Optimization in one-way car sharing with electric cars and relocations

Mohammed Amine Ait Ouahmed

ACADÉMIE D'AIX-MARSEILLE
UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE

THÈSE

présentée à l'Université d'Avignon et des Pays de Vaucluse
pour obtenir le diplôme de DOCTORAT

SPÉCIALITÉ : Informatique

École Doctorale 536 «Sciences et Agrosciences»
Laboratoire Informatique d'Avignon (EA 4128)

Optimisation dans l'auto-partage à un seul sens avec voitures électriques et relocalisations

par
Mohammed Amine Ait-ouahmed

Soutenue publiquement le 15 Octobre 2018 devant un jury composé de :

M^{me}	Feng Chu	Professeur, IBISC, Université d'Evry	Rapporteur
M^{me}	Marie-Jo Huguet	Professeur, LAAS, INSA de Toulouse	Rapporteur
M.	Alain Lhostis	Chercheur HDR, IFSTTAR, LVMT, Paris	Examinateur
M.	David Coudert	Directeur de Recherche, INRIA Sophia Antipolis	Examinateur
M.	Thomas Devogele	Professeur, LI, Université de Tours	Examinateur
M.	Didier JOSSELIN	Directeur de Recherche, UMR ESPACE 7300, CNRS	Directeur de thèse
M.	Fen Zhou	Maître de conférence, LIA, Université d'Avignon	Co-Directeur de thèse

---

Laboratoire d'Informatique d'Avignon

Remerciements

En préambule à cette thèse, je souhaite adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leur aide et qui ont contribué à l'élaboration de cette thèse.

Ainsi, je tiens à remercier sincèrement M. Didier Josselin et M. Fen Zhou, en tant qu'encadrants de cette thèse. Leurs directives et apports scientifiques ont été cruciaux au bon déroulement de la thèse, j'ai pu apprendre aussi à leurs cotés les aspects méthodologiques nécessaires pour aborder un travail scientifique. En outre, sur un plan humain, ils se sont toujours montrés bienveillants, encourageants et à l'écoute tout au long de la réalisation de ce travail. Je les remercie aussi pour leur patience, l'aide et le temps qu'ils ont bien voulu me consacrer.

Mes remerciements s'adressent également à Mme Monia Rekik, professeur à l'université de Laval et sous la tutelle de laquelle j'ai pu effectuer un stage de trois mois au centre de recherche sur la logistique et le transport (CIRRELT) durant lequel une partie de ce travail a été effectuée.

J'exprime ma gratitude à tous les chercheurs du laboratoire informatique d'Avignon, rencontrés lors des recherches effectuées et avec qui j'ai eu la chance d'échanger scientifiquement et amicalement.

Un grand merci à mes parents qui m'ont toujours soutenu, encouragé et su me donner l'ambition de faire des hautes études et sans lesquels je n'en serais pas là au sens propre comme au figuré ! Également à mon autre moitié, la femme de ma vie, pour son aide et son soutien et au prés de laquelle je m'excuse pour les week-ends passés au bureau. Sans oublier, mon petit frère qui a toujours été à mes cotés et s'est montré très serviable dans les moments à forte pression sans que ça frêne son ambition de faire une thèse ! Je n'oublie pas aussi de remercier chaleureusement ma petite sœur pour sa contribution.

Enfin, j'adresse mes plus sincères remerciements à tous mes amis, qui m'ont toujours soutenu et encouragé au cours de la réalisation de cette thèse.

Merci à toutes et à tous.

Résumé

Cette thèse a pour objectif de modéliser et résoudre des problèmes d'optimisation d'un système d'auto-partage avec des voitures électriques dit «*à un seul sens*», où les utilisateurs peuvent prendre une voiture dans une station et la laisser ensuite dans une autre. Ce fonctionnement conduit généralement à une situation de déséquilibre dans la répartition des voitures avec certaines stations pleines et d'autres vides. Une des solutions utilisées par les opérateurs d'auto-partage pour palier ce problème est le recours à des agents pour déplacer les voitures selon le besoin. Identifier et répondre à ce besoin est un problème d'optimisation non trivial, notamment à cause de l'usage de véhicules électriques, ce qui engendre des contraintes de rechargement de batteries et d'autonomie. Le problème d'optimisation est décomposé en deux sous-problèmes : le premier est le problème d'affectation des voitures aux clients, ainsi que leurs routages, que nous nommons *ROCSP* pour *Recharging One way Car Sharing Problem* ; le second problème est celui du planning des agents et leurs routages que nous nommons *ESRP* pour *Employee Scheduling Routing Problem*.

1. Résolution du *ROCSP* : deux modélisations en Programmation Linéaire en Nombres Entiers (PLNE) sont proposées, la première basée sur les flots et la deuxième sur les chemins, ce qui fait que les deux modèles intègrent de manière différente les contraintes de recharge électrique. Comme la résolution exacte à travers les modèles PLNE s'avère très gourmande en temps de calcul et non adaptée aux instances d'auto-partage de taille réelle, nous proposons des heuristiques qui permettent dans un temps raisonnable d'optimiser la redistribution des voitures et la gestion du service. Ces heuristiques permettent de calculer le nombre de voitures et les différentes opérations de relocalisation (redistribution des voitures) à réaliser sur une journée donnée.
2. Résolution du *ESRP* : un modèle PLNE est proposé pour la résolution exacte du *ESRP*, et, en complément, des heuristiques sont proposées pour une résolution approchée et relativement rapide. L'objectif est la détermination du nombre minimal d'agents nécessaire pour effectuer les opérations de relocalisation qui découlent du premier problème, le *ROCSP*.

Dans une partie prospective, et une fois les *ROCSP* et *ESRP* résolus dans leur version statique, nous nous focaliserons sur une autre variante du problème avec réservation dynamique. Nous proposons également d'explorer un nouveau concept - l'auto-copartage - qui se veut une hybridation entre auto-partage et covoiturage.

Les algorithmes proposés ont été validés sur le réseau Auto Bleue de la ville de Nice essentiellement, qui gère une flotte de véhicules électriques, en s'appuyant sur des modèles de génération de flux pour estimer la demande, mais aussi d'autres instances que nous avons générées pour simuler d'autres villes, au sein d'un Système d'Information Géographique.

Mots clefs : Auto-partage de véhicules électriques, relocalisation de voitures, algorithme génétique, Programmation Linéaire en Nombres Entiers (PLNE), génération de colonnes, Système d'Information Géographique (SIG).

Abstract

This thesis aims at modelling and solving optimization problems related to the management of one-way-electric-car-sharing systems, where users can take a car from a station, use it, and then return it to another station. This generally leads to an imbalanced distribution of cars, with some full stations and other empty ones. A solution to this problem, implemented by car-sharing operators, is to employ staff agents to move cars as needed. However, identifying this need is a non-trivial optimization problem, especially since the system may be more constrained when the vehicles used are electric, which generates battery recharging and autonomy constraints. The global optimization problem addressed is then divided into two sub-problems. The first one is assigning the cars to customers, as well as their routing ; it is denoted by ROCSP (Recharging One Way Car Sharing Problem). The second problem involves agents planning and routing ; it is denoted by ESRP (Employee Scheduling Routing Problem).

1. For the ROCSP, we propose two Mixed-integer linear programming (MILP) modelizations of the problem : One based on flows and the other based on paths. This means that the two models inculde the battery-recharging constraints in two different ways. As the exact resolution through the MILP models is quite expensive in terms of computational time and is not adapted for the resolution of real-size car-sharing instances, we introduce heuristics that enable the optimization of cars-redistribution and service management of the service within a reasonable amount of time. These heuristics allows the calculation of the number of cars and the various redistribution operations to be performed on a given day.
2. For the ESRP, this second problem is also addressed with MILP models for the exact resolution, and some heuristics are suggested for an approximate resolution. This process has reasonable calculation time and aims at finding the minimum number of agents to perform the necessary relocation operations that stem from the first problem, namely, the ROCSP.

Once the ROCSP and ESRP solved in their static versions, we then focus on the ROCSP by exploring another variant of the problem : ROCSP with dynamic

reservation. We also suggest to explore a new concept : *Auto-CoPartage*, which is a hybridization of car-sharing and carpooling. The stated algorithms are validated on the Auto Bleue electrical vehicles fleet in the network of the city of Nice, essentially by relying on flow generation models to estimate the demand, but also using other instances that we have generated for other cities. All the data are handled using a Geographical Information System.

keywords : Electric carsharing, car redistribution, genetic algorithm, mixed-integer linear programming (MILP), column generation, geographical information system (GIS).

Table des matières

Introduction générale

Contexte. Le défi du siècle est la lutte contre le réchauffement climatique, auquel nous les êtres humains devons apporter des solutions. La cause principale de ce réchauffement est la présence dans l'atmosphère de gaz à effet de serre, due en partie à l'activité humaine et essentiellement aux émissions de dioxyde de carbone (CO2) dans l'atmosphère. Le secteur des transports est le plus émetteur de CO2, par exemple selon Macqueron (2017) en 2010 en France Métropolitaine, 36,5 % des émissions de CO2 sont dues aux activités de transports dont le transport routier représente 93,6% de ces émissions.

L'une des solutions pour réduire les émissions de gaz à effet de serre réside dans le recours à la voiture électrique qui n'émet pas de CO2 durant son utilisation. Par contre, c'est se bercer d'illusions, que de croire que la solution miracle est le simple fait du passage du véhicule thermique à l'électrique. En France, selon Canaguier et Ag (2012), les émissions de CO2 d'une voiture électrique durant toute sa "vie" équivalent à 40% des émissions d'une thermique, car bien que l'utilisation de voiture électrique ne produit pas de CO2, la source de l'énergie électrique utilisée est souvent nucléaire à hauteur de 80%. Outre la source énergétique, la fabrication des batteries est très émettrice de CO2 et représente 35% des émissions d'une voiture.

En plus du passage à l'électrique, l'idéal serait de réduire le nombre de voitures. La question se pose alors : comment ? Aussi simple que cela puisse paraître, la solution réside dans le partage de voitures entre plusieurs utilisateurs. Mais la culture du partage est devenue étrangère dans la société moderne de consommation, qui pousse chaque individu à avoir une ou plusieurs voitures sans que le besoin n'en soit justifié. Posséder est devenu une fin en soit, ce qui fait qu'il est plus qu'urgent de remettre en question le mode de vie des pays dit développés où la propriété privée en est partie intégrante. Sans oublier de prévenir les pays en voie de développement, que la voie empruntée vers la surconsommation n'est pas nécessairement la bonne et que pour le bien des générations futures, il est essentiel de perpétuer la culture du partage. D'autant que cette culture ne rend pas forcement malheureux, d'après ce qu'en témoigne la photo dans la figure 1, où l'on observe des visages tchadiens bien plus souriants

FIGURE 1 – *Partage optimal de véhicules au Tchad ! Comme dirait un dicton Algérien : l'exiguïté est dans les cœurs ! Source image : content.time.com*

que ceux qu'on peut croiser dans les embouteillages des grandes villes ! Cependant l'espoir est permis, vu que les sociétés modernes font des efforts pour rattraper le retard accumulé en matière de partage, ce qui a donné le jour à des systèmes où les véhicules sont partagés entre plusieurs utilisateurs, comme le covoiturage et l'auto-partage. Dans cette thèse, nous nous intéressons essentiellement à l'auto-partage.

L'auto-partage. L'auto-partage est le partage d'une flotte de voitures généralement entre abonnés au service. Derrière cette définition se cache un principe simple : l'utilisation, occasionnelle et parfois récurrente, d'une voiture sans en être le propriétaire, ce qui permet de faire des économies sur les coûts d'entretien. Un système de partage de voitures est aussi une solution pour les embouteillages et le manque de parkings, par la réduction du nombre de voitures roulant. Selon une enquête nationale française (Louvet et Godillon, 2013), l'auto-partage représente une solution de substitution à la voiture privée. Avant d'être abonnés, environ un tiers des ménages ne possède pas leur propre voiture, tandis que 75 % d'entre eux ne possèdent plus de voiture après la souscription. En outre, une grande proportion des personnes interrogées a déclaré que l'auto-

partage leur a permis de renoncer à l'achat d'une première voiture (34,4 %) ou d'une voiture supplémentaire (8,8 %). La voiture est moins possédée, mais aussi moins utilisée : le nombre de kilomètres parcourus en voiture par an diminue de 41 %, suite à l'adhésion à l'auto-partage.

L'auto-partage dans le monde. Selon bcg.perspectives (2017), l'auto-partage se met en place dans les grandes zones urbaines dans les pays développés et les pays en voie de développement. Bien que le plus grand marché est la région Asie-Pacifique (dont l'Australie, la Chine, le Japon, la Malaisie, la Nouvelle-Zélande, le Singapour, la Corée du Sud et le Taïwan), avec 2,3 millions d'utilisateurs et 33.000 véhicules, l'Europe (y compris la Turquie et la Russie) possède le plus grand service par habitant, avec 2,1 millions d'utilisateurs et 31.000 véhicules. L'Amérique du Nord (le Canada et les États-Unis), 1,5 millions d'utilisateurs partageant 22.000 véhicules. Ensemble, les trois régions représentent 2,5 milliards de minutes d'utilisation par an et 650 millions d'euros en chiffre d'affaires (cf. figure 2).

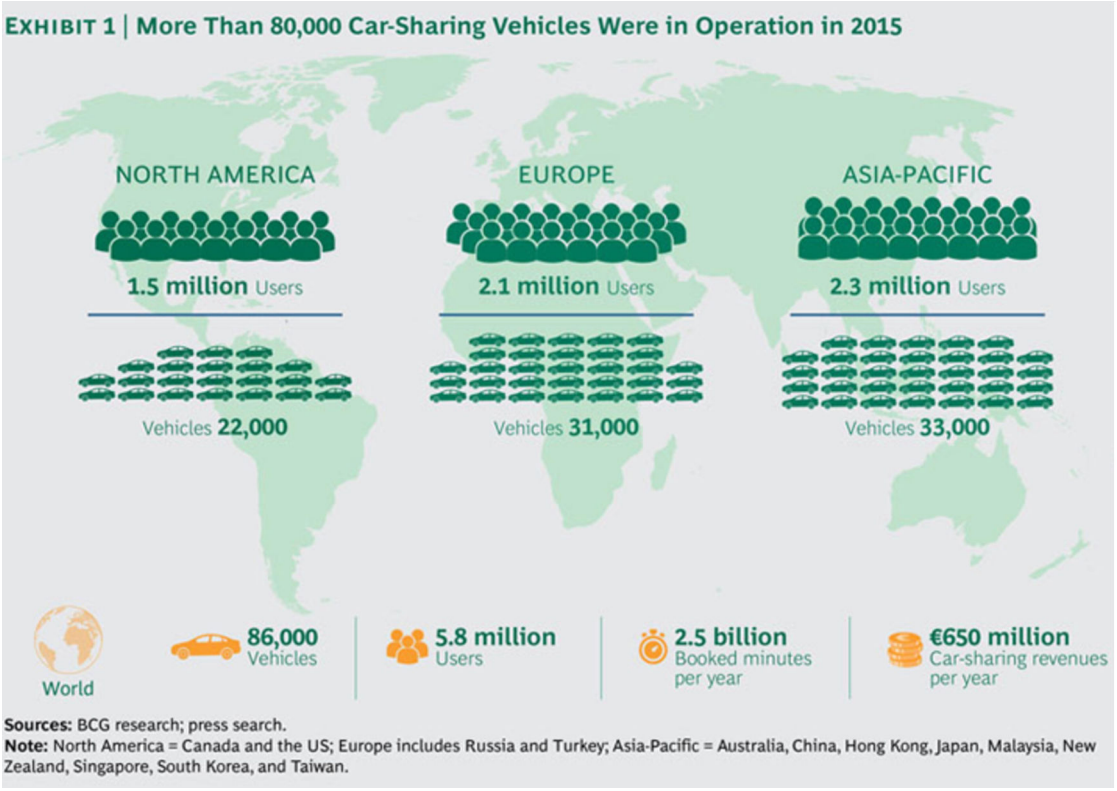


FIGURE 2 – *L'auto-partage dans le monde en 2015. Source image : bcg.perspectives (2017)*

