	43,309	68,7575	5
r-18	60	9	6	90	94,4737	110,738	5,11111
r-19	70	9	2	0	4,94935	-	0
r-20	70	9	4	0	22,9744	-	0
r-21	70	9	6	20	35,8226	96,774	4,5
r-22	80	9	2	0	6,17276	-	0
r-23	80	9	4	0	21,0037	-	0
r-24	80	9	6	30	37,605	118,146	2
r-25	90	9	2	0	2,56046	-	0
r-26	90	9	4	0	16,9323	-	0
r-27	90	9	6	20	48,9702	132,281	8,5
r-28	100	9	2	0	4,96431	-	0
r-29	100	9	4	0	9,97431	-	0
r-30	100	9	6	10	23,342	128,283	2

TABLE A.14 – Les résultats de RBS pour t=200

Req	Coût Min.	Coût Moy.,	Coût Max.	QoS Min.	QoS Moy,	QoS Max.	CO_2 Min.	CO_2 Moy.	CO_2 Max.
r-1	244145	278786	310664	478695	3,58E+06	8,37E+06	172145	206838	237995
r-2	221153	283087	366100	1590,26	1,82E+06	1,10E+07	160559	219171	282435
r-3	182830	275119	339331	159,428	1,86E+06	1,17E+07	110847	171033	249034
r-4	-	-	-	-	-	-	-	-	-
r-5	335544	392423	457173	31994,3	847672	2,68E+06	238991	306615	347355
r-6	335419	404834	524196	864,498	1,42E+06	9,20E+06	241931	294316	392344
r-7	-	-	-	-	-	-	-	-	-
r-8	511006	547853	642513	78388,9	793503	3,30E+06	394854	430195	499598
r-9	490288	538953	589001	95759,6	742047	4,98E+06	338981	407368	475941
r-10	-	-	-	-	-	-	-	-	-
r-11	639482	670737	731979	145648	2,38E+06	5,68E+06	490473	518767	573529
r-12	623382	693055	747846	15280,9	2,09E+06	1,04E+07	473697	537713	582304
r-13	-	-	-	-	-	-	-	-	-
r-14	763982	816603	875144	1,06E+06	2,70E+06	7,90E+06	619455	651049	688506
r-15	723509	822996	885539	12600,8	598870	3,65E+06	552359	648979	723026
r-16	-	-	-	-	-	-	-	-	-
r-17	927482	1,01E+06	1,15E+06	2,66E+06	3,51E+06	1,03E+07	716304	806430	968567
r-18	912284	974979	1,05E+06	231830	1,68E+06	8,87E+06	721826	781695	851545
r-19	-	-	-	-	-	-	-	-	-
r-20	-	-	-	-	-	-	-	-	-
r-21	1,06E+06	1,09E+06	1,11E+06	720366	724775	740199	824784	857394	881802
r-22	-	-	-	-	-	-	-	-	-
r-23	-	-	-	-	-	-	-	-	-
r-24	1,04E+06	1,07E+06	1,14E+06	1,01E+06	1,14E+06	1,36E+06	817424	848297	930669
r-25	-	-	-	-	-	-	-	-	-
r-26	-	-	-	-	-	-	-	-	-
r-27	1,10E+06	1,18E+06	1,26E+06	1,51E+06	2,42E+06	3,73E+06	923122	982473	1,04E+06
r-28	-	-	-	-	-	-	-	-	-
r-29	-	-	-	-	-	-	-	-	-
r-30	1,15E+06	1,15E+06	1,15E+06	3,11E+06	3,11E+06	3,11E+06	941392	941839	942286

TABLE A.15 – Les valeurs des trois critères (Min, Moy, Max) obtenues par RBS (insertion à t=200)

Instance	#Req.Stat.	#Req.Dyn.	Véhicule	% réussi.	%insert.	CPU	#Front
r-1	10	7	2	40	72,121	0,6575	2,5
r-2	10	7	4	100	100	1,4167	3,9
r-3	10	7	6	100	100	1,9407	4,8
r-4	20	6,9	2	10	48,0995	1,516	1
r-5	20	6,9	4	90	91,1111	5,01722	6
r-6	20	6,9	6	90	91,1111	9,60067	4,33333
r-7	30	7	2	0	18,3335	-	0
r-8	30	7	4	80	92,6282	6,40275	5,25
r-9	30	7	6	100	100	9,0899	6,7
r-10	40	7	2	0	9,9792	-	0
r-11	40	7	4	40	50,9816	15,375	5
r-12	40	7	6	90	94,058	19,9693	6,33333
r-13	50	7	2	0	13,7629	-	0
r-14	50	7	4	20	37,7654	17,33	3
r-15	50	7	6	50	55,0995	44,2876	10
r-16	60	7	2	0	7,96421	-	0
r-17	60	7	4	0	19,8041	-	0
r-18	60	7	6	60	63,4992	69,0087	4,83333
r-19	70	7	2	0	4,4975	-	0
r-20	70	7	4	0	9,30038	-	0
r-21	70	7	6	30	44,417	79,479	7,66667
r-22	80	7	2	0	5,38727	-	0
r-23	80	7	4	0	19,4881	-	0
r-24	80	7	6	20	38,2516	71,6365	3,5
r-25	90	7	2	0	4,86354	-	0
r-26	90	7	4	0	13,9265	-	0
r-27	90	7	6	20	47,5508	150,103	1,5
r-28	100	7	2	0	2,8403	-	0
r-29	100	7	4	0	9,35233	-	0
r-30	100	7	6	0	12,6801	-	0