Types d'auto-partage. En ce qui concerne l'auto-partage exploité par les autorités locales ou par des entreprises privées, nous distinguons deux catégories :

1. L'auto-partage *en boucle* : cette forme de partage de voiture est la plus classique car à ses débuts, l'auto-partage requiert un système de boucle où les utilisateurs sont obligés de restituer leur voiture à leur lieu de départ (système tendant à disparaître pour son manque de souplesse).
2. L'auto-partage *à un seul sens* : il est différent du système en boucle car l'utilisateur peut restituer sa voiture n'importe où et pas nécessairement au point de départ. Ce système présente des avantages pour les clients, car la restitution de la voiture est beaucoup moins contraignante et peut être effectuée dans un lieu proche de la destination du client. Dans notre travail, nous nous intéressons à l'auto-partage *à un seul sens*, et à l'intérieur même de ce dernier, on peut trouver plusieurs modes d'usages :
   — Auto-partage *à un seul sens* avec stations : ce système repose sur un ensemble de stations avec parking de voitures et bornes de rechargement en cas de voitures électriques ; les utilisateurs sont obligés de prendre et restituer les voitures dans les stations ; la station de départ n'est pas nécessairement celle de la restitution.
   — Auto-partage *à un seul sens* sans station (*Free-floating*) : les utilisateurs localisent des véhicules situés dans leur voisinage, grâce à des applications informatiques de géolocalisation et ils peuvent déposer le véhicule à n'importe quelle place de stationnement dans la rue.
   — Auto-partage *à un seul sens* avec réservation : dans ce cas de figure, les clients doivent réserver à l'avance un véhicule et éventuellement une place de restitution, même si la réservation peut être contraignante pour le client ; ce système présente l'avantage de garantir la disponibilité d'un véhicule ce qui le rend plus fiable.
   — Auto-partage *à un seul sens* sans réservation : comme son nom l'indique, l'utilisateur peut se servir et utiliser n'importe quel véhicule disponible à proximité dans une station ou dans la rue (cas du *free-float*) sans avoir le besoin de réserver ; par contre l'utilisateur n'est pas sûr de trouver un véhicule aux lieu et moment voulus.

Problématique de la thèse. Bien que l'auto-partage *à un seul sens* ait plusieurs avantages, il génère un problème majeur de redistribution (relocalisation) des véhicules. En effet, le fait que l'utilisateur ne ramène pas le véhicule à sa localisation d'origine peut générer un déséquilibre dans la distribution des véhicules à travers la ville. Par exemple, dans l'auto-partage *à un seul sens* avec stations, on se retrouve avec des stations pleines dans les centres d'agglomération, ce qui empêche de nouveaux arrivants de se garer, et d'autres stations vides dans les

périphéries, où il n'est plus possible de servir de nouvelles requêtes de clients. Ce problème est généré par les flux urbains non équilibrés et généralement plus intenses en direction des centres villes le matin et à contre-sens en fin de journée. Les opérateurs d'auto-partage essayent de remédier à ce déséquilibre par l'introduction d'agents mobiles qui déplacent les véhicules entre les stations afin d'équilibrer leur répartition. Plusieurs problèmes d'optimisation découlent de cette manière de procéder avec des agents. En effet, il s'agit de trouver l'ensemble des relocalisations de voitures qui maximise le nombre de clients satisfaits, tout en minimisant le coût de la relocalisation et donc la somme des distances parcourues par les agents ainsi que leur nombre. D'autres objectifs peuvent se greffer à cette problématique, comme le nombre (à minimiser) de voitures à utiliser.

Objectif de la thèse. L'objectif principal de nos travaux est l'étude d'un problème d'optimisation assez nouveau qui est la réservation de voitures, et la relocalisation (l'équilibrage de la distribution des voitures) dans un système d'auto-partage *à un seul sens* avec des contraintes de recharge électrique qui ajoutent de la difficulté au problème. L'étude de cette problématique comprend plusieurs sous-objectifs, parmi lesquels on peut citer :

— Proposer des modélisations pour le problème statique qui permettent de prendre en compte les différentes contraintes du problème ;

— Proposer des heuristiques pour résoudre le problème et donc étudier les facteurs influant sur l'optimisation du système ;

— Estimer l'apport que peut avoir la relocalisation et son optimisation sur le fonctionnement du système ;

— Estimer le nombre de voitures et d'agents nécessaire pour le bon fonctionnement d'un tel système ;

— Calculer le taux de partage possible des voitures avec une gestion optimisée ;

— Proposer des outils qui permettent la gestion opérationnelle optimisée du service avec des réservations dynamiques.

Méthodologie et contexte de travail. Pour atteindre nos objectifs, nous nous sommes bien sûr appuyés sur les travaux déjà publiés sur l'optimisation de service d'auto-partage, mais aussi inspirés des approches existantes pour la résolution d'autres problèmes de transport qui peuvent présenter des similitudes. Comme pour tout problème d'optimisation, nous allons proposer une ou plusieurs modélisations, ainsi que des méthodes de résolution associées. Nous validerons ensuite nos différentes approches de résolution avec des simulations. La

thèse a été abordée avec une vision pluridisciplinaire. D'une part, nous avons eu recours à des méthodes d'analyse des territoires et aux Systèmes Informatiques Géographiques pour faire des simulations sur des instances générées à partir du monde réel. Nous nous sommes intéressés à l'étude de ce nouveau mode de transport qui est l'auto-partage et les paramètres clefs qui peuvent influencer son bon fonctionnement. Le cœur de la thèse reste toutefois dans le domaine de la recherche opérationnelle appliquée au transport partagé, les questions essentielles qui y sont abordées concernent l'optimisation.

Organisation de la thèse. La thèse est organisée de la manière suivante :

1. Le chapitre 1 se compose de 3 sections et décrit de façon synthétique un état de l'art sur l'optimisation dans l'auto-partage et les problèmes de transport de manière générale.
   - Dans la section 1.2, nous présentons les problèmes d'optimisation et de routage classiques dans le transport et leur complexité.
   - La section 1.3 présente les méthodes de résolution utilisées dans les problèmes de routage de véhicules.
   - La section 1.4 aborde, quant à elle, les problèmes d'optimisation dans les transports urbains avec partage de véhicules comme le vélo partage ou le covoiturage, en complément de l'auto-partage qui est le sujet de cette thèse.
2. Le chapitre 2 est dédié à la modélisation mathématique du problème de relocalisation. Le problème d'optimisation est décomposé en deux sous-problèmes : le *Recharging One way Car Sharing Problem* (*ROCSP*) et le *Employee Scheduling Routing Problem* (*ESRP*). Des modélisations en Programmation Linéaire en Nombres Entiers (PLNE) sont proposées pour les deux sous-problèmes. Ce chapitre permet aussi de présenter les instances d'auto-partage utilisées dans les simulations, ainsi qu'une théorie pour générer des flux de déplacement entre les stations d'auto-partage basée sur un modèle gravitaire.
3. Le chapitre 3 est consacré à la résolution de la variante statique du problème de relocalisation dans l'auto-partage électrique *à un seul sens* avec ses deux sous problèmes : le *ROCSP* et l'*ESRP*. Différentes méthodes de résolution sont proposées pour les problèmes étudiés. Dans un premier temps, nous validons les modèles PLNE proposés dans le chapitre 2 à l'aide du solveur CPLEX. Cette première étape nous permet de constater rapidement que les méthodes exactes se limitent à des instances de taille très réduite et non réalistes. Comme pour la plupart des problèmes d'optimisation de grande taille, nous nous orientons vers des approches de résolution heuristiques.

4. Dans le chapitre 4, le problème de relocalisation dans l'auto-partage électrique *à un seul sens* avec réservation dynamique des usagers est relevé. Nous présentons aussi l'Auto-CoPartage, un nouveau concept, hybridation entre auto-partage et covoiturage.
5. La conclusion générale nous permet de faire un bilan de nos travaux et de proposer quelques perspectives de recherche.

Chapitre 1

État de l'art

Sommaire

1.1 Introduction

Dans ce chapitre, nous présentons les problèmes d'optimisation dans le transport à travers les variantes les plus classiques et connues, qui se rapprochent, en termes de contraintes et d'objectifs, des problèmes d'auto-partage qui seront traités dans cette thèse. Nous verrons aussi les méthodes de résolution utilisées dans la littérature pour ces problèmes d'optimisation. Pour finir, nous abordons un ensemble de problèmes de transports urbains où les véhicules sont partagés entre plusieurs utilisateurs.

1.2 Problèmes de routage classiques dans le transport

Avant de présenter les variantes classiques de problème d'optimisation dans le transport, nous commençons par introduire la notion de complexité des problèmes d'optimisation de manière générale.

1.2.1 Notions de complexité

La complexité temporelle d'un algorithme est le nombre d'opérations élémentaires comme les opérations arithmétiques et les comparaisons. Cette complexité est déterminée en fonction de la taille des données en entrée de l'algorithme et calculée au pire des cas. À partir de cette définition de la complexité algorithmique, on peut classer les problèmes de décision, dont ceux d'optimisation en :

— Problèmes dans la classe P : le meilleur algorithme de résolution qui existe est de complexité polynomiale.
— Problèmes dans la classe NP : le meilleur algorithme de vérification des solutions qui existe est de complexité polynomiale.
— Problèmes dans la classe Exp : le meilleur algorithme de résolution qui existe est de complexité exponentielle.
— Problèmes dans la classe R : le meilleur algorithme de résolution qui existe est de complexité finie.

On dit qu'un problème est *NP-difficile*, s'il est au moins aussi difficile que tous les problèmes dans NP. Cependant, les problèmes *NP-difficiles* ne sont pas forcément dans NP et peuvent être dans une classe plus difficile comme EXP

ou *R*. L'ensemble des problèmes *NP-difficiles* dans *NP* sont classés *NP-complets* et ce sont les problèmes qui sont dans l'intersection des deux classes *NP* et *NP-difficile*. Même chose pour *EXP-complet* et *R-complet* qui représentent *EXP* ∩ *EXP-difficile* et *R* ∩ *R-difficile* respectivement. Pour plus de détails voir Papadimitriou (2003).

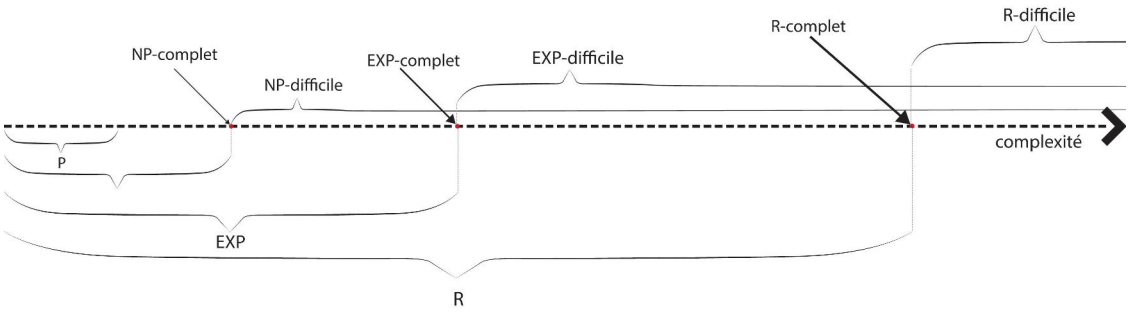


FIGURE 1.1 – *Illustration des différentes classes de problèmes selon leurs complexités*

1.2.2 Problèmes de routage

Le routage consiste à sélectionner un chemin entre deux points sur un réseau, de manière générale, le routage s'effectue dans de nombreux types de réseaux, tels que le réseaux routier, les réseau téléphoniques et les réseaux informatiques.

Problème de plus court chemin

Un problème de plus court chemin consiste en le calcul d'un chemin entre deux sommets d'un graphe qui minimise ou maximise une fonction objectif. Chaque arc ou arête est associé à un coût. Le chemin le plus court entre deux nœuds donnés minimise la somme des coûts des arcs traversés. Il existe de nombreuses variantes de ce problème. La variante de base est de complexité polynomiale, mais lorsqu'on ajoute des contraintes supplémentaires, comme des contraintes de ressource ou des fenêtres de temps, le problème peut devenir *NP-difficile*.

Problème du voyageur de commerce

Le problème du voyageur de commerce se résume dans la recherche d'un trajet minimal, permettant à un voyageur de visiter *n* villes. C'est également un

problème de plus court chemin dans un graphe, mais avec la contrainte supplémentaire de visiter tous les sommets du graphe. Le problème du voyageur de commerce est un problème *NP-complet*.

Problèmes de tournées de véhicules *VRP* pour *Vehicle Routing Problem*

Le problème de tournées de véhicules *VRP* est une généralisation de celui du voyageur de commerce à plusieurs véhicules, où il s'agit d'affecter des commandes de de clients à des véhicules et construire la tournée de chacun de ces véhicules vers les sites de livraison des clients, en satisfaisant certaines contraintes et en optimisant un objectif donné. Plus précisément, connaissant les demandes et la localisation des clients, une flotte de véhicules partant d'un dépôt doit desservir ces clients en un seul passage et retourner à un dépôt. Le *VRP* est *NP-difficile*, la démonstration de la complexité théorique du *VRP* a été formalisée par Lenstra et Kan (1981). Dans la pratique, le *VRP* est souvent traité simultanément avec des problèmes de planification de production ce qui complexifie le problème car l'objectif est de synchroniser les deux problèmes (planification et transport) et de construire une solution globale, par exemple, Yantong et al. (2018) proposent un modèle PLNE et des méthodes de résolution pour un problème de planification de systèmes de logistique alimentaire où un des objectifs est de réduire la durée de stockage des aliments.

Variantes du *VRP* Une grande variété de problèmes de routage de véhicules a été étudiée dans la littérature. Une classification de ces variantes a été donnée par Berbeglia et al. (2007). Dans cet état de l'art, nous nous limiterons aux variantes les plus connues, en précisant pour chacune d'elle l'objectif et le critère de faisabilité d'une solution.

Le *CVRP* : La version de base du *VRP* est le problème de tournées de véhicules avec contraintes de capacité ou *CVRP* pour *Capacitated Vehicle Routing Problem*. Dans ce cas, seules les contraintes de capacité pour les véhicules sont imposées. L'objectif consiste à minimiser le coût total de la desserte des clients.

Le *VRPPD* : Le problème de ramassage et de livraison ou *VRPPD* pour *Vehicle Routing Problem with Pick-up and Delivery* est un *VRP* pour lequel la possibilité que les clients renvoient certains produits est envisagée. Il est donc nécessaire de tenir compte du fait que les quantités de marchandises que les clients rapportent ne dépassent pas la capacité du véhicule chargé de les ramasser. Cette contrainte rend le problème de planification plus difficile et peut conduire

à une mauvaise utilisation de la capacité des véhicules, à une augmentation des distances parcourues ou à un besoin croissant de véhicules. L'objectif de ce problème est de réduire la flotte de véhicules et la somme des temps de parcours, avec la condition que le véhicule doit avoir une capacité suffisante pour transporter les produits à livrer et ceux à ramasser chez les clients pour leur retour au dépôt.

Le *VRPTW* : Le *VRPTW* pour *Vehicle Routing Problem with Time Windows* est une extension du *VRP* avec la contrainte supplémentaire qu'une fenêtre de temps est associée à chaque client, définissant un intervalle de temps durant lequel le client doit être servi. L'objectif est de réduire la flotte de véhicules utilisée et la somme des temps de parcours de ces véhicules ainsi que les temps d'attente nécessaires pour satisfaire tous les clients dans leurs fenêtres de temps. Les *Time Windows* peuvent être greffés à toutes les variantes du *VRP* en ajoutant des fenêtres de temps aux commandes des clients.

Le *DARP* : Le *Dial-a-Ride Problem (DARP)* consiste à concevoir des itinéraires et des horaires de véhicules pour un nombre d'utilisateurs qui émettent des requêtes de ramassage et de dépôt entre une origine et une destination avec des horaires de prise en charge. Du point de vue de la modélisation, le *DARP* généralise un certain nombre de *VRP* comme le *VRPPD* avec *Time Windows* (*VRPPDTW*) où il s'agit aussi de gérer des requêtes de transport entre deux points, avec des fenêtres de temps. Le point de vue humain fait que le *DARP* est différent de la plupart des problèmes de routage. Lors du transport de passagers, un compromis doit être choisi entre la qualité de service pour l'utilisateur et la minimisation des coûts d'exploitation. De plus, la capacité du véhicule est normalement contraignante dans le *DARP* car c'est généralement des petits bus qui transportent une dizaine de personnes au maximum alors que dans le *VRPPDTW* c'est plus souvent des petits objets transportés avec des camions.

Le *MDVRP* : Une entreprise peut avoir plusieurs dépôts à partir desquels elle peut servir ses clients. Si les clients sont regroupés autour des dépôts, alors le problème de la distribution devrait être modélisé comme un ensemble de *VRP* indépendant. Toutefois, si les clients et les dépôts ne sont pas répartis géographiquement de manière régulière, alors un *MDVRP* pour *Multi Depot Vehicle Routing Problem* doit être résolu.

Le *FSMVRP* : Le *FSMVRP* pour *Fleet Size and Mix Vehicle Routing Problem* généralise le problème classique de tournées de véhicules, en supposant que la flotte consiste en un ensemble hétérogène et illimité de véhicules. A chaque

type de véhicule est associé un coût, dépendant de la distance et de sa capacité de chargement. L'objectif est de déterminer la meilleure composition de véhicules ainsi que l'ensemble des routes qui réduisent au minimum les frais de déplacement.

Le problème d'auto-partage traité dans cette thèse présente des similitudes avec quelques variantes du *VRP*, car en effet, il s'agira de router les voitures pour servir des utilisateurs qui effectuent des requêtes de transport d'un point A vers un point B avec les horaires correspondants. Par exemple, dans le *DARP* on retrouve les requêtes clients avec horaires correspondants, même si, dans le *DARP*, les clients peuvent être regroupés dans le même véhicule contrairement à l'auto-partage.

1.2.3 Problèmes de flots dans un réseau de transport

Contrairement aux problèmes de routage, les problèmes de flots ne permettent pas de tracer des chemins dans les réseaux mais se limitent au calcul de la quantité d'objets à envoyer d'un point du réseau vers les points suivants. L'information sur les objets transportés est donc agrégée et ne permet pas le suivi des objets individuellement. La problématique de réseau de transport ou de flot correspond au besoin de livrer des objets d'une ville source à une autre destination en sachant qu'il est possible de les livrer en passant par diverses liaisons intermédiaires, de ville en ville, mais que chaque liaison a une capacité limitée. On peut imaginer l'exemple d'un problème de transport de marchandises avec des camions allant d'une ville où se trouve un port vers une ville intérieure en respectant la contrainte de non congestion des routes et donc un nombre de camions maximum par route selon la capacité de la route. La question à laquelle il faut répondre dans cette problématique est de connaître le nombre de camions à envoyer sur chaque route, en cherchant éventuellement à minimiser un coût, comme par exemple celui du péage des autoroutes. Pour résoudre cette question, les auteurs spécialisés dans la théorie des graphes utilisent ce que l'on appelle le *graphe de réseaux de flux* pour modéliser différents problèmes de transport de flux. Dans cette thèse, nous aurons recours à ce type de modèles.

Un graphe est une représentation symbolique d'un réseau. Il s'agit d'une abstraction de la réalité, de sorte à permettre sa modélisation. Soit $G = (V, E)$ un graphe orienté avec deux sommets spéciaux : le sommet source s et le sommet destination t. Chaque sommet représente une ville où il est possible d'envoyer ou de recevoir des marchandises. Un arc (u, v) du graphe signifie qu'il y a une route qui lie directement de u à v. Chaque arc entre deux sommets (villes par exemple) a une capacité associée, toujours finie, représentant la quantité

maximale possible de camions sur la route représentée par l'arc en question, la Figure 1.2 est un exemple d'un graphe de flux réseau.

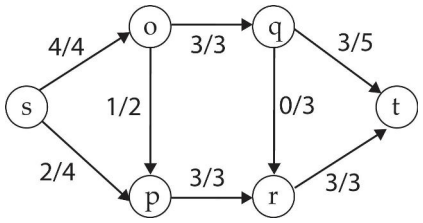


FIGURE 1.2 – *Un exemple de graphe de flot avec un flot maximum. La source est s, et le puits est t. Les nombres indiquent le flot et la capacité.*

Une des approches pour résoudre les problèmes de flot est la Programmation Linéaire. Le modèle PLNE *MaxFlot* est le plus utilisé et modélise un problème de flot maximum avec $f_{(u,v)}$ le flot qui passe dans l'arc (u,v) et $k_{(u,v)}$ la capacité de l'arc.

$$\max F \qquad (MaxFlot)$$

$$\sum_{u \in V} f_{(u,v)} - \sum_{u \in V} f_{(v,u)} = \begin{cases} F & \forall v = s \\ 0 & \forall v \neq s,t \\ -F & \forall v = t \end{cases}$$

$$0 \leq f_{(u,v)} \leq k_{(u,v)}$$

Les problèmes de flot maximum sont de complexité polynomiale et résolus en temps polynomial par des modèles de *PL*, parce que leur matrice de contraintes est totalement unimodulaire (cf. Hoffman et Kruskal (2010)), ce qui donne des solutions entières au modèle de *PL*, même si on ne restreint pas les valeurs des variables à des entiers. Il existe d'autres variantes du problème de flot maximum, certaines sont facilement réductibles au problème de flot maximum standard et sont donc résolues en utilisant les mêmes modèles comme la variante du flot de coût minimum. D'autres variantes sont de classe *NP*-difficile comme le problème de multi-flots où au minimum deux flots distincts circulent sur le même réseau et se partagent donc les capacités totales sur chaque arc en plus des contraintes de capacité spécifiques à chaque flot.

Après avoir abordé les problématiques de transport classique dans cette section, la section suivante nous permettra de présenter différentes méthodes de résolution exactes ou approchées, avec lesquelles ont été abordés les problèmes de routage de véhicules dans la littérature.

1.3 Méthodes de résolution appliquées aux problèmes de routage dans le transport

Ces cinquante dernières années ont connu une émergence très forte de nombreuses méthodes de résolution des problèmes d'optimisation combinatoire. Dans cette sous-section, nous avons choisi de présenter quelques méthodes en détaillant certaines d'entre elles. Certaines méthodes sont adaptées aux problèmes de grande taille, d'autres ne traitent que des problèmes de petite taille. Par ailleurs, certains algorithmes sont construits pour être rapides au détriment de la qualité de la solution, alors que d'autres recherchent l'optimalité. On distingue deux types de méthodes :

— les méthodes exactes comme la *programmation dynamique*, le *Branch and Bound*, le *Branch and Cut* et la *programmation mathématique*.

— les méthodes heuristiques qui peuvent être constructives ou améliora-trices.

1.3.1 Méthodes exactes

Le but des méthodes exactes est de trouver, en un temps le plus court possible, la solution optimale du problème. Cela représente un véritable défi pour tous les problèmes de l'optimisation combinatoire classés *NP*-difficiles. La résolution du *VRP* par les méthodes exactes a intéressé plusieurs auteurs de la communauté de l'optimisation.

Programmation linière et le Solveur CPLEX

Un large éventail de problèmes d'optimisation peut être modélisé et résolu avec la Programmation Linéaire (*PL*), un programme linéaire étant constitué d'un ensemble de variables de décisions, une fonction objectif et un ensemble d'inégalités modélisant les contraintes. Un Programme Linéaire (*PL*) est un problème d'optimisation de la forme :

$$(\min or \max) Z = \sum_{j=1}^{n} c_j \cdot x_j \qquad \text{(Objectif)}$$

$$\sum_{j=1}^{n} a_{ij} \cdot x_j (\leq or = or \geq) b_i \qquad \forall i = 1, 2, ..., m \qquad \text{(Contrainte)}$$

$$x_j \geq 0 \qquad \forall j = 1, 2, ..., n \qquad \text{(Contrainte)}$$

On parle de programmation linéaire, dans le sens où la fonction objectif et l'ensemble des contraintes doivent être formulées avec des équations linéaires. Certains problèmes peuvent être formulés avec uniquement des variables continues et ce type de problèmes est relativement facile à résoudre. Cependant, dans de nombreux cas de la vie réelle, on est obligé d'utiliser des variables entières, ou des variables binaires, encore plus restreintes. L'utilisation de variables entières ajoute de la difficulté au problème de programmation linéaire, en raison du manque de continuité et on parle de Programmation Linéaire en Nombre Entiers (*PLNE*).

Un des solveurs les plus utilisés pour la *PLNE* est *CPLEX* Optimizer d'IBM ILOG, qui fournit des solveurs de programmation mathématique flexibles et performants pour la programmation linéaire et la programmation en nombres entiers mixtes. Ces solveurs comprennent un algorithme parallèle distribué pour la programmation mixte d'entiers, afin de tirer parti de plusieurs processeurs pour résoudre des problèmes difficiles. La version *CPLEX 12.2* est utilisée dans les simulations de nos travaux.

Branch and Bound et *Branch and Cut*

Parmi les méthodes de résolution exactes appliquées au *VRP*, celle du *Branch and Cut*, version améliorée du *Branch and Bound*, est très utilisée. Le principe de base de cette méthode est de séparer l'espace des solutions en plusieurs parties (phase de branchement), puis d'évaluer pour chacune de ces branches, la borne inférieure qui servira à couper les branches qui ne mènent pas à une solution optimale. Les techniques de branchement et de calcul de bornes inférieures sont très nombreuses, nous ne donnerons ici que quelques références bibliographiques :

— Les travaux de Kolen et al. (1987) sont parmi les premiers à aborder un *VRP* avec fenêtres de temps à l'aide d'un *Branch and Bound*.

— Bard et al. (2002) ont proposé un *Branch and Cut* pour résoudre le problème du *VRP* avec fenêtres de temps (*VRPTW*) et un nombre minimum de véhicules. La flotte est homogène et se situe à un dépôt commun. Leurs

travaux ont porté sur la résolution de problèmes de petites tailles de 50 à 100 clients. Ils ont utilisé des techniques heuristiques de coupe relativement efficaces pour pouvoir atteindre l'optimalité dans un grand nombre de problèmes traités.

— Letchford et al. (2007) ont proposé un *Branch and Cut* qui s'adapte aux contraintes spécifiques de l'*Open VRP*, qui est une variante du *VRP* pour laquelle un véhicule n'est pas obligé de rentrer au dépôt à la fin de sa tournée.

Branch and Price et Génération de Colonnes (GC)

Branch and Price est une méthode de *Branch and Bound* dans laquelle, à chaque nœud de l'arbre de recherche, on applique l'heuristique de la génération de colonnes, dont le principe repose sur le fait que toutes les variables d'un programme linéaire ne sont pas forcément utiles pour atteindre la solution optimale. L'objectif est alors de résoudre un problème réduit en prenant en compte un ensemble limité de variables. Le problème initial est appelé *problème maître* (*problème maître*) et contient l'ensemble de toutes les variables possible P, le problème avec un nombre réduit de variables est appelé *problème maître restreint* (*Primal problème maître restreint*) et contient un sous ensemble de variables $P_k \subset P$ de telle sorte que ce sous ensemble permet de construire une solution initiale faisable.

$$
\begin{aligned}
&\min \quad c^T \cdot x && \text{(\textit{problème maître})}\\
&A \cdot x \geq b\\
&x \geq 0\\
\\
&\min \quad c_k^T \cdot x && \text{(\textit{Primal problème maître restreint})}\\
&A_k \cdot x \geq b_k\\
&x \geq 0\\
\\
&\max \quad b_k^T \cdot \pi && \text{(\textit{Dual problème maître restreint})}\\
&A_k^T \cdot \pi \leq c_k\\
&\pi \geq 0
\end{aligned}
\tag{1.1}
$$

Le problème restreint est plus facile à résoudre, toutefois, si l'ensemble de ses variables ne contient pas celles utilisées par la solution optimale du problème maître, nous n'obtiendrons pas l'optimalité. Pour atteindre la solution optimale du problème maître, il faut ajouter au problème restreint des variables pouvant être utilisées pour améliorer la solution. Le problème consistant à trouver la meilleure variable à ajouter au problème maître restreint est appelé *problème esclave* ou *problème du pricing* dans la littérature. Son but est de trouver la variable (ou colonne) de coût réduit minimum (*i.e.* la plus prometteuse pour améliorer la solution du *problème maître*). Les variables duales du *Dual problème maître restreint* sont utilisées pour calculer le coût réduit minimum d'une variable $\bar{c} = \min(c_j - \pi \cdot A_j)$. Soit X_k la solution optimale (pas forcément entière) du *problème maître restreint* alors :

— si $\bar{c} < 0$: passer à l'itération suivante de la GC en rajoutant à P_k la colonne j correspondant à $\bar{c}$.

— sinon : X_k est aussi la solution optimale (pas forcément entière) du *problème maître* et donc arrêter la GC.

La génération de colonne basée sur la décomposition de *Dantzig-Wolfe* (Vanderbeck, 2000) est une méthode très utilisée pour résoudre le *VRP*. En effet, la décomposition permet de reformuler le modèle PLNE du problème de telle sorte que les colonnes(variables) générées sont des variables complexes qui prennent en charge une partie des contraintes du problème, permettant donc d'alléger le problème maître et déléguer la gestion de ces contraintes au problème du *Pricing (esclave)*. Généralement, dans le *VRP* les variables utilisées représentent des chemins au lieu de simples arcs dans la formulation originale du problème, ce qui permet de déléguer les contraintes de continuité de chemins au problème du *Pricing*. Les fenêtres de temps sont aussi gérées dans le problème du *Pricing* en résolvant le *The shortest path problem with resource constraints (SPPRC)* introduit par Desrosiers et al. (1986) où la ressource représente le temps.

L'heuristique de génération de colonnes peut produire des solutions non entières, ce qui fait que, dans la méthode du *Branch and Price*, des contraintes appelées dans la littérature *règles de branchement*, sont ajoutées dans le but de forcer la génération de colonnes à trouver des solutions entières. Il existe deux types de règles de branchement, l'un garantit théoriquement la convergence vers une solution entière et l'autre ne la garantit pas, mais peut s'avérer efficace dans le cas pratique. C'est le fait de brancher qui permet de passer d'un nœud à un autre de l'arbre de recherche comme le montre le diagramme de la figure 1.3.