TABLE A.16 – Les résultats de RBS pour t=400

Req	Coût Min.	Coût Moy.,	Coût Max.	QoS Min.	QoS Moy,	QoS Max.	CO_2 Min.	CO_2 Moy.	CO_2 Max.
r-1	222474	236196	254856	21068,2	5,93E+06	1,65E+07	169933	191410	199395
r-2	207767	244766	301482	19,3497	445266	4,68E+06	147648	184113	234823
r-3	180457	242110	281921	54,4271	531616	4,37E+06	113079	153083	208074
r-4	306743	306743	306743	2,69E+06	2,69E+06	2,69E+06	247342	247342	247342
r-5	339242	419097	492466	225576	2,09E+06	6,62E+06	253164	317197	378398
r-6	331863	378329	451865	3,91294	594507	6,64E+06	236832	265638	304371
r-7	-	-	-	-	-	-	-	-	-
r-8	465172	527086	592380	0	714281	2,52E+06	342530	406608	474566
r-9	473618	518689	559763	98,6526	687891	7,07E+06	325010	373584	410292
r-10	-	-	-	-	-	-	-	-	-
r-11	607704	634671	663472	2828,3	4,71E+06	1,00E+07	468151	493344	510853
r-12	581867	661281	779861	2828,3	1,59E+06	8,24E+06	417025	512808	647946
r-13	-	-	-	-	-	-	-	-	-
r-14	691380	739486	782863	2,01E+06	5,32E+06	9,89E+06	554738	593238	632355
r-15	702161	852109	932294	3,98E-13	1,81E+06	6,90E+06	539764	666758	735042
r-16	-	-	-	-	-	-	-	-	-
r-17	-	-	-	-	-	-	-	-	-
r-18	875701	924024	994913	46851,4	866139	4,52E+06	693939	741144	796917
r-19	-	-	-	-	-	-	-	-	-
r-20	-	-	-	-	-	-	-	-	-
r-21	951494	1,01E+06	1,07E+06	1,42E+06	2,43E+06	2,68E+06	720307	790700	863146
r-22	-	-	-	-	-	-	-	-	-
r-23	-	-	-	-	-	-	-	-	-
r-24	944019	963571	976885	1,35E+06	1,38E+06	1,46E+06	770612	797256	817430
r-25	-	-	-	-	-	-	-	-	-
r-26	-	-	-	-	-	-	-	-	-
r-27	1,05E+06	1,07E+06	1,09E+06	696496	1,13E+06	1,98E+06	825314	841091	868660
r-28	-	-	-	-	-	-	-	-	-
r-29	-	-	-	-	-	-	-	-	-
r-30	-	-	-	-	-	-	-	-	-

TABLE A.17 – Les valeurs des trois critères (Min, Moy, Max) obtenues par RBS (insertion à t=400)

Instance	#Req.Stat.	#Req.Dyn.	Véhicule	% réussi.	%insert.	CPU	#Front
r-1	10	4,8	2	30	64,3496	0,293667	5,66667
r-2	10	4,8	4	100	100	0,6655	5,6
r-3	10	4,8	6	100	100	0,8326	3,6
r-4	20	4,8	2	20	39,9977	2,0765	1,5
r-5	20	4,8	4	70	72,791	1,70586	3,28571
r-6	20	4,8	6	60	68,149	2,37333	6
r-7	30	5	2	0	23,3031	-	0
r-8	30	5	4	90	98,5714	3,14056	5,11111
r-9	30	5	6	100	100	4,0057	6,2
r-10	40	5	2	0	20,3849	-	0
r-11	40	5	4	60	87,4628	6,04767	2,5
r-12	40	5	6	100	100	7,3277	6,5
r-13	50	5	2	0	10,4422	-	0
r-14	50	5	4	30	32,6106	16,935	6,33333
r-15	50	5	6	50	50,9076	19,2988	4,2
r-16	60	5	2	0	3,76475	-	0
r-17	60	5	4	0	7,27877	-	0
r-18	60	5	6	40	44,4748	32,3128	4,25
r-19	70	5	2	0	6,94763	-	0
r-20	70	5	4	0	20,5524	-	0
r-21	70	5	6	30	31,513	53,4053	6
r-22	80	5	2	0	4,4146	-	0
r-23	80	5	4	0	10,6961	-	0
r-24	80	5	6	30	43,4047	81,009	4,66667
r-25	90	5	2	0	7,91556	-	0
r-26	90	5	4	0	14,7091	-	0
r-27	90	5	6	20	38,1654	40,099	2
r-28	100	5	2	0	3,91045	-	0
r-29	100	5	4	0	4,57958	-	0
r-30	100	5	6	10	22,6481	49,616	14