Qureshi et al. (2009) utilisent une méthode de génération de colonnes pour résoudre un *VRP* avec des fenêtres de temps souples. Dans ce problème, les contraintes de fenêtres de temps peuvent être violées, *modulo* une pénalité qui

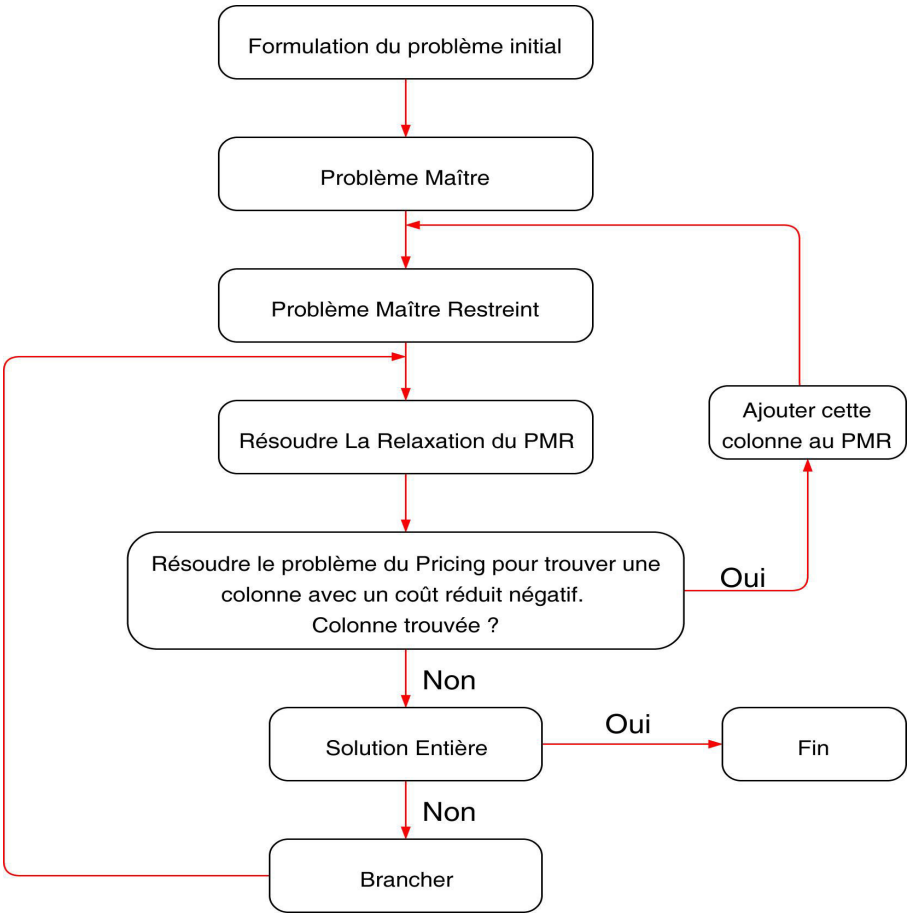


FIGURE 1.3 – *Algorithme du Branch and Price*

affecte la fonction objectif. La technique de génération de colonnes a été souvent hybridée avec des heuristiques. On peut à ce titre se référer au travail de Xu et al. (2003), qui ont combiné la méthode de génération de colonnes et une métaheuristique pour améliorer les performances d'un *VRPPD*. Fukasawa et al. (2006) ont utilisé un *Branch and Cut and Price* pour résoudre un *VRP* avec des contraintes de rechargement à deux dimensions.

Les méthodes citées dans cette sous-section et les méthodes exactes de façon générale, sont très performantes pour la résolution de problèmes de petite taille, voir des problèmes de taille moyenne. Malheureusement, elles deviennent quasiment inutilisables face aux problèmes de grande taille.

1.3.2 Méthodes heuristiques

Comme pour tous les problèmes *NP*-difficiles, la recherche d'heuristiques performantes est une des préoccupations des spécialistes du domaine. En effet, le principe d'une méthode heuristique est de trouver en un temps polynomial la meilleure solution possible parmi les solutions réalisables. Dans cette recherche, les objectifs poursuivis sont parfois différents. Certains privilégient la rapidité de calcul au détriment de l'optimalité. D'autres s'intéressent plus à la quasi-optimalité du résultat au détriment du temps. Nous proposons de décrire certaines heuristiques connues à ce jour, en précisant leur intérêt.

Heuristiques constructives

Le principe de ces méthodes est de construire une solution du problème à partir des données initiales. Les méthodes proposées dans ce contexte sont nombreuses, mais le principe est généralement inspiré de l'une de ces heuristiques :

— Heuristique de Clarke et Wright : en 1964, Clarke et Wright (1964) ont publié un algorithme pour la résolution du problème de tournées de véhicules, basé sur un concept dit de *gain*. Partant d'une solution initiale où chaque client est desservi séparément, l'algorithme fusionne les deux tournées qui maximisent le gain en distance, de telle sorte que la tournée résultante respecte les contraintes liées au problème. Ce processus est itéré jusqu'à ce qu'il n'ait plus de gain possible. Cette méthode a été améliorée par Gaskell (1967) et Yellow (1970) afin d'avoir moins de tournées «circulaires», en pondérant la distance au dépôt dans le calcul du gain de distances.

— L'heuristique *Sweep* accorde la priorité à la construction des tournées avant leurs regroupements (*Route First-Cluster Second*). Différentes variantes de

cette heuristique sont présentées par Gillett et Miller (1974), Wren et Holliday (1972) et Renaud et Boctor (2002).

Le principe de l'heuristique *Sweep* peut être résumé en trois étapes :

— Etape 1 : initialisation de la tournée
  - Choisir un véhicule non utilisé

— Etape 2 : construction de la tournée
  - Choisir un client non desservi présentant le plus petit angle d'écart par rapport au dernier client de la tournée courante,
  - Affecter le client à la tournée pour autant que la capacité ou la longueur de parcours maximale ne soit pas dépassée,
  - S'il reste des clients non routés, alors aller à l'étape 1.

— Etape 3 : optimisation de la tournée
  - Optimiser l'itinéraire d'un véhicule séparément en résolvant le problème de voyageur de commerce de façon exacte ou approchée,
  - Appliquer l'heuristique Pétale (voir ci-dessous)

Une extension naturelle de l'algorithme *Sweep* consiste à générer plusieurs tournées, appelées *pétales* (cf. Ryan et al. (1993)), puis d'établir la sélection définitive des tournées les plus prometteuses. Renaud et al. (1996) ont proposé à ce niveau une heuristique *Sweep* qui donne de bons résultats.

Heuristiques amélioratrices

Le principe de ces méthodes n'est plus de construire une solution à partir des données initiales du problème, mais de modifier une solution admissible, en vue d'améliorer la valeur de la fonction objectif. À chaque étape de ce processus, la solution choisie n'améliore pas forcément la valeur de la fonction objectif, ce qui permet de sortir de *minima* locaux.

La méthode *tabou*

La méthode tabou, élaborée par Glover (1989) et Glover (1990) est basée sur les notions de voisinage de solutions et de mouvements interdits, ces derniers ont pour but d'interdire les mouvements vers des solutions déjà visitées et donc d'éviter de converger vers un optimal local. Le voisinage d'une solution est un ensemble de solutions que l'on peut atteindre à partir de la première, en effectuant un mouvement prédéfini. Il existe plusieurs façons de définir ces mouvements. On peut par exemple permuter deux clients dans une tournée. Dans

Ceschia et al. (2011), les auteurs présentent une recherche tabou avec hybridation de plusieurs types de mouvements spécifiques aux *VRP* avec fenêtres de temps. Chaque mouvement est ajouté à une liste de taille fixée (liste tabou) qui contient les mouvements interdits. L'algorithme suit les étapes et les conditions suivantes :

— 1. Calcul de la solution initiale, solution réalisable avec une méthode constructive ;

— 2. Recherche du meilleur mouvement pour obtenir la meilleure solution voisine, sans utiliser les mouvements placés dans la liste tabou ;

— 3. Mise à jour de la liste en stockant le mouvement dans la liste tabou, en enlevant le mouvement le plus ancien de la liste ;

— 4. Application d'un test d'arrêt, c'est à dire recommencer la recherche de solution jusqu'à vérifier une condition d'arrêt (nombre limité de mouvements, temps de calcul, etc.).

Le recuit simulé Comme la méthode tabou, le recuit simulé est l'une des approches les plus utilisées pour la résolution des problèmes d'optimisation combinatoire. Basée sur un algorithme de simulation de recuit de métaux, la méthode du recuit simulée s'inspire des modèles de la physique statistique. Le principe de voisinage est identique à celui de la méthode tabou. Seules changent la sélection de la solution suivante dans le processus itératif et la condition d'arrêt. Le fait d'affecter aux solutions du voisinage une probabilité de sélection permet de ne pas autoriser la fonction objectif à décroître strictement et à s'enfermer dans un minimum local comme dans une méthode de descente. Comme la méthode tabou, cette technique permet de sélectionner des solutions voisines en faisant augmenter la fonction objectif, avec une certaine probabilité. Tavakkoli-Moghaddam et al. (2006) utilisent un recuit simulé pour résoudre le *CVRP*.

Les algorithmes génétiques

Les algorithmes génétiques sont inspirés de la théorie de l'évolution humaine de Darwin et datent des travaux de Rechenberg (1973). Contrairement aux algorithmes de recherche locale, tels que le recuit simulé ou la recherche tabou qui manipulent une seule solution réalisable, les algorithmes génétiques considèrent une *population* de solutions réalisables ou non (selon la méthode de génération utilisée) appelées *individus*. Le but de la méthode est de faire évoluer la population en effectuant des mutations, des croisements et une sélection des individus. La mutation est une opération unaire qui modifie la structure

d'un individu, semblable à la mutation génétique. Le croisement est une opération binaire qui, à partir de deux individus, produit deux nouveaux individus, comme le cross-over pour le croisement des gènes. La phase de sélection consiste à supprimer les individus les moins forts, c'est à dire les solutions les moins intéressantes (ayant obtenu les moins bons scores par rapport à un critère donné).

L'idée de base derrière cette méthode est que travailler avec une population permet d'identifier et d'explorer les propriétés communes aux bonnes solutions. Des améliorations de cette méthode sont possibles. On peut par exemple utiliser des méthodes amélioratrices (tabou, recuit simulé) pour faire évoluer les individus après la phase de mutation et de croisement. Les problèmes liés à cette méthode sont extrêmement nombreux. Le premier est bien entendu la taille de la population et sa création. Considérer une grande population ralentirait considérablement la phase de mutation et de croisement alors qu'une taille réduite de la population nuirait à la diversification qui est un des atouts de cette méthode. Le principe des algorithmes génétiques suit les étapes suivantes :

— 1. Calcul de la population initiale : il s'agit de construire une population de solutions réalisables en utilisant des méthodes constructives ;
— 2. Calcul des nouveaux individus : le but est d'accroître la population en ajoutant des individus issus de mutations ou de croisements d'individus de la génération précédente ;
— 3. Sélection des individus : sélectionner les meilleurs individus pour revenir à la taille de la population initiale ;
— 4. Condition d'arrêt : recommencer les croisements et les mutations jusqu'à vérifier la condition d'arrêt (nombre limité de générations, temps de calcul).

Dans le paragraphe précédent, nous avons présenté de façon très générale les grands principes des algorithmes génétiques. Le lecteur intéressé par l'utilisation des algorithmes génétiques pour la résolution du *VRP* peut se référer aux articles suivants :

— Prins (2004) présente un algorithme génétique pour le *VRP* avec contraintes de capacité dont la particularité est la façon très pratique de coder le chromosome. En effet, une solution du problème revient à la définition d'un ordre total de passages chez les clients sans leur affectation aux différents véhicules. Ce n'est que dans un deuxième temps qu'une fonction *split* permet d'avoir une décomposition optimale en différentes tournées. La modélisation de Prins (2004) entraîne une grande facilité dans l'implémentation des fonctions de croisement et de mutation. En se basant sur cette modélisation, Chang et Chen (2007) élaborent une fonction de *split* qui s'adapte aux contraintes de temps dans le but de résoudre un *VRPTW*.

— Wang et Chen (2013) ont proposé un algorithme génétique parallèle pour résoudre le problème du *VRP* avec fenêtres de temps. Cet algorithme se base sur deux types de populations. Le premier a pour but l'intensification visant à s'approcher de l'optimum. Quant au deuxième, il vise plutôt la diversification afin d'éviter de tomber dans des optimaux locaux.

Dans cette section, nous avons vu les principales méthodes d'optimisation pour les problèmes de routage de véhicules. Dans nos travaux, nous nous inspirons de ces méthodes pour proposer des méthodes exactes et heuristiques aux problématiques d'optimisation dans l'auto-partage. Quant à la section suivante, elle réalise un état de l'art plus spécifique à l'auto-partage et aux nouveaux modes de transport urbain partagés.

1.4 Nouveaux modes de transports urbains avec partage de véhicules entre utilisateurs

Dans cette dernière partie de l'état de l'art, nous commençons par exposer quelques problèmes d'optimisation liés aux nouveaux modes de transport urbain où les véhicules utilisés sont partagés entre plusieurs utilisateurs d'une manière ou d'une autre. Par la suite, nous nous focaliserons sur le cas particulier de l'auto-partage et nous citerons certains des travaux effectués jusqu'ici.

1.4.1 Transport à la demande

Le transport à la demande (TAD) (Elodie Castex, 2007) est un service de transport où il n'existe pas de ligne de bus ou de trajet prédéfini, comme son nom l'indique. Le transport se fait à la demande du client et généralement avec des minibus ou des monospaces pour regrouper les clients qui ont une partie du trajet en commun et des horaires de trajets assez similaires. Ce mode de transport public est généralement utilisé pour les zones de faible demande, où une ligne régulière n'est pas forcement rentable, telles que les zones rurales. Cependant, ces dernières années, ce mode de transport a connu un tel succès qu'on l'utilise même dans les grandes villes couvertes par le transport public classique. Avec le développement technologique (géolocalisation et applications mobiles), on est passé au transport à la demande réalisé avec des petites voitures, où on peut regrouper seulement quelques personnes (si ces personnes sont d'accord pour partager le trajet). Ces services sont devenus en quelque sorte des transports collectifs individualisés. Les transports à la demande posent un problème d'optimisation de type *Dial-A-Ride Problem (DARP)*. Dans le TAD, la qualité du service est un facteur important : en plus de l'objectif de minimiser la somme des distances parcourues, on vise aussi à minimiser les durées de trajets des clients et donc les détours pour chaque client. De nombreux auteurs se sont intéressés à ce problème d'optimisation. Nous pouvons citer par exemple Garaix et al. (2010).

1.4.2 Covoiturage

Le covoiturage est le partage de trajet entre des personnes ayant au moins une partie du trajet en commun. Généralement, le propriétaire de la voiture propose un trajet et un temps ou une distance de détours possibles. Se joignent à lui les personnes intéressées par l'offre. Les personnes qui partagent des trajets se rencontrent et échangent via des sites web ou applications mobiles spécialisées.

Le covoiturage est une solution de transport qui permet d'augmenter les taux d'occupation des véhicules et par conséquent de lutter contre la congestion automobile et réduire les émissions de gaz à effet de serre. Il existe deux types de covoiturage : le covoiturage statique où les réservations sont effectuées avant le début du trajet partagé et le covoiturage dynamique, qui permet d'affecter en temps réel des passagers aux véhicules qui ont déjà entamé le trajet. Le temps réel induit un problème de routage dynamique, où il s'agit d'affecter les utilisateurs aux trajets et minimiser les détours. Aïvodji et al. (2018) proposent un protocole de pré-réservation pour le covoiturage qui permet de faire la correspondance tout en respectant les informations privées des utilisateurs. Agatz et al. (2012) proposent une revue de littérature pour ce problème d'optimisation.