TABLE A.18 – Les résultats de RBS pour t=600

Req	Coût Min.	Coût Moy.,	Coût Max.	QoS Min.	QoS Moy,	QoS Max.	CO_2 Min.	CO_2 Moy.	CO_2 Max.
r-1	177515	204889	236165	0	1,82E+06	1,27E+07	149247	166506	184698
r-2	143339	224058	296640	1,14E-13	1,58E+06	1,46E+07	103733	164965	221543
r-3	183777	230266	279618	0	262778	1,51E+06	106097	140261	196300
r-4	340544	344492	348144	1,51E+06	2,03E+06	2,77E+06	282545	289824	296599
r-5	291880	404661	467143	899,816	3,04E+06	1,22E+07	230332	315518	354086
r-6	320892	386081	444877	3,91294	173352	1,18E+06	203631	258297	297397
r-7	-	-	-	-	-	-	-	-	-
r-8	432034	486763	542257	48509,5	626007	2,37E+06	325123	376485	415604
r-9	409345	490643	548812	1,20462	510633	5,11E+06	297592	359814	453322
r-10	-	-	-	-	-	-	-	-	-
r-11	507994	589472	625366	77981,3	2,32E+06	5,59E+06	387707	455135	488788
r-12	553937	617003	734842	0,00061217	817209	6,92E+06	389925	474440	613611
r-13	-	-	-	-	-	-	-	-	-
r-14	665419	740959	781869	444325	2,75E+06	3,43E+06	526725	604533	631644
r-15	633467	834220	893470	227,057	827591	1,11E+06	508459	651405	712581
r-16	-	-	-	-	-	-	-	-	-
r-17	-	-	-	-	-	-	-	-	-
r-18	855659	909587	963548	16416,2	1,43E+06	2,70E+06	649091	723249	798392
r-19	-	-	-	-	-	-	-	-	-
r-20	-	-	-	-	-	-	-	-	-
r-21	840162	892013	936518	27204,9	784215	2,62E+06	666792	716605	756390
r-22	-	-	-	-	-	-	-	-	-
r-23	-	-	-	-	-	-	-	-	-
r-24	862248	885710	908880	256339	698470	1,06E+06	699634	715294	740851
r-25	-	-	-	-	-	-	-	-	-
r-26	-	-	-	-	-	-	-	-	-
r-27	945682	1,02E+06	1,06E+06	811132	7,16E+06	9,45E+06	751732	811298	833162
r-28	-	-	-	-	-	-	-	-	-
r-29	-	-	-	-	-	-	-	-	-
r-30	1,03E+06	1,06E+06	1,08E+06	1,02E+06	1,03E+06	1,05E+06	857078	876726	893808

TABLE A.19 – Les valeurs des trois critères (Min, Moy, Max) obtenues par RBS (insertion à t=600)

Instance	#Req.Stat.	#Req.Dyn.	Véhicule	% réussi.	%insert.	CPU	#Front
r-1	10	2,7	2	60	72,2179	0,0495	1
r-2	10	2,7	4	90	100	0,0858889	3,11111
r-3	10	2,7	6	90	92	0,114	4
r-4	20	2,6	2	40	61,5895	0,3955	1
r-5	20	2,6	4	80	92,8205	0,199625	3
r-6	20	2,6	6	70	82,0192	0,353429	3,28571
r-7	30	2,9	2	20	50,7604	0,678	1
r-8	30	2,9	4	90	98	0,624	4,77778
r-9	30	2,9	6	90	90,5882	0,517	4,77778
r-10	40	3	2	0	18,7615	-	0
r-11	40	3	4	70	83,6228	1,86714	2,42857
r-12	40	3	6	90	90,4762	1,41033	2,66667
r-13	50	3	2	0	11,0094	-	0
r-14	50	3	4	30	35,8247	2,98367	1,66667
r-15	50	3	6	70	78,7254	3,73386	6,71429
r-16	60	3	2	0	8,46509	-	0
r-17	60	3	4	30	46,2541	6,15667	2,66667
r-18	60	3	6	70	75,7709	10,1946	6,71429
r-19	70	3	2	0	8,17806	-	0
r-20	70	3	4	10	24,3422	41,787	11
r-21	70	3	6	30	41,5204	22,463	8,33333
r-22	80	3	2	0	7,87598	-	0
r-23	80	3	4	0	20,5664	-	0
r-24	80	3	6	40	52,1741	17,3955	1,75
r-25	90	3	2	0	7,95883	-	0
r-26	90	3	4	0	4,3065	-	0
r-27	90	3	6	0	8,51488	-	0
r-28	100	3	2	0	9,54901	-	0
r-29	100	3	4	0	11,5775	-	0
r-30	100	3	6	20	24,6473	16,245	4

TABLE A.20 – Les résultats de RBS pour t=800

Req	Coût Min.	Coût Moy.,	Coût Max.	QoS Min.	QoS Moy,	QoS Max.	CO_2 Min.	CO_2 Moy.	CO_2 Max.
r-1	102803	199719	280077	0	1,46E+06	3,80E+06	69560,9	160309	237967
r-2	122673	197037	289162	0	398904	2,63E+06	84546	145535	214703
r-3	168061	229300	297383	20,5174	545945	3,97E+06	84171,8	141377	197165
r-4	218675	300080	348429	0	3,47E+06	9,42E+06	206258	261980	289022
r-5	282776	394598	464082	864,498	4,13E+06	1,53E+07	229537	306979	341956
r-6	326127	379876	419218	4,26E-13	900689	3,82E+06	187516	257157	282159
r-7	407998	408309	408621	1,22E+06	1,26E+06	1,30E+06	303792	307554	311316
r-8	383819	468259	513830	474,242	1,11E+06	6,05E+06	318462	356129	414542
r-9	407872	455404	511797	1,14E-13	1,08E+06	4,93E+06	285395	328067	404946
r-10	-	-	-	-	-	-	-	-	-
r-11	500310	569912	638268	157,119	453344	4,37E+06	373846	443077	471903
r-12	531052	603337	662140	6,82E-13	236140	3,71E+06	358736	475615	559351
r-13	-	-	-	-	-	-	-	-	-
r-14	622617	663933	711243	12776,4	2,34E+06	4,99E+06	491544	519206	539940
r-15	581700	732242	841278	0,00128724	559591	3,16E+06	466856	586951	667361
r-16	-	-	-	-	-	-	-	-	-
r-17	719090	757985	837809	382355	705624	2,49E+06	577920	607755	675839
r-18	736510	842570	919661	18,9271	309894	3,08E+06	624899	679790	750708
r-19	-	-	-	-	-	-	-	-	-
r-20	745158	761391	777533	57058,6	165135	283846	603654	617241	631647
r-21	675040	842638	916083	18,2113	2,17E+06	5,30E+06	586685	671260	738489
r-22	-	-	-	-	-	-	-	-	-
r-23	-	-	-	-	-	-	-	-	-
r-24	764825	854760	912818	299989	2,04E+06	4,13E+06	599579	687251	728981
r-25	-	-	-	-	-	-	-	-	-
r-26	-	-	-	-	-	-	-	-	-
r-27	-	-	-	-	-	-	-	-	-
r-28	-	-	-	-	-	-	-	-	-
r-29	-	-	-	-	-	-	-	-	-
r-30	871085	890461	897358	1,26E+06	2,07E+06	2,98E+06	728716	744955	751844

TABLE A.21 – Les valeurs des trois critères (Min, Moy, Max) obtenues par RBS (insertion à t=800)

Bibliographie

[Ambrosino *et al.*, 2004] Ambrosino, G., Nelson, J. et Romnazzo, M. (2004). Demand responsive transport services : Towards the flexible mobility agency. *Rome, ENEA,*, page 325 p.

[Atahran *et al.*, 2011a] Atahran, A., Lenté, C. et T'Kindt, V. (2011a). Un algorithme génétique pour résoudre un problème de darp multi-objectif. *In In Actes du 12me congrés de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF11), Saint-Étienne, France, .*

[Atahran *et al.*, 2011b] Atahran, A., Lenté, C. et T'Kindt, V. (2011b). Optimal timing computation for quality of service in a dial a ride problem. *In International Conference on Operations Research, Zurich, Switzerland.*

[Atahran *et al.*, 2011c] Atahran, A., Lenté, C. et T'Kindt, V. (2011c). Solution of a multicriteria dial-a-ride problem with an ecological measure and heterogeneous vehicles. *Proceedings of International Conference on Industrial Engineering and Systems Management*, pages 1233–1242.

[Atahran *et al.*, 2012] Atahran, A., Lenté, C. et T'Kindt, V. (2012). A multicriteria dial-a-ride problem with an ecological measure and heterogeneous vehicles. *soumis à Journal of Multi-Criteria Decision Analysis.*

[Attanasio *et al.*, 2004] Attanasio, A., Cordeau, J., Ghiani, G. et Laporte, G. (2004). Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30(3):377–387.

[Baugh *et al.*, 1998] Baugh, J., Kakivaya, D. et Stone, J. (1998). Intracability of the dial-a-ride problem and a multi-objective solution using simulated annealing. *Engineering Optimisation*, 30(2):91–124.

[Beaudry *et al.*, 2010] Beaudry, A., Laporte, G., Melo, T. et Nickel, S. (2010). Dynamic transportation of patients in hospitals. *OR spectrum*, 32(1):77–107.

[Bent et Van Hentenryck, 2003] Bent, R. et Van Hentenryck, P. (2003). Dynamic vehicle routing with stochastic requests. *In International Joint Conference on Artificial Intelligence*, volume 18, pages 1362–1363. Lawrence Erlbaum Associates Ltd.

[Berbeglia *et al.*, 2007] Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I. et Laporte, G. (2007). Static pickup and delivery problems : a classification scheme and survey. *Operation Research*, 15:1–31.

[Berbeglia *et al.*, 2010a] Berbeglia, G., Cordeau, J.-F. et Laporte, G. (2010a). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8 – 15.

[Berbeglia *et al.*, 2010b] Berbeglia, G., Cordeau, J.-F. et Laporte, G. (2010b). A hybrid tabu search and constraint programming algorithm for the dynamic dial a ride problem. *CIIRRELT Working Paper*, 14.