1.4.3 Auto-partage

Dans cette sous-section, nous présentons quelques travaux qui s'intéressent aux problèmes d'optimisation dans l'auto-partage. Au cours de ces dernières années, plusieurs études ont traité des plans de déploiement du service d'auto-partage et de sa gestion. Par exemple, Jorge et Correia (2013) proposent une revue de la littérature à ce sujet. Ces travaux abordent différentes situations pratiques correspondant à différentes contraintes et différents objectifs qui peuvent être, par exemple, un profit à maximiser ou un temps opérationnel total à minimiser. L'idéal serait bien sûr de faire une classification comparative des différentes méthodes proposées en fonction de leurs performances. Hélas, les auteurs ne traitent pas exactement les mêmes problématiques d'auto-partage et n'ont pas exactement les mêmes objectifs. L'ensemble des travaux porte sur différents niveaux de décision stratégiques et opérationnels. Les décisions se déclinent selon l'horizon temporel : à court terme, on parle de décisions opérationnelles ; à long terme, on parle de décisions stratégiques. Le niveau stratégique et le niveau opérationnel abordent différents aspects de l'auto-partage *à un seul sens*. Le niveau stratégique influence le déploiement du service, alors que le niveau opérationnel affecte la gestion quotidienne du service. La frontière entre les deux niveaux n'est pas facile à délimiter, car les décisions ne sont pas complètement stratégiques ou complètement opérationnelles.

Décisions stratégiques

Le niveau de décision stratégique est l'ensemble des décisions qui relèvent de la direction de l'opérateur d'auto-partage et qui ont pour ambition de définir la stratégie de l'opérateur. Ces décisions ont un impact à long terme et ont pour objectif principal d'assurer le développement et la pérennité du service. Par exemple, les décisions concernant le nombre de stations d'auto-partage,

leur localisation et la taille de la flotte des voitures représentent des éléments stratégiques du problème.

Parmi les travaux d'optimisation qui se sont intéressés à ce niveau de décision, nous pouvons citer le modèle de Programmation Linéaire en Nombres Entiers (PLNE) proposé par Correia et Antunes (2012), avec deux niveaux de décision dans le but de maximiser les bénéfices de l'organisation de l'auto-partage *à un seul sens* : la localisation des dépôts et la sélection des tournées. Dans cet article, les auteurs présentent une approche d'optimisation de l'emplacement des stations dans le système d'auto-partage *à un seul sens*, où les problèmes de déséquilibre des stocks de véhicules sont traités. L'approche est basée sur des modèles PLNE, dont l'objectif est de maximiser les profits d'un opérateur d'auto-partage, en considérant tous les revenus et les coûts impliqués. L'utilité pratique de l'approche est illustrée par une étude de cas impliquant la municipalité de Lisbonne, au Portugal. Les résultats obtenus par cette étude ont fourni un aperçu de l'impact de la localisation des dépôts et des systèmes de sélection des trajets sur la rentabilité de tels systèmes. L'une des principales conclusions tirée de cette étude est que la situation de déséquilibre des trajets qui caractérise Lisbonne (et beaucoup d'autres villes du monde) entraînerait de graves pertes financières dans un scénario où toute demande d'auto-partage serait satisfaite, même si on répercutait la perte sur une facture très élevée au client. En effet, pour faire face à la demande, une grande flotte de véhicules serait nécessaire avec un rapport d'environ 22,7 véhicules pour 100 voyages, dont la plupart resteraient inutilisées pendant une grande partie de la journée. Une autre conclusion importante est que les pertes financières pourraient être réduites par des choix appropriés du nombre, l'emplacement et à la taille des dépôts. Des bénéfices positifs pourraient être obtenus si les voyages partagés étaient choisis de manière optimale parmi la demande totale, par le biais de réservation ou en refusant des voyages quand il n'y a pas de véhicule disponible dans les stations.

D'autres auteurs traitent le problème de la détermination de la taille optimale de la flotte. George et Xia (2011) abordent ce problème pour une société de location de véhicules et ont obtenu des résultats analytiques pour sa relation avec la disponibilité des véhicules dans chaque station de location. Les auteurs proposent un modèle de réseau de files d'attente fermé du système, obtenu en visualisant le système du point de vue du véhicule. Cela permet de dériver le comportement asymptotique de la disponibilité du véhicule à une station de location selon la taille de la flotte. L'analyse des déséquilibres a permis de proposer quelques principes de base pour la conception des méthodes d'équilibrage des systèmes. Un modèle PLNE pour un problème d'optimisation maximisant le profit pour déterminer la taille optimale de la flotte a été proposé. La nature à grande échelle des systèmes du monde réel entraîne des difficultés de calcul

pour obtenir la solution exacte, ce qui pousse les auteurs à proposer une formulation approximée qui est plus facile à résoudre.

De leur côté, Luminita et al. (2009) utilisent un algorithme du domaine de la logique floue, basé sur un ensemble de règles pour sélectionner les stations électriques d'auto-partage dans une petite ville. Les critères de classification des stations potentielles sont basés sur l'analyse des préférences des habitants de chaque zone urbaine. Les préférences sont collectées à l'aide d'un nombre restreint de questions avec des règles *IF-THEN* qui alimentent un algorithme de logique floue et qui peut être utilisé dans un système expert et être utile pour les décideurs (urbanistes, autorités locales). Selon les auteurs, l'outil proposé peut être aussi utilisé pour étudier les emplacements des stations de bus ou pour la mise en œuvre d'un service de vélo-partage.

Décisions opérationnelles

Le niveau opérationnel correspond aux décisions prises par la hiérarchie intermédiaire de l'opérateur chargé du transport et concerne la gestion courante du service. Ces décisions impliquent le court ou le moyen terme et ont pour objectif l'optimisation des ressources pour atteindre les objectifs fixés. Ce niveau de décision touche à la relocalisation des voitures.

Pour ce qui est du niveau de décision opérationnel, Jorge et al. (2012) testent un modèle PLNE pour résoudre le problème de relocalisation des voitures. Ce modèle de simulation tient compte de la variabilité de la demande et d'une politique de relocalisation des véhicules et valide les solutions fournies par le modèle PLNE précédent. Les tests ont permis de conclure que ces facteurs influencent significativement le bénéfice de l'entreprise et devraient être pris en compte dans la planification des systèmes d'auto-partage *à un seul sens*. De plus, il est montré que la variabilité de la demande et les opérations de relocalisation ont un impact significatif sur les solutions. En ce qui concerne la variabilité de la demande, la réduction de la demande a un impact négatif sur les solutions, diminuant le profit, même s'il est possible d'économiser de l'argent sur les véhicules et les places de stationnement. Le contraire ne se produit que lorsque la demande est réduite à un niveau tel que certaines stations «disparaissent». Les résultats permettent de conclure que la variabilité de la demande devrait être incluse dans les modèles de planification pour l'auto-partage *à un seul sens* et suggèrent que plusieurs options opérationnelles, telles que les opérations de relocalisation, doivent être étudiées de manière intégrée, puisque ces deux facteurs causent des impacts significatifs sur les solutions.

Kek et al. (2009) présentent un algorithme d'aide à la décision pour les opérateurs d'auto-partage de Singapour. Cet article propose un nouveau système

d'aide à la décision d'optimisation et simulation de tendance pour les opérateurs d'auto-partage afin de déterminer un ensemble de paramètres de personnels et de fonctionnement quasi-optimaux pour le problème de relocalisation. Testés sur un ensemble de données commercialement opérationnelles provenant d'une société d'auto-partage à Singapour, les résultats de la simulation suggèrent que les paramètres recommandés par le système conduisent à une réduction des coûts de personnel de 50% et une réduction du nombre de relocalisations comprises entre 37,1% et 41,1%.

Jorge et al. (2014) comparent les relocalisations optimales calculées avec un modèle PLNE, avec deux politiques de relocalisation simulée. En effet, deux méthodes sont présentées, à savoir un nouveau modèle mathématique pour optimiser les opérations de relocalisation qui maximisent la rentabilité d'un service d'auto-partage et un modèle de simulation pour étudier différentes politiques de relocalisation en temps réel. Les deux méthodes ont été appliquées à des instances du service d'auto-partage de Lisbonne. Les résultats montrent que la relocalisation des véhicules, en utilisant l'une des méthodes développées, peut produire des augmentations significatives de bénéfices. Par exemple, dans le cas où le système d'auto-partage offre une couverture maximale de la zone urbaine sans relocalisation, les déséquilibres dans le réseau ont entraîné une perte d'exploitation de 1160 euros/jour. Cependant, lorsque les politiques de relocalisation ont été appliquées, les résultats des simulations indiquent que des profits de 854 euros/jour pourraient être réalisés. En utilisant un modèle PLNE, les résultats sont encore meilleurs, avec un bénéfice atteignant 3865,7 euros/jour. Ces résultats démontrent l'importance des opérations de relocalisation pour rentabiliser un service d'auto-partage *à un seul sens*.

Uesugi et al. (2007) proposent de leur côté une politique d'affectation de véhicules selon leur distribution. Barth et al. (2004) introduisent quant à eux un mécanisme de relocalisation des véhicules appelé «*ride sharing*», basé sur les utilisateurs afin de réduire le personnel de la relocalisation. Les relocalisations peuvent être effectuées par le personnel du système, ce qui peut être lourd et coûteux. Afin d'alléger le problème de redistribution et de réduire le nombre de relocalisations, ils introduisent deux mécanismes de relocalisation basés sur l'utilisateur. Lorsque le système «se rend compte» qu'il est déséquilibré, il invite les utilisateurs qui ont plus d'un passager à prendre des véhicules distincts lorsque davantage de véhicules sont nécessaires à la station de destination (répartition des trajets). Inversement, si deux utilisateurs se trouvent en même temps à la station d'origine et se rendent à la même destination, le système peut les inviter à faire du covoiturage (se joindre à un voyage). Ce concept a été mis en œuvre à la fois sur un système réel d'auto-partage d'un campus universitaire et dans un modèle de simulation informatique. Les résultats du modèle montrent qu'il peut y avoir une réduction de 42% du nombre de relocalisations

utilisant ces techniques.

1.4.4 Vélo-partage

D'autres problèmes de transport présentent des contraintes ou des objectifs similaires à l'auto-partage. Le plus proche est le vélo-partage qui présente aussi un problème de relocalisation des vélos. En effet, les systèmes de partage de vélo sont des systèmes de transport qui permettent aux utilisateurs de louer un vélo dans l'une des nombreuses stations automatiques disséminées dans une ville, de les utiliser pour un court trajet et de les retourner à n'importe quelle station. Un facteur crucial pour le succès d'un tel système est sa capacité à assurer une bonne qualité de service aux utilisateurs. Cela implique la disponibilité des vélos pour le ramassage et l'existence d'emplacements libres pour les retourner. Ceci est réalisé au moyen d'une opération de rééquilibrage qui consiste à retirer des bicyclettes de certaines stations et à les transférer vers d'autres stations, à l'aide de véhicules dédiés déplaçant les vélos par paquets. La différence avec l'auto-partage est qu'un grand nombre de vélos peut être déplacé en utilisant un seul (ou quelques) petit (s) camion (s), bien que chaque voiture nécessite un agent pour être déplacée dans un service d'auto-partage. En outre, les véhicules utilisés pour le service d'auto-partage et la relocalisation spatiale sont les mêmes : un client peut utiliser un véhicule, qui peut également être déplacé par un agent. Les lecteurs intéressés peuvent se référer au travail de Schuijbroek et Hampshire (2013) sur le problème du partage de vélo.

Kaspi et al. (2014) proposent d'améliorer les performances des systèmes de partage de vélos à *un seul sens* en intégrant des règles de réservation de places de stationnement. En effet, les utilisateurs doivent indiquer leur station de destination et le système leur réserve alors une place de stationnement jusqu'à leur arrivée. Un modèle markovien du système est formulé avec comme objectif de réduire les temps que passent les clients dans le système avant de restituer leur vélo. Les simulations montrent que la politique de réservation proposée réduit le temps total d'attente des utilisateurs de 14% à 34% pour le système de partage de vélos à Tel-Aviv.

Dans l'article de Labadi et al. (2015), une approche est développée à événements discrets pour la modélisation et l'évaluation des performances des systèmes publics de partage de vélos en utilisant des réseaux de Petri avec le temps, les arcs inhibiteurs et les poids d'arcs variables.

Pour Chen et Sun (2015), le problème de localisation géographique des stations de vélo-partage dans une ville implique de sélectionner un certain nombre de stations, puis de les construire dans une zone de planification comportant de nombreuses stations de vélos candidates. Dans cet article, un modèle ma-

thématique est proposé pour formuler le problème de localisation des stations, dans le but de minimiser le temps total de déplacement des utilisateurs et les contraintes budgétaires d'investissement. Ce modèle permet de calculer simultanément le nombre et l'emplacement des stations de vélo ainsi que leurs capacités.

Dans l'article de Kadri et al. (2016), le problème de relocalisation de vélo est traité comme un problème de routage de véhicule. Le véhicule chargé de la relocalisation effectue des visites entre les stations pour les ramener à leur niveau souhaité, qui est connu à l'avance, et chaque station doit être visitée une fois et une seule fois par un véhicule. Ce problème est similaire au problème du *VRPPD* car il s'agit de collecte et de dépôt d'objets avec des contraintes supplémentaires. L'objectif est de trouver un planning optimal du véhicule chargé de la relocalisation qui minimise le temps d'attente total des stations en état de déséquilibre. Les simulations effectuées sur un grand nombre d'instances et les résultats obtenus montrent l'efficacité de leur algorithme de *Branch and Cut*.

1.4.5 Auto-partage électrique et positionnement de la thèse

Dans le domaine de l'optimisation et de l'aide à la décision pour l'auto-partage *à un seul sens*, Boyacı et al. (2015) ont tenu compte des contraintes de recharge électrique des voitures. Dans leurs travaux, le problème de relocalisation des véhicules a été traité comme un problème d'échange de flux de voitures entre les stations. Un modèle PLNE simplifié est proposé dans le but de réduire les temps d'exécution. Mais même le modèle simplifié prend en moyenne 3 heures d'exécution pour des instances d'auto-partage de la ville de Nice. Les auteurs supposent qu'après chaque utilisation, les véhicules doivent se recharger pendant la même durée (2 heures) indépendamment de la distance parcourue. En ce qui nous concerne, et ce pour une étude plus réaliste des contraintes de recharge électrique, nous modélisons le problème comme un problème de routage de véhicules au lieu d'un problème de flot général. Cela nous permet de suivre les véhicules et de leur affecter des temps de recharge en fonction des distances parcourues. Nous tenons également compte du temps d'utilisation du véhicule par un client, d'où découle un temps de recharge proportionnel.

Plus généralement, les problèmes d'optimisation dans la littérature ne traitent pas les contraintes de recharge électriques ou les prennent en compte de manière plutôt simpliste, ce qui permet de formuler le problème en tant qu'un problème de flot. Cette formulation facilite la résolution et la rend faisable avec des méthodes exactes et notamment des PLNE, sans avoir recours à des méthodes approchées ou heuristiques. Dans le chapitre suivant, nous allons voir qu'inclure les contraintes de rechargement électrique de manière plus réaliste

nécessite le passage d'une modélisation basée sur un flot de voitures à une modélisation basée sur un ensemble de chemins. Un chemin pour chaque voiture qui trace sa trajectoire durant la journée, ce qui débouche sur un problème de routage de voitures. Le passage à un problème de routage augmente la complexité du problème et rend la résolution d'instances issues du monde réel impossibles dans des temps raisonnables, ce qui nous amène, en complément des travaux existant dans la littérature, à proposer des heuristiques aux problèmes traités. Autres aspects qui n'ont pas été traités d'une manière suffisamment réaliste à notre connaissance : l'optimisation du nombre d'agents et de leurs horaires, ainsi que les réservations dynamiques des clients. Deux problématiques traitées par la suite dans nos travaux, de manières exacte et heuristique.