[Berbeglia *et al.*, 2010c] Berbeglia, G., Laporte, G., Cordeau, J. et Centre interuniversitaire de recherche sur les réseaux d'entreprise, l. l. e. l. t. (2010c). *A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem.* CIRRELT.

[Bianchi, 2000] Bianchi, L. (2000). Notes on dynamic vehicle routing-the state of the art. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale.*

[Borndörfer *et al.*, 1997] Borndörfer, R., Grötschel, M., Klostermeier, F. et Küttner, C. (1997). Telebus berlin : vehicle scheduling in a diala-ride system. *Technical Report SC 97-23, Konrad-Zuse-Zentrum für Informationstechnik, Berlin.*

[Bräysy et Gendreau, 2005] Bräysy, O. et Gendreau, M. (2005). Vehicle routing problem with time windows, part i : Route construction and local search algorithms. *Transportation science*, 39(1):104–118.

[Burkhardt *et al.*, 1995] Burkhardt, J., Hamby, B. et McGavock, A. (1995). *Unsers' manual for assesing service delivery systems for rural passenger transportation.* TCRP Report 6, Washington : National Academy Press.

[Castex, 2007] Castex, E. (2007). Le transport a la demande (tad) en france : de l'état des lieux à l'anticipation. modélisation des caractéristiques fonctionnelles des tad pour développer les modes flexibles de demain.

[Chevrier *et al.*, 2010] Chevrier, R., Liefooghe, A., Jourdan, L. et Dhaenens, C. (2010). On Optimizing a Demand Responsive Transport with an Evolutionary Multiobjective Approach. *In ITSC'2010, 13th IEEE Intelligent Transport Systems Conference*, Madeire, Portugal.

[Chrétienne et Sourd, 2003] Chrétienne, P. et Sourd, F. (2003). Pert scheduling with convex cost functions. *Theoretical Computer Science*, 292(1):145–164.

[Chrétienne, 2001] Chrétienne, P. (2001). Minimizing the earliness and tardiness cost of a sequence of tasks on a single machine. *RAIRO - Operations Research - Recherche Opérationnelle*, 35(2):165–187.

[Coello *et al.*, 2007] Coello, C., Lamont, G. et Van Veldhuizen, D. (2007). *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer-Verlag New York Inc.

[Colorni et Righini, 2001] Colorni, A. et Righini, G. (2001). Modeling and optimizing dynamic dial-a-ride problems. *International Transactions in Operational Research*, 8(2): 155–166.

[Cordeau, 2006] Cordeau, J. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54:573–586.

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

[Cordeau et Laporte, 2003] Cordeau, J. et Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research*, B(37):579–594.

[Coslovich *et al.*, 2006] Coslovich, L., Pesenti, R. et Ukovich, W. (2006). A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research*, 175(3):1605–1615.

[Cubillos *et al.*, 2009] Cubillos, C., Urra, E. et Rodríguez, N. (2009). Application of genetic algorithms for the DARPTW problem. *International Journal of Computers, Communication and Control*, 4(2):127–136.

[Deb *et al.*, 2002] Deb, K., Pratap, A., Agrawal, S. et Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[Della Croce *et al.*, 2004] Della Croce, F., Ghirardi, M. et Tadei, R. (2004). Recovering beam search : Enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics*, 10(1):89–104.

[Della Croce *et al.*, 2002] Della Croce, F., Tkindt, V. *et al.* (2002). A recovering beam search algorithm for the one-machine dynamic total completion time scheduling problem. *Journal of the Operational Research Society*, 53(11):1275–1280.

[Desrosiers *et al.*, 1995] Desrosiers, J., Dumas, Y., Solomon, M. et Soumis, F. (1995). Time constrained routing and scheduling. *Handbooks in operations research and management science*, 8:35–139.

[Desrosiers *et al.*, 1986] Desrosiers, J., Dumas, Y. et Soumis, F. (1986). A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6(3):301–326.

[Dessouky *et al.*, 2003] Dessouky, M., Rahimi, M. et Weidner, M. (2003). Jointly optimizing cost, service, and environmental performance in demand-responsive transit scheduling. *Transportation Research Part D : Transport and Environment*, 8(6):433–465.

[Diana et Dessouky, 2004] Diana, M. et Dessouky, M. (2004). A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B : Methodological*, 38(6):539–557.

[Dondo et Cerda, 2007] Dondo, R. et Cerda, J. (2007). A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, 176(3):1478–1507.

[Drezner et Drezner, 2006] Drezner, T. et Drezner, Z. (2006). Multiple facilities location in the plane using the gravity model. *Geographical Analysis*, 38(4):391–406.

[Ehrgott et Gandibleux, 2000] Ehrgott, M. et Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *Or Spectrum*, 22(4):425–460.

[Erdoğan *et al.*, 2010] Erdoğan, G., Cordeau, J. et Laporte, G. (2010). The attractive traveling salesman problem. *European Journal of Operational Research*, 203(1):59–69.

[Fu, 2002] Fu, L. (2002). Scheduling dial-a-ride paratransit under time-varying, stochastic congestion. *Transportation Research Part B : Methodological*, 36(6):485–506.

[Garaix, 2007] Garaix, T. (2007). *Étude et résolution exacte de problèmes de transport à la demande avec qualité de service*. These, Université d'Avignon.

[Garey *et al.*, 1988] Garey, M., Tarjan, R. et Wilfong, G. (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research*, pages 330–348.

[GART, 1997] GART, P. (1997). Le transport à la demande, état de l'art et perspectives. Rapport technique, Rapport public de recherche , IDF Conseil, Paris.

[Ghirardi et Potts, 2005] Ghirardi, M. et Potts, C. (2005). Makespan minimization for scheduling unrelated parallel machines : A recovering beam search approach. *European Journal of Operational Research*, 165(2):457–467.

[Gillett et Miller, 1974] Gillett, B. et Miller, L. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, pages 340–349.

[Goldberg et ., 1985] Goldberg, D. et ., R. L. (1985). Alleles, loci, and the tsp. *proceeding*, pages 154–159.

[Gordon *et al.*, 2002] Gordon, V., Proth, J. et Chu, C. (2002). A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, 139(1):1–25.

[Grabisch, 2006] Grabisch, M. (2006). L'utilisation de l'intégrale de choquet en aide multicritère à la décision. *Newsletter of the European Working Group Multicriteria Aid for Decisions*, 3(14):5–10.

[Hanne *et al.*, 2009] Hanne, T., Melo, T. et Nickel, S. (2009). Bringing robustness to patient flow management through optimized patient transports in hospitals. *Interfaces*, 39(3):241.

[Honda *et al.*, 2003] Honda, N., Mohri, S. et Ishii, H. (2003). Backtracking beam search applied to multi-objective scheduling problem. *In The Fifth Metaheuristics International Conference, Kyoto*, pages 1–6.

[Hoogeveen et Velde, 1996] Hoogeveen, J. et Velde, S. (1996). A branch-and-bound algorithm for single-machine earliness-tardiness scheduling with idle time. *INFORMS Journal on Computing : charting new directions in OR and CS*, pages 403–412.

[Horn, 2002] Horn, M. (2002). Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research Part C : Emerging Technologies*, 10(1):35–63.

[Huff, 1964] Huff, D. (1964). Defining and estimating a trading area. *The Journal of Marketing*, pages 34–38.

[Jain *et al.*, 1999] Jain, A., Murty, M. et Flynn, P. (1999). Data clustering : a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

[Jaszkiewicz, 2004] Jaszkiewicz, A. (2004). Evaluation of multiple objective metaheuristics. *Metaheuristics for Multiobjective Optimisation*, pages 65–89.

[Jaw *et al.*, 1986] Jaw, J.-J., Odoni, A. R., Psaraftis, H. N. et Wilson, N. H. (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B : Methodological*, 20(3):243 – 257.

[Jorgensen *et al.*, 2007] Jorgensen, R., Larsen, J. et Begvinsdottir, K. (2007). Solving the dial-a-ride problem using genetic algorithms. *J Oper Res Soc*, 58:1321–1331.

[Kaminsky *et al.*, 2004] Kaminsky, P., Hochbaum, D. et Leung, J. (2004). Due date quotation models and algorithms. *Handbook of Scheduling : Algorithms, Models, and Performance Analysis*.

[Kessler, 2004] Kessler, D. (2004). *Computer-aided scheduling and dispatch in demand-responsive transit services*, volume 57. Transportation Research Board.

[Kung *et al.*, 1975] Kung, H., Luccio, F. et Preparata, F. (1975). On finding the maxima of a set of vectors. *Journal of the ACM (JACM)*, 22(4):469–476.

[Lacomme *et al.*, 2006] Lacomme, P., Prins, C. et Sevaux, M. (2006). A genetic algorithm for a bi-objective capacitated arc routing problem. *Computers & Operations Research*, 33(12):3473–3493.

[Larsen et Madsen, 2000] Larsen, A. et Madsen, O. (2000). *The dynamic vehicle routing problem*. Thèse de doctorat, Technical University of Denmark, Department of Informatics and Mathematical Modeling.

[Lim *et al.*, 2006] Lim, A., Rodrigues, B. et Zhang, X. (2006). Scheduling sports competitions at multiple venues-revisited. *European journal of operational research*, 175(1):171–186.

[Madsen *et al.*, 1995] Madsen, O., Ravn, H. et Rygaard, J. (1995). A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research*, 60(1):193–208.

[Malucelli *et al.*, 1999] Malucelli, F., Nonato, M. et Pallottino, S. (1999). Demand adaptive systems : some proposals on flexible transit. *Operations research in industry*, pages 157–182.

[Masson *et al.*, 2011] Masson, R., Lehuédé, F. et Péton, O. (2011). A tabu search algorithm for the Dial-a-Ride Problem with Transfers. *In Proceedings of the IESM 2011 Conference*, pages –, Metz, France.

[Mauri et Lorena, 2006] Mauri, G. et Lorena, L. (2006). A multiobjective model and simulated annealing approach for a dial-a-ride problem. *In Workshop dos Cursos de Computação. http ://mtc18. sid. inpe. br/col/sid. inpe. br/ePrint@ 80/2006/10.09. 16.10/doc/v1. pdf*.