1.5 Conclusion

Dans ce chapitre, nous avons présenté les problématiques de transport classiques jugées proches de l'auto-partage et les méthodes d'optimisation utilisées pour résoudre ces problèmes. En effet, le *VRP* et le problème de flot présentent des similitudes avec les problèmes d'optimisation de l'auto-partage, celui-ci peut être formalisé comme un problème de flot entre les stations ou bien comme une version étendue du *VRP*, qui intègre à la fois les spécificités liées aux véhicules électriques et aux capacités des stations. Ce chapitre nous a aussi permis de donner une brève présentation aux travaux existants qui traitent de l'auto-partage spécifiquement. Également, nous avons introduit d'autres problématiques d'optimisation cousines (car du domaine du transport partagé) de l'auto-partage, comme le covoiturage, le transport à la demande, ou encore le vélo-partage. La cloche de fin de ce premier chapitre sur l'état de l'art a été sonnée par un positionnement de notre problématique par rapport à la littérature. Le chapitre suivant permettra de définir plus formellement le problème d'optimisation de l'auto-partage tel que nous l'envisageons.

Chapitre 2

Description et modélisation des problèmes d'optimisation dans l'auto-partage *à un seul sens* de voitures électriques avec stations, réservations et relocalisations

Sommaire

2.1 Introduction

Nous considérons deux problèmes d'optimisation dans l'auto-partage *à un seul sens* qui fait l'objet de cette étude :

— Dans le premier problème, se décide la réservation des voitures (affectation aux clients) et les relocalisations nécessaires. Nous l'appellerons ***ROCSP (Recharging One way Car Sharing Problem)***

— Dans le second problème, on s'intéresse à la planification des agents chargés de réaliser les relocalisations. Nous l'appellerons ***ESRP (Employee Scheduling Routing Problem)***

Ce chapitre est dédié à la définition et à la modélisation des deux problèmes d'optimisation. Nous commencerons par présenter les données et contraintes liées aux deux problèmes, ainsi que leurs objectifs. Nous étudierons différentes formulations mathématiques, modélisant plusieurs variantes du premier problème, tout en stipulant la complexité de chaque variante. Quant au second problème de planification d'agents, une seule formulation sera proposée. Dans la dernière section, nous appliquerons un modèle de génération de requêtes de réservation des voitures pour la ville de Nice.

2.2 Description de l'auto-partage *à un seul sens*

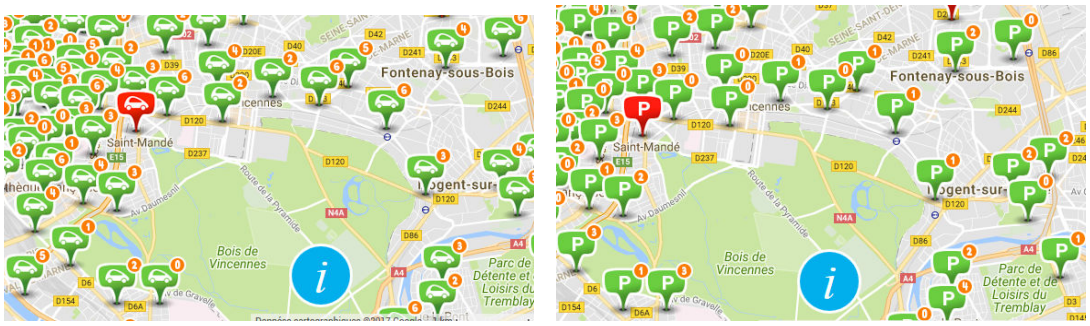
Avant de pouvoir définir plus formellement les deux problèmes d'optimisation considérés, nous allons préciser les données manipulées et les contraintes imposées et illustrer le problème d'optimisation que pose l'auto-partage *à un seul sens* par un petit exemple, afin de faciliter la compréhension.

2.2.1 Données

Les données de l'auto-partage pour une journée de service donnée se composent d'un ensemble de stations, un ensemble de voitures électriques, un ensemble de requêtes de réservation de voitures et un ensemble d'agents qui effectuent la relocalisation des voitures.

Stations

Nous disposons d'un ensemble $S = \{s_1, s_2, ..., s_{S^{max}}\}$ de stations à la fois pour le stationnement et pour la recharge électrique des véhicules, chaque station $s_i \in S$ ayant une capacité K_{s_i} maximale de places. On peut observer dans la figure 2.1, l'exemple d'une carte de recherche de stations *Autolib* en temps réel à Paris. Dans cet exemple, on peut géolocaliser une station et visualiser le nombre de voitures disponibles 2.1(a) ainsi que les places de rechargement 2.1(b) en temps réel.



(a) Voitures disponibles (b) Places disponibles

FIGURE 2.1 – *Carte de recherche de stations Autolib en temps réel à Paris, source image : https ://www.autolib.eu/fr/*

Les véhicules électriques

L'ensemble $W = \{w_1, w_2, ..., w_{W^{max}}\}$ réunit les véhicules électriques, qui doivent se recharger sur les bornes disponibles dans les stations. La question de la charge et de l'autonomie des véhicules électriques est le sujet le plus controversé concernant la voiture électrique. L'autonomie des voitures 100 % électriques utilisées par les opérateurs d'auto-partage varie de 150 à 250 km, par exemple :

— RENAULT ZOE donnée à 250 km d'autonomie efficace pour sa dernière version ;

— PEUGEOT iOn donnée à 150 km d'autonomie ;

— RENAULT KANGOO donnée à 200 km d'autonomie pour sa dernière version ;

— Bluecar d'Autolib donnée à 250 km d'autonomie.

Cependant, l'autonomie des voitures n'est pas le souci numéro un des opérateurs d'auto-partage, car l'auto-partage est utilisé généralement comme un moyen de transport urbain pour des petits trajets, comme le montre le rapport d'activité d'Autolib (2014), qui donne le chiffre de 9,8 km par trajet en moyenne. En revanche, le temps de recharge représente une contrainte importante du système, vu que chaque temps de recharge pendant la journée de service est un temps éventuel de location en moins pour les opérateurs. Recharger une voiture électrique est différent du remplissage d'un plein d'essence. Dans l'ensemble, on passe une dizaine de minutes pour un plein d'essence et un certain nombre d'heures pour une recharge totale de batterie. Le nombre d'heures dépend du type de charge utilisé parmi les 4 types existants :

1. La recharge « lente » : sur une prise électrique domestique (220 V) ou sur une prise renforcée ;
2. La recharge « normale » : sur une borne de recharge délivrant une puissance comprise entre 3 et 7 kW ;
3. La recharge « accélérée » : sur une borne de recharge délivrant une puissance de 22 kW ;
4. La recharge « rapide » : sur une station de recharge délivrant une puissance comprise entre 43 et 53 kW.

Par exemple, la recharge utilisée par les opérateurs d'auto-partage comme Autolib est une recharge « normale » de 3 kW.

Les requêtes de réservation des clients

L'ensemble des requêtes des clients est $Q = \{q_1, q_2, ..., q_{Q^{max}}\}$, où l'on associe à chaque requête (*query*) $q_i \in Q$, un départ de la station $S^d_{q_i}$ à la période de temps $T^d_{q_i}$ vers la station $S^f_{q_i}$ à la période de temps $T^f_{q_i}$ ainsi qu'un gain G_{q_i} dû à la satisfaction de la requête.

Les agents

On dispose également d'un ensemble d'agents (salariés de l'opérateur du service) $A = \{a_1, a_2, ..., a_{A^{max}}\}$ capables de déplacer un seul véhicule à la fois

FIGURE 2.2 – *Voiture électrique garée dans une station entrain de se recharger, source image https ://www.auto-bleue.org/fr*

entre les stations. On peut avoir deux types d'agents chargés de la relocalisation des voitures. Chaque service de partage de voitures utilise l'un des deux types, ou les deux à la fois, selon le nombre d'utilisateurs et la nécessité ou non d'accueillir les utilisateurs dans les stations.

— *Agent d'accueil et d'exploitation*. Les opérations de relocalisation ne représentent qu'une petite partie de l'activité des agents d'accueil et d'exploitation, qui comprend également les activités de saluer, d'assister les clients et d'assurer le bon fonctionnement du service. Ce type de personnel est chargé de déplacer les véhicules pour équilibrer la distribution entre les stations. Mais cela constitue une petite partie «saupoudrée» dans leur journée de travail. Dans ce cas de figure, on ne s'intéresse pas à l'optimisation des trajets, ni aux horaires de ce type d'agents, car cela sort du cadre du problème de relocalisation des voitures et constitue un nouveau problème assez complexe, notamment si l'on intègre les contraintes des conventions collectives dans des activités multiples ;

— *Agent de relocalisation*. Un petit nombre d'agents s'occupe de la relocalisation à proprement parler, leur principale mission étant l'équilibrage des stations en termes de répartition des voitures. Si un agent doit se rendre à une autre station, sans déplacer une voiture, il peut prendre les transports publics ou un petit scooter disponible pour voyager entre les stations. Ce scooter peut être remorqué à une voiture du système. Dans ce cas de figure, l'optimisation des trajets de ce type d'agents peut être traitée dans le cadre du problème d'optimisation de la relocalisation. Ce type d'agents est celui qui sera plus particulièrement considéré dans la suite de cette thèse.

La journée de service

Potentiellement, on peut prendre la décision de déplacer une voiture d'une station vers une autre à n'importe quel instant de la journée, ce qui donne un nombre infini de décisions possibles et rend difficile la modélisation et la résolution de notre problème. C'est pourquoi nous proposons une discrétisation du temps de la journée de service, dans le but d'avoir un nombre fini de décisions à prendre à chaque pas de temps.

Soit l'ensemble $T = \{1, 2, ..., T^{max}\}$ des périodes de temps, qui divise le temps d'une journée de service en des petites périodes de temps de même durée, chaque $t \in T$ représente l'instant de début de la période de temps mais par abus de langage nous parlerons de période de temps $t \in T$. La durée de chaque période est fixée à *15 min* dans nos travaux, ce qui nous semble assez précis pour la nature du problème traité. Réduire encore plus la durée des périodes de temps rend le problème difficile à résoudre dans des temps raisonnables. Le nombre de périodes de temps nécessaire pour effectuer le trajet entre les stations s_i et s_k est donné par $T_{s_i s_k}$.

2.2.2 Règles, contraintes et suppositions

Le problème étant soumis à des contraintes de temps et de ressources, nous posons aussi un certain nombre d'hypothèses comme suit :

— Un client peut demander une location de voiture, caractérisée par une station et une date de départ, conjointement avec une station et une date d'arrivée ;

— Une voiture peut satisfaire des clients, être déplacée par un agent ou bien être garée dans une station de recharge ;

— Un agent peut déplacer une voiture d'une station à une autre, attendre dans une station ou se déplacer entre deux stations sans utiliser des voitures du service ;

— Chaque voiture ou chaque agent peut commencer la journée dans une station et la terminer dans une autre station ;

— Le nombre de voitures garées dans une station ne doit pas dépasser la capacité de la station ;

— Aucun trajet ne doit dépasser l'autonomie des voitures électriques ;

— Chaque agent a un nombre maximal d'heures travaillées ;

— Toutes les distances considérées sont transformées en distances temporelles (temps de parcours en fait) en utilisant les limites de vitesse dans les routes utilisées ainsi qu'une vitesse moyenne pour la transformation ;

— Le temps de décharge de la batterie est proportionnel à la durée du trajet effectué. Plus on roule, plus on consomme de l'énergie. Dans nos simulations, nous considérons le pire cas, à savoir que le client utilise sa voiture tout au long de la période de réservation ;

— Le taux de recharge est également proportionnel au temps de recharge passé dans une station, jusqu'au moment où la charge maximale de la batterie est atteinte ;

— deux contraintes de recharge de la batterie électrique des voitures sont considérées :

1. contrainte de recharge *forte* : chaque voiture doit être rechargée après chaque voyage avec un taux de charge égale à celui consommé lors du trajet ;
2. contrainte de recharge *normale* : la voiture n'est pas obligée de se recharger avant que la batterie soit vide ;

— Les éventuels ralentissements ou bouchons dans la circulation ne sont pas considérés à ce stade de la modélisation, malgré leur effet sur le respect des horaires et l'énergie consommée, car nous considérons qu'ils interviennent davantage sur les matrices de temps origine-destination, qui pourraient être potentiellement introduites dynamiquement dans le système ;

— Notre système suppose la connaissance préalable des réservations et un engagement des utilisateurs au niveau des stations et des horaires ; le problème est traité de façon statique dans ce chapitre et le chapitre suivant. Une prise en compte dynamique des réservations et le recours à des fenêtres de temps fera l'objet du chapitre 4.

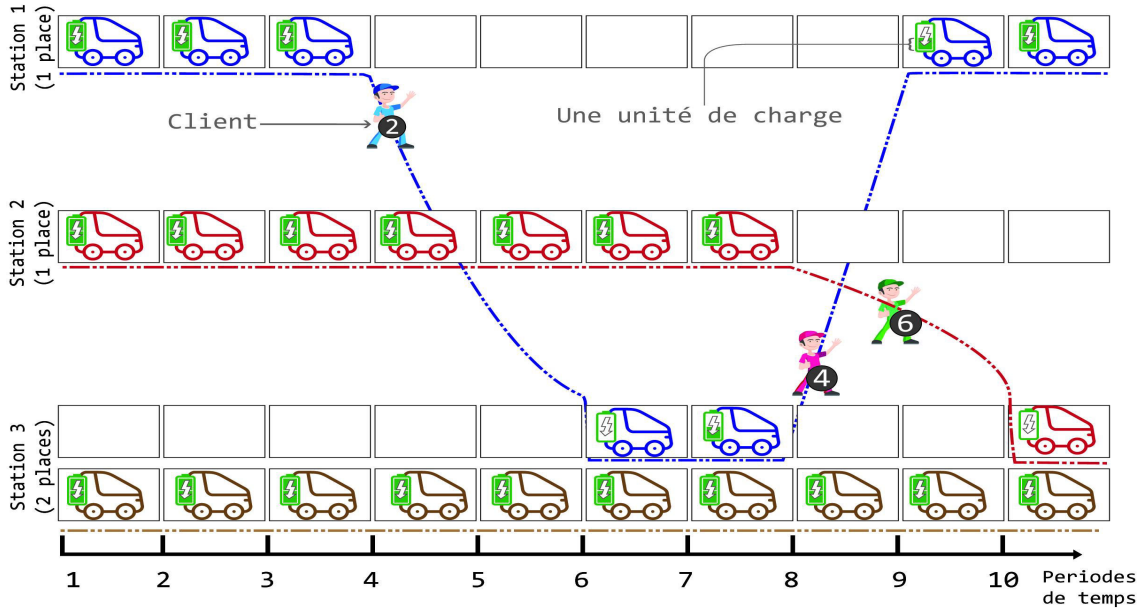
2.2.3 Exemple illustratif

La figure 2.3 décrit deux schémas logistiques simplifiés associés à un exemple d'une journée de service d'auto-partage avec des voitures électriques. L'exemple a pour but d'illustrer de façon didactique le propos : le premier schéma 2.3(a) montre l'utilisation du service sans relocalisation tandis que le deuxième 2.3(b) montre l'apport de la relocalisation. Dans cet exemple, le service d'auto-partage est basé sur 3 voitures électriques et 3 stations tel que :