[M.BLONDEAU, 2004] M.BLONDEAU (2004). Les transports à la demande entre droit au transport et recherche de performance. Mémoire de D.E.A., Grenoble : Université Joseph Fourier.

[Paquette *et al.*, 2009] Paquette, J., Cordeau, J. et Laporte, G. (2009). Quality of service in dial-a-ride operations. *Computers & Industrial Engineering*, 56(4):1721–1734.

[Paquette *et al.*, 2011] Paquette, J., Cordeau, J., Laporte, G. et Pascoal, M. (2011). Combining multicriteria analysis and tabu search for dial-a-ride problems.

[Parragh, 2009] Parragh, S. (2009). *Ambulance Routing Problems with Rich Constraints and Multiple Objectives*. Thèse de doctorat, University of Vienna, Faculty of Business, Economics and Statistics.

[Parragh *et al.*, 2008] Parragh, S., Doerner, K. et Hartl, R. (2008). A survey on pickup and delivery problems. part i : transportation between customers and depot. *J. Betriebswirt*, 58:21–51.

[Parragh *et al.*, 2010] Parragh, S., Doerner, K. et Hartl, R. (2010). Variable neighborhood search for dial-a-ride problem. *Computers and OR*, 37(6):1129–1138.

[Parragh *et al.*, 2009] Parragh, S., Doerner, K., Hartl, R. et Gandibleux, X. (2009). A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks*, 54(4):227–242.

[Parragh et K.F. Doerner, 2008] Parragh, S. et K.F. Doerner, R. (2008). A survey on pickup and delivery problems. part ii : transportation between pickup and delivery locations. *J. Betriebswirt*, 58:81–117.

[Parragh, 2011] Parragh, S. N. (2011). Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C : Emerging Technologies*, 19(5):912 – 930.

[Psaraftis, 1980] Psaraftis, H. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial a ride problem. *Transportation Science*, 14:130–154.

[Psaraftis, 1983] Psaraftis, H. (1983). An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17:351–357.

[Psaraftis, 1995] Psaraftis, H. (1995). Dynamic vehicle routing : Status and prospects. *annals of Operations Research*, 61(1):143–164.

[Quadrifoglio et Dessouky, 2004] Quadrifoglio, L. et Dessouky, M. (2004). Mobility allowance shuttle transit (mast) services : formulation and simulation comparison with conventional fixed route bus services. *In Modelling, simulation, and optimization Á Proceedings of the 4th IASTED international conference. Kauai, HI, USA, 17Á19 August. Calgary : Acta Press, 6pp.*

[Renaud et Boctor, 2002] Renaud, J. et Boctor, F. (2002). A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, 140(3):618–628.

[Richardson *et al.*, 1989] Richardson, J., Palmer, M., Liepins, G. et Hilliard, M. (1989). Some guidelines for genetic algorithms with penalty functions. *In Proceedings of the third international conference on Genetic algorithms*, pages 191–197. Morgan Kaufmann Publishers Inc.

[Ropke et Cordeau, 2009] Ropke, S. et Cordeau, J. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286.

[Roudenko, 2004] Roudenko, O. (2004). *Application des algorithmes évolutionnaires aux problèmes d'optimisation multi-objectif avec contraintes.* Thèse de doctorat.

[Schaffer, 1985] Schaffer, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *In Proceedings of the 1st international Conference on Genetic Algorithms*, pages 93–100. L. Erlbaum Associates Inc.

[Sexton et Bodin, 1985] Sexton, T. et Bodin, L. (1985). Optimizing single vehicle many-to-many operations with desired delivery times : I. routing. *Transportation Science*, 19(4):411–435.

[Sharma *et al.*, 2009] Sharma, S., Ukkusuri, S. et Mathew, T. (2009). Pareto optimal multiobjective optimization for robust transportation network design problem. *Transportation Research Record : Journal of the Transportation Research Board*, 2090(-1):95–104.

[Smerk, 1991] Smerk, G. (1991). *The federal role in urban mass transportation.*

[Smith *et al.*, 2003] Smith, B., Demetsky, M. et Durvasula, P. (2003). A multiobjective optimization model for flexroute transit service design. *Journal of Public Transportation*, 6(1):81–100.

[Solomon, 1987] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

[Talbi, 1999] Talbi, E. (1999). Métaheuristiques pour l'optimisation combinatoire multiobjectifs : Etat de l'art. *Rapport CNET (France Telecom) Octobre.*

[Teodorovic et Radivojevic, 2000] Teodorovic, D. et Radivojevic, G. (2000). A fuzzy logic approach to dynamic dial-a-ride problem. *Fuzzy Sets and Systems*, 116:23–33.

[T'kindt et Billaut, 2002] T'kindt, V. et Billaut, J.-C. (2002). *Multicriteria scheduling : theory, models and algorithms.* Springer Verlag.

[T'kindt et Billaut, 2006] T'kindt, V. et Billaut, J.-C. (2006). Multicriteria scheduling : Theory, models and algorithms. *Springer-Verlag (Heidelberg), seconde édition,*, page 360 pages.