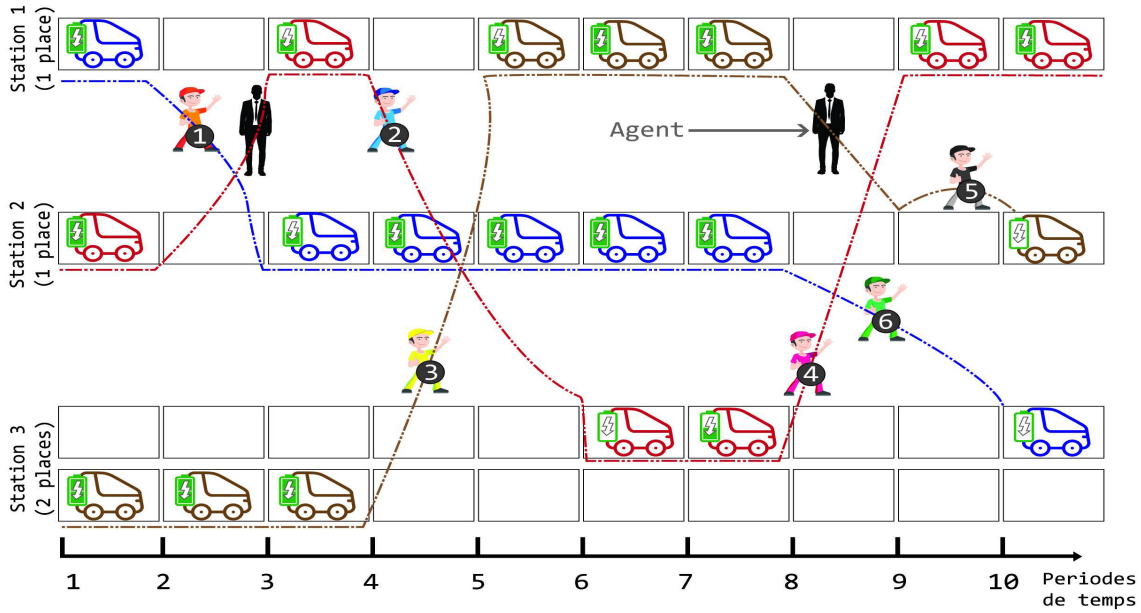
— La *Station 1* (1 place de parking et de recharge) ;

— La *Station 2* (1 place de parking et de recharge) ;

— La *Station 3* (2 places de parking et de recharge).



(a) Scénario d'auto-partage sans relocalisation avec 3 véhicules et 3 stations distinctes ; le client 2 a utilisé la voiture bleue de $t = 4$ à $t = 6$



(b) Scénario d'auto-partage avec relocalisation et 6 clients satisfaits ; l'agent a déplacé la voiture rouge de la stations 2 à t=2 à la station 1 et la voiture marron de la station 1 à la station 2 à t=8

FIGURE 2.3 – *Exemple illustratif d'auto-partage avec et sans relocalisation*

TABLE 2.1 – *Exemple avec six requêtes de clients*

Clients	Départ		Arrivée	
	Station	Période de temps	Station	Période de temps
client 1	3	3	1	4
client 2	2	2	3	3
client 3	2	2	3	4
client 4	3	2	2	3
client 5	3	2	2	4
client 6	2	4	3	5

La journée de service est divisée en dix périodes de temps. La voiture a une autonomie de deux unités de charge (deux périodes de temps) et le temps nécessaire pour recharger une voiture avec une unité de recharge est égal à une période de temps. Le temps du trajet entre chaque couple de stations est égal à une période de temps également. Une requête de réservation de client est caractérisée par un couple (station, période de temps) de départ et d'arrivée. Les six requêtes de cet exemple sont indiquées dans le tableau 2.1.

Dans le scénario sans relocalisation 2.3(a), chaque voiture change de station plusieurs fois dans la journée. Ce changement est la conséquence d'un client effectuant son déplacement. L'ensemble des déplacements d'une voiture trace sa trajectoire durant la journée de service. Observons par exemple la trajectoire de la voiture bleue qui commence à la *Station 1*, où elle reste en attente jusqu'à la période *t=3* et sert le *client 2*. Ce dernier consomme toute la charge de la batterie et donc la voiture se recharge pendant la période de temps *t=6* et *t=7* avant de pouvoir satisfaire la requête du *client 4* à *t=8*. Dans cette situation, il est impossible de satisfaire davantage de clients.

Dans le scénario avec relocalisation 2.3(b), les déplacements des voitures sont soit la conséquence d'un client effectuant son trajet, soit d'un agent réalisant une opération de relocalisation. L'agent déplace la voiture rouge à *t=2* de la *Station 2* à la *Station 1*. Ensuite, il reste à cette dernière station jusqu'à *t=8*, où il déplace la voiture marron à la *Station 2* pour servir le *client 5*. Dans ce scénario, les six requêtes des clients sont satisfaites.

En conclusion, on peut observer dans cet exemple, que grâce aux relocalisations opérées par l'agent, les six requêtes de clients ont pu être satisfaites et l'on pressent toute la potentialité de cette approche pour l'optimisation du service.

2.3 Définition de deux problèmes d'optimisation : *ROCSP* et *ESRP*

Nous devons résoudre deux problèmes d'optimisation dans l'auto-partage *à un seul sens* séparément dans l'ordre. Le premier problème est celui de la réservation des voitures et le calcul des relocalisations nécessaires (*Recharging One way Car-Sharing Problem* ou *ROCSP*). Le second problème est celui du planning des agents et leurs routages (*Employee Scheduling Routing Problem* ou *ESRP*).

2.3.1 *Recharging One way Car-Sharing Problem* ou *ROCSP* : réservation et relocalisation

Le premier problème d'optimisation contient deux niveaux de décision : l'acceptation ou le refus des requêtes des clients et le calcul de l'ensemble des opérations de relocalisation nécessaires pour satisfaire les requêtes acceptées tout en respectant les contraintes de capacité des stations et de recharge électrique des voitures.

Données et contraintes

Les données du *ROCSP* sont résumées dans le tableau 2.2. Certaines notations ne sont pas encore définies et le seront au fil de cette thèse en fonction des problèmes adressés. Quant au contraintes, ce sont les contraintes exprimées dans 2.2.2, en dehors des contraintes liées aux agents.

Objectif

Le *ROCSP* peut être formulé comme un problème de flots de voitures ou de routage, comme nous allons le voir dans la section suivante. Dans les deux cas, on vise à minimiser une fonction objectif Ct obtenue par la somme pondérée des trois critères C_V , C_R et C_C .

$$\min \quad \{Ct = \alpha \cdot C_V + \beta \cdot C_R - \gamma \cdot C_C\} \tag{Ct}$$

où :

— C_V est le nombre de voitures utilisées ;

— C_R est la somme des distances parcourues lors des opérations de relocalisation ;

TABLE 2.2 – *Données du ROCSP (Recharging One way Car-Sharing Problem)*

S	L'ensemble des stations de l'auto-partage.
$s_i \in S$	la station numéro i avec i allant de 1 à S^{max}.
K_{s_i}	La capacité ou le nombre de places de la station s_i.
W	L'ensemble des voitures de l'auto-partage.
$w_j \in W$	La voiture numéro j avec j allant de 1 à W^{max}.
T	L'ensemble des périodes de temps qui divisent la journée de service et allant de 1 à T^{max}.
$T_{s_i s_k}$	Le nombre de périodes de temps nécessaire pour faire le trajet de la station s_i à la station s_k.
Q	L'ensemble des requêtes de réservation des clients de l'auto-partage.
$q_i \in Q$	la requête numéro i avec i allant de 1 à Q^{max}.
$S^d_{q_i} \in S$	La station de départ de la requête q_i.
$T^d_{q_i} \in T$	La période de temps de départ de la requête q_i.
$S^f_{q_i} \in S$	La station d'arrivée de la requête q_i.
$T^f_{q_i} \in T$	La période de temps d'arrivée de la requête q_i.
$T^r_{s_i s_k}$	Le nombre de périodes de temps d'autonomie électrique consommées après le trajet entre la station s_i et s_k.
$T^r_{q_i}$	Le nombre de périodes de temps d'autonomie électrique consommées après le trajet du client de la requête q_i.
G_{q_i}	Le gain économique dû à la satisfaction de la requête q_i.
T^r	Le nombre de périodes de temps d'autonomie électrique rechargée pour une voiture qui reste dans la même station pendant une période de temps.
R_{w_j}	L'autonomie maximale en termes de périodes de temps de la voiture électrique w_j.
R'_{w_j}	L'autonomie initiale en termes de périodes de temps de la voiture électrique w_j au début de la journée de service.
Ct	La fonction objectif du *ROCSP* avec $Ct = \alpha \cdot C_V + \beta \cdot C_R - \gamma \cdot C_C$ où C_V est le nombre de voitures utilisées, C_R la somme des distances des opérations de relocalisation des voitures et C_C le gain dû aux requêtes satisfaites. Les coefficients α,β et$-\gamma$ leurs poids respectifs dans la fonction objectif.
$G^{ROCSP}(V, U, R)$	Graphe spatio-temporel modélisant le *ROCSP* avec V l'ensemble des sommets, U l'ensemble des arcs et R l'ensemble des consommations de la ressource électrique des arcs, voir 2.4.1.

— C_C est la somme des gains relatifs aux requêtes satisfaites ; ce critère est multiplié par un coefficient négatif qui doit être maximisé (non minimisé).

Dans les travaux présentés dans cette étude, nous n'introduisons pas d'optimisation multi-critères à proprement parler. La pondération des trois critères nous permet toutefois d'adapter la fonction objectif selon les besoins. Les valeurs des poids α, β et γ correspondent en effet à l'importance des différents objectifs que nous voulons souligner dans notre problème d'optimisation. Dans les simulations des chapitres qui suivent, nous nous baserons sur une estimation économique. Aussi, nous envisagerons d'autres scénarios de fonction objectif en fixant arbitrairement un gradient net d'importance relative entre les critères.

Quand Ct est négative, elle correspond à un gain, car cela veut dire que le critère correspondant aux recettes de locations C_C pondéré avec un coefficient négatif, est plus grand que la somme des deux autres critères C_V et C_R pondérés avec des coefficients positifs. Ces deux derniers critères correspondent quant à eux aux coûts de fonctionnement du service. Minimiser Ct quand elle prend des valeurs négatives revient à maximiser sa valeur absolue qui représente un gain.

2.3.2 *Employee Scheduling Routing Problem* ou *ESRP* : planification des agents

Le problème de planning des agents de relocalisation prend comme entrée l'ensemble des opérations de relocalisation obtenu après la résolution du premier problème *ROCSP*. Il s'agit de fixer le planning et le routage des agents pour effectuer l'ensemble des opérations de relocalisation en entrée.

Données et contraintes

Avant de pouvoir modéliser l'$ESRP$, on définit l'ensemble $O = \{o_1, o_2, ..., o_{O^{max}}\}$, où pour chaque opération $o_i \in O$ sont associées $S^d_{o_i}$ la station de départ ou de début, $T^d_{o_i}$ la période de temps à laquelle commence l'opération, $S^f_{o_i}$ la station d'arrivée ou de fin, $T^f_{o_i}$ la période de temps à laquelle se finit l'opération, de telle sorte que le temps de fin $T^f_{o_i}$ est égal au temps de départ $T^d_{o_i}$ plus le temps de trajet $T_{S^d_{o_i} S^f_{o_i}}$ qui sépare les stations de départ et d'arrivée.

Les données du $ESRP$ sont celles déjà définies pour le $ROCSP$, en plus des données spécifiques au $ESRP$ résumées dans le tableau de notations 2.3. Les contraintes du $ESRP$ sont celles exprimées précédemment dans 2.2.2 et qui sont propres aux agents.

TABLE 2.3 – *Données du ESRP : (Employee Scheduling Routing Problem)*

A	L'ensemble des agents de relocalisation.
$a_j \in A$	L'agent numéro j avec j allant de 1 à A^{max}.
R_{a_j}	Le nombre de périodes de temps maximum travaillé par l'agent a_j dans une journée.
$T'_{s_i s_k}$	Le nombre de périodes de temps nécessaire pour faire le trajet de la station s_i à la station s_k sans utiliser une voiture du service.
O	L'ensemble des opérations de relocalisation.
$o_i \in O$	L'opération de relocalisation numéro i avec i allant de 1 à O^{max}.
$S^d_{o_i} \in S$	La station de départ de l'opération de relocalisation o_i.
$T^d_{o_i} \in T$	La période de temps de départ de l'opération de relocalisation o_i.
$S^f_{o_i} \in S$	La station d'arrivée de l'opération de relocalisation o_i.
$T^f_{o_i} \in T$	La période de temps d'arrivée de l'opération de relocalisation o_i.
Ct'	La fonction objectif du $ESRP$.
C_A	Le nombre d'agents déployés.
C_D	La somme des distances parcourues par les agents pour relocaliser les voitures sans l'utilisation des voitures du service.
$G^{ESRP}(V', U', R')$	Graphe modélisant l'$ESRP$ avec V' l'ensemble des sommets, U' l'ensemble des arcs et R' l'ensemble des consommations de la ressource qui est le temps de travail des agents R_{a_j}, voir 2.5.1.

Objectif

L'objectif se décompose en deux sous-objectifs hiérarchisés avec :

— comme objectif principal, la minimisation du nombre d'agents nécessaires pour effectuer les opérations de relocalisation,

— et ensuite, comme objectif secondaire, la minimisation de la somme des distances de déplacement des agents entre les stations effectuée sans prendre une voiture de service.

Pour ce faire, nous minimisons un coût total Ct' obtenu par la somme pondérée de deux critères, la pondération permettant de hiérarchiser les deux objectifs.

$$\min \quad \{Ct' = \delta \cdot C_A + \omega \cdot C_D\} \tag{Ct'}$$

où :

— C_A est le nombre d'agents déployés ;

— C_D est la somme des distances parcourues par les agents pour relocaliser les voitures (sans voiture du service, en transport public ou en scooter).

2.4 Modélisation PLNE du *ROCSP*

Le problème de relocalisation d'une flotte de véhicules électriques en auto-partage peut être modélisé comme un problème de flots dans un réseau de transport ou un problème de routage de voitures. Nous allons voir les différences entre les deux modélisations en termes de complexité pour bien modéliser les différentes contraintes du problème. Avant de présenter ces deux modèles plus en détails, nous introduisons un graphe spatio-temporel modélisant le problème et sur lequel se basent tous les modèles de Programmation Linéaire en Nombres Entiers (PLNE) proposés, mais aussi d'autres heuristiques présentées dans les chapitres suivants.

2.4.1 Graphe spatio-temporel

Dans une solution donnée du problème traité, chaque véhicule est associé à un chemin qui commence au début d'une journée de service dans une station et visite d'autres stations jusqu'à la fin du service. Un tel chemin est fondamentalement caractérisé par ses traces dans deux dimensions : l'espace et le temps. Il est donc naturel d'utiliser un graphe spatio-temporel.