[Toth et Vigo, 1997] Toth, P. et Vigo, D. (1997). Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, 31(1):60–71.

[Ulungu *et al.*, 1999] Ulungu, E., Teghem, J., Fortemps, P. et Tuyttens, D. (1999). Mosa method : a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236.

[Valente et Alves, 2005] Valente, J. et Alves, R. (2005). Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time. *Computers & Industrial Engineering*, 48(2):363–375.

[Van Veldhuizen et Lamont, 1998] Van Veldhuizen, D. et Lamont, G. (1998). Evolutionary computation and convergence to a pareto front. *In Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 221–228.

[Veldhuizen, 1999] Veldhuizen, D. A. V. (1999). *Multiobjective Evolutionary Algorithms : Classifications, Analyses, and New Innovations (1999).* Thèse de doctorat, Departement of Electrical and Computer Enginnering, Graduate School of Engineering, Air Force institute of Technology. Wright-Patterson AFB, Ohio.

[Viana et Pinho de Sousa, 2000] Viana, A. et Pinho de Sousa, J. (2000). Using metaheuristics in multiobjective resource constrained project scheduling. *European Journal of Operational Research*, 120(2):359–374.

[Vidal *et al.*, 2011] Vidal, T., Crainic, T. G., Gendreau, M. et Prins, C. (2011). A unifiying view on timing problems and algorithms. Rapport technique 43, CIRRELT, Montréal (Canada).

[Wolfer Calvo et Colorni, 2007] Wolfer Calvo, R. et Colorni, A. (2007). An effective and fast heuristic for the dial-a-ride problem. *4OR : A Quarterly Journal of Operations Research*, 5(1):61–73.

[Xiang *et al.*, 2006] Xiang, Z., Chu, C. et Chen, H. (2006). A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European Journal of Operational Research*, 174(2):1117–1139.

[Zhihai *et al.*, 2007] Zhihai, X., Chengbin, C. et Haoxun, C. (2007). The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. *European Journal of Operational Research*, 185(2):534–551.

[Zitzler *et al.*, 2000] Zitzler, E., Deb, K. et Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms : Empirical results. *Evolutionary computation*, 8(2):173–195.

[Zitzler *et al.*, 2001] Zitzler, E., Laumanns, M. et Thiele, L. (2001). SPEA2 : Improving the strength pareto evolutionary algorithm.

[Zitzler *et al.*, 1998] Zitzler, E., Thiele, L., Zitzler, E., Zitzler, E., Thiele, L. et Thiele, L. (1998). *An evolutionary algorithm for multiobjective optimization : The strength pareto approach*. Citeseer.

Glossaire

- TSP : Traveling Salesman Problem,
- $MTSP$: Multiple Traveling Salesman Problem,
- $CVRP$: Capacitated Vehicle Routing Problem,
- $VRPTW$: Vehicle Routing Problem with Time Windows,
- PDP : Pickup and Delivery Problem,
- $PDPTW$: Pickup and Delivery Problem with Time Windows,
- $DARP$: Dial-a-Ride Problem,
- SIG : Systèmes d'Informations géographiques,
- GPS : Global Position System,
- SP : Swapping Problem,
- $1-PDTSP$:One-Commodity Pickup and Delivery Traveling Salesman Problem,
- $Q-DTSP$: the Q-Delivery Traveling Salesman Problem,
- $VRPB$: the Vehicle Routing Problem with Backhauls,
- SCP : stacker crane problem,
- CAH : Classification Ascendante Hiérarchique,
- $FAMS$: Flexible Agency for Collective Demand Responsive Mobility and Services,
- $CFFS$: Copenhagen Fire-Fighting Service,

Résumé :

Les travaux présentés dans cette thèse visent à proposer des méthodes permettant de résoudre un problème de Transport à la Demande multicritère.
Le premier travail réalisé dans cette thèse est l'étude d'un problème de *Dial-a-Ride* (DARP) statique multicritère. Trois critères qui peuvent être conflictuels ont été définis : le premier consiste à minimiser le coût de transport, le deuxième critère consiste à minimiser l'insatisfaction des passagers et enfin le troisième critère consiste à minimiser la quantité de CO_2 émise par l'ensemble des véhicules. Nous avons développé une méthode évolutionnaire NSGA-II pour chercher un ensemble approximatif d'optimas de Pareto.
Le second travail réalisé est l'étude d'un problème d'*Optimal Timing* dans une tournée. Ce problème consiste à calculer les dates de début de service optimales des points d'arrêts d'une tournée afin de minimiser l'insatisfaction des passagers.
Le dernier travail de cette thèse a porté sur l'étude d'un problème de Transport à la Demande dynamique dans lequel de nouvelles requêtes à traiter arrivent en cours de journée. Deux méthodes ont été proposées pour résoudre ce problème : la première est une heuristique d'insertion rapide et la seconde est une méthode arborescente tronquée connue sous le nom de *Recovering Beam Search*.

Mots clés :

Problème de Transport à la Demande, statique, dynamique, horodatage, optimisation multicritère, algorithmes évolutionnaires multi-objectifs.