Soit $G^{ROCSP}(V, U, R)$ un graphe sur lequel on va se baser pour modéliser le *ROCSP* tel que :

— L'ensemble des sommets $V = \{v_{s_i}^t : t \in T, s_i \in S\} \cup \{V^d = v_{s_0}^0, V^f = v_{s_0}^{T^{max}+1}\}$ est composé de tous les sommets $v_{s_i}^t$ où s_i représente une station et t une période de temps, en plus des sommets fictifs V^d et V^f représentant respectivement le début et la fin de la journée de service ;

— L'ensemble des arcs $U = U_1 \cup U_2 \cup U_3 \cup U_4$ où :

- $U_1 = \{(V^d, v_{s_i}^1), (v_{s_i}^{T^{max}}, V^f) : s_i \in S\}$, chaque arc dans U_1 représente un lien entre les sommets fictifs V^d et V^f et les autres sommets de début de journée à $t = 1$ et de fin de journée à $t = T^{max}$;
- $U_2 = \{(v_{s_i}^t, v_{s_i}^{t+1}) : s_i \in S, t \in T \setminus \{T^{max}\}\}$, chaque arc $(v_{s_i}^t, v_{s_i}^{t+1})$ représente un lien au sein de la même station entre deux périodes de temps consécutives ;
- $U_3 = \{(v_{s_i}^t, v_{s_k}^{t'}) : s_i, s_k \in S,\ t, t' \in T,\ t' = t + T_{s_i s_k},\ s_i \neq s_k\}$, chaque arc $(v_{s_i}^t, v_{s_k}^{t'})$ représente un lien entre deux stations différentes s_i et s_k ; $t' = t + T_{s_i s_k}$ assure que les voitures respectent le temps du trajet entre les deux stations ;
- $U_4 = \{u_{q_i} : u_{q_i} = (v_{S_{q_i}^d}^{T_{q_i}^d}, v_{S_{q_i}^f}^{T_{q_i}^f}), q_i \in Q\}$, chaque arc u_{q_i} représente une demande de location de voiture à partir de la station $S_{q_i}^d$ à la période $T_{q_i}^d$ vers la station $S_{q_i}^f$ à la période $T_{q_i}^f$.

— L'ensemble des consommations de ressources électriques $R = R_1 \cup R_2 \cup R_3 \cup R_4$ où :

- $R_1 = \{r_u = 0 : u \in U_1\}$, aucune consommation de charge pour les arcs qui relient les sommets fictifs aux sommets de début de la journée de service n'est supposée ;
- $R_2 = \{r_{(v_{s_i}^t, v_{s_i}^{t+1})} = T^r : (v_{s_i}^t, v_{s_i}^{t+1}) \in U_2\}$, chaque voiture augmente son autonomie de T^r périodes de temps lorsqu'elle reste dans la même station pendant une période de temps ;
- $R_3 = \{r_{(v_{s_i}^t, v_{s_k}^{t'})} = T_{s_i s_k}^r : (v_{s_i}^t, v_{s_k}^{t'}) \in U_3\}$, chaque $r_{(v_{s_i}^t, v_{s_k}^{t'})}$ représente la consommation en autonomie entre les stations s_i et s_k ;
- $R_4 = \{r_{u_{q_i}} = T_{q_i}^r : u_{q_i} \in U_4\}$, chaque $r_{u_{q_i}}$ représente la consommation en autonomie de la requête q_i.

Pour plus de clarification, le graphe modélisant l'exemple illustratif 2.2.3 est présenté dans la figure 2.4. Pour des raisons de lisibilité de la figure, seuls les ensembles U_1, U_2 et U_4 ont été modélisés. Le nombre d'arcs dans l'ensemble U_3 est trop grand pour qu'on puisse le représenter lisiblement.

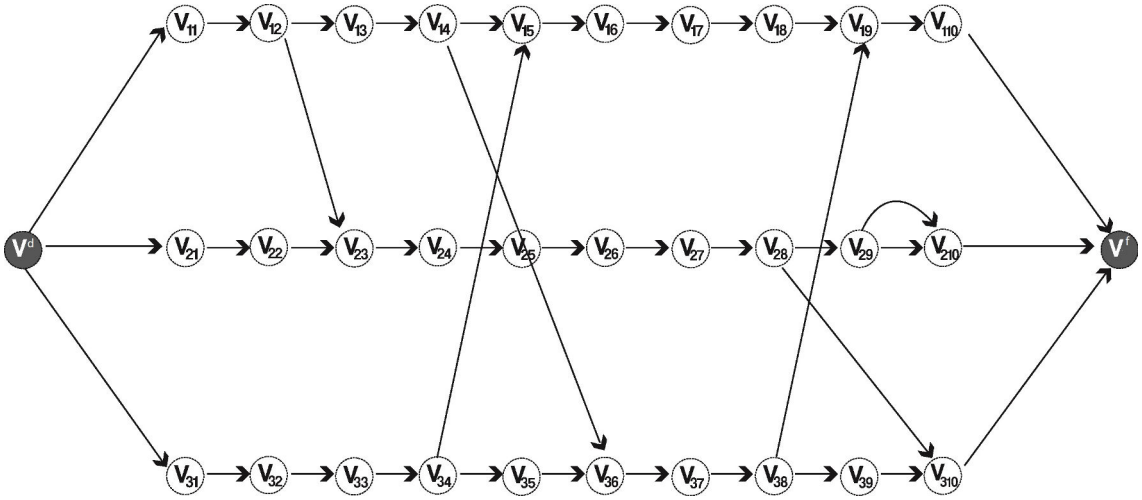


FIGURE 2.4 – *Graphe spatio-temporel modélisant l'exemple 2.2.3*

2.4.2 Modèle PLNE basé sur les flots avec contrainte de recharge *forte* : $F\text{-}PLNE^{ROCSP}$

Dans un premier temps, on va omettre les contraintes de recharge électrique et modéliser un problème de relocalisation dans un système d'auto-partage avec des voitures non électriques. La relaxation des contraintes de recharge électrique donne le modèle $F\text{-}PLNE^{OCSP}$: *Flow PLNE for the One way Car Sharing Problem* (*OCSP* sans *le Recharging*), en tant que problème de flot de coût minimum classique ayant une complexité polynomial (Jewell et University of California, 1966). Dans un second temps, nous introduisons les contraintes de recharge électrique dans le modèle de flot, ce qui va déboucher sur le modèle $F\text{-}PLNE^{ROCSP}$. Mais la modélisation basée sur des flots nous oblige à considérer une contrainte de recharge plus forte que dans le cas réel. Plus précisément, comme ne nous pouvons pas tracer l'historique des déplacements des voitures dans un modèle de flots, nous forçons chaque voiture à se recharger après chaque trajet effectué, au lieu de considérer une autonomie globale sur toute la journée.

Modèle PLNE de flot sans contrainte de recharge électrique : $F\text{-}PLNE^{OCSP}$

Dans le modèle $F\text{-}PLNE^{OCSP}$, deux vecteurs de variables de décision f_u et z_{q_i} représentant les flots sont utilisés , tels que :

— $f_u \in [0, W^{max}]$: le nombre de voitures qui passent par l'arc $u \in U_1 \cup U_2 \cup U_3$.

— $z_{q_i} \in \{0,1\}$: égale à 1 si la requête q_i est satisfaite, ce qui correspond à une voiture qui passe par l'arc $u_{q_i} \in U_4$, 0 sinon.

$$\min Ct = \alpha \cdot C_V + \beta \cdot C_R - \gamma \cdot C_C \qquad (F\text{-}PLNE^{OCSP})$$

avec :

— $C_V = \sum_{(V^d, v^t_{s_i}) \in U_1} f_{(V^d, v^t_{s_i})}$

— $C_R = \sum_{(v^t_{s_i}, v^{t'}_{s_k}) \in U_3} T_{s_i s_k} \cdot f_{(v^t_{s_i}, v^{t'}_{s_k})}$

— $C_C = \sum_{q_i \in Q} G_{q_i} \cdot z_{q_i}$

sous contraintes exprimées par les équations numérotées de de (2.1) à (2.3).

$$\sum_{v^t_{s_i} \in V} f_{(v^t_{s_i}, v^{t'}_{s_k})} + \sum_{q_i \in Q : S^f_{q_i} = s_k, T^f_{q_i} = t'} z_{q_i} = \sum_{v^{t''}_{s_l} \in V} f_{(v^{t'}_{s_k}, v^{t''}_{s_l})} + \sum_{q_i \in Q : S^d_{q_i} = s_k, T^d_{q_i} = t'} z_{q_i}$$

$$\forall v^{t'}_{s_k} \in V \setminus \{V^d, V^f\} \qquad (2.1)$$

$$\sum_{(V^d, v^t_{s_i}) \in U_1} f_{(V^d, v^t_{s_i})} \leq W^{max} \qquad (2.2)$$

$$f_{(v^t_{s_i}, v^{t+1}_{s_i})} \leq K_{s_i} \qquad \forall (v^t_{s_i}, v^{t+1}_{s_i}) \in U_2 \qquad (2.3)$$

L'équation (2.1) assure la contrainte de continuité du flot des voitures, en imposant que la somme des flots entrant à chaque sommet égale celle des sortants. L'équation (2.2) garantit que le flot des voitures ne dépasse pas le nombre de voitures. Enfin, l'équation (2.3) garantit la contrainte de capacité à chaque station et à chaque période de temps du jour de service.

Introduction de la contrainte de recharge électrique *forte* : $F\text{-}PLNE^{ROCSP}$

Le seul moyen d'ajouter les contraintes électriques au modèle PLNE basé sur les flots est de considérer une contrainte électrique forte, à savoir, obliger chaque voiture à se recharger après chaque trajet effectué. Si une voiture effectue une relocalisation entre la station s_i et s_k, alors elle devra se recharger pendant un temps $T^r_{s_i s_k}$ égal à sa consommation. De même, si la voiture satisfait une requête q_i, alors elle devra se recharger $T^r_{q_i}$ périodes de temps. On obtient le nouveau

modèle *F-PLNE*^{ROCSP} par l'ajout de la contrainte (2.4) qui oblige le nombre de voitures stationnées dans chaque station s_i entre deux périodes de temps consécutives t et $t+1$, à être plus grand que la somme de l'ensemble des arcs entrants dans la station s_i à une période t'', telle que $t'' \leq t$ et $t'' + T^r_{s_k s_i} < t+1$ ou $t'' + T^r_{q_l} < t+1$, cet ensemble d'arcs entrants représente les voitures qui n'ont pas fini leur recharge avant $t+1$.

$$\min Ct = \alpha \cdot C_V + \beta \cdot C_R - \gamma \cdot C_C \qquad (F\text{-}PLNE^{ROCSP})$$

Le modèle *F-PLNE*^{ROCSP} est soumis à l'ensemble des contraintes de (2.1) à (2.3) ainsi que la contrainte (2.4).

$$f_{(v^t_{s_i}, v^{t+1}_{s_i})} \geq \sum_{(v^{t'}_{s_k}, v^{t''}_{s_i}) \in U_3 : t'' \leq t < t'' + T^r_{s_k s_i}} f_{(v^{t'}_{s_k}, v^{t''}_{s_i})} + \sum_{q_l \in Q : S^f_{q_l} = s_i, T^f_{q_l} \leq t < T^f_{q_l} + T^r_{q_l}} z_{q_l} \qquad \forall s_i \in S, \forall t \in T \setminus \{T^{max}\} \qquad (2.4)$$

Complexité

La résolution du modèle *F-PLNE*^{ROCSP} est de complexité *NP-difficile* car la résolution de modèles PLNE d'optimisation est un problème de complexité *NP-difficile* (cf. Papadimitriou (1981)). Sauf lorsque les contraintes prennent la forme d'une matrice totalement unimodulaire et entière, la complexité est alors polynomiale car les solutions du problème relaxé sont entières. Le modèle *F-PLNE*^{OCSP} de flot classique a une matrice totalement unimodulaire et entière, mais le fait d'ajouter la contrainte (2.4), fait que la complexité de résolution du modèle *F-PLNE*^{ROCSP} devient *NP-difficile* car rajouter une contrainte qui lie des flots d'arcs qui n'ont pas forcement de sommet en commun fait que la matrice des contraintes ne vérifie plus la propriété d'unimodularité totale.

2.4.3 Modèle PLNE basé sur les chemins avec contrainte de recharge *normale* : *R-PLNE*^{ROCSP}

Pour ajouter des contraintes électriques plus réalistes, c'est à dire ne pas contraindre les voitures à se recharger après chaque voyage, il faut pouvoir calculer le taux de charge des batteries à chaque période de temps. Or, il est

impossible de distinguer les flots des voitures déchargées dans le modèle *F-PLNE*^{ROCSP}, car la variable f_u donne seulement le nombre de voitures traversant un arc du graphe et ne permet pas de tracer le chemin de chaque voiture à travers les stations tout au long de la journée. Pour contraindre une voiture à se recharger dans le seul cas où elle n'a plus d'autonomie, il faut connaître l'historique de ses déplacements, ce qui nous pousse à aller vers une modélisation basée sur les chemins traçant le trajet de chaque voiture, modélisation complémentaire au modèle de flot. On passe alors d'un problème de flot à un problème de routage, décrit par le modèle *R-PLNE*^{ROCSP} pour *Routing PLNE for the Recharging One way Car-Sharing Problem*, pour lequel est défini B, constante représentant un grand nombre entier positif, ainsi que cinq vecteurs de variables de décisions entières :

— $x^{w_j} \in \{0,1\}$: égale à 1 si la voiture w_j est utilisée durant la journée de service, 0 sinon ;

— $x_u^{w_j} \in \{0,1\}$: égale à 1 si la voiture w_j passe par l'arc $u \in U_1 \cup U_2 \cup U_3$, 0 sinon ;

— $z_{q_i}^{w_j} \in \{0,1\}$: égale à 1 si la requête (query) q_i est satisfaite par la voiture w_j, 0 sinon ;

— $h_{w_j}^t \in [0, R_{w_j}]$: égale au taux de charge électrique de la voiture w_j à la période de temps $t \in T$;

— $y_{w_j}^t \in \{0,1\}$: égale à 1 si la voiture w_j a une autonomie maximale R_{w_j} à la période $t \in T \setminus \{1\}$, 0 sinon.

En conséquence, la fonction objectif se ré-écrit de la manière suivante :

$$\min Ct = \alpha \cdot C_V + \beta \cdot C_R - \gamma \cdot C_C \qquad (R\text{-}PLNE^{ROCSP})$$

avec :

— $C_V = \sum_{w_j \in W} x^{w_j}$

— $C_R = \sum_{w_j \in W} \sum_{(v_{s_i}^t, v_{s_k}^{t'}) \in U_3} x_{(v_{s_i}^t, v_{s_k}^{t'})}^{w_j} \cdot T_{S_i S_k}$

— $C_C = \sum_{w_j \in W} \sum_{q_i \in Q} G_{q_i} \cdot z_{q_i}^{w_j}$

sous contraintes de (2.5) à (2.14).

$$\sum_{v^t_{s_i} \in V} x^{w_j}_{(v^t_{s_i}, v^{t'}_{s_k})} + \sum_{q_i \in Q, S^f_{q_i} = s_k, T^f_{q_i} = t'} z^{w_j}_{q_i} = \sum_{v^{t''}_{s_l} \in V} x^{w_j}_{(v^{t'}_{s_k}, v^{t''}_{s_l})} + \sum_{q_i \in Q, S^d_{q_i} = s_k, T^d_{q_i} = t'} z^{w_j}_{q_i}$$
$$\forall v^{t'}_{s_k} \in V \setminus \{V^d, V^f\}, \forall w_j \in W \quad (2.5)$$

$$x^{w_j} \geq x^{w_j}_{(v^t_{s_i}, v^{t'}_{s_k})} \qquad \forall (v^t_{s_i}, v^{t'}_{s_k}) \in U, \forall w_j \in W \quad (2.6)$$

$$x^{w_j} \geq z^{w_j}_{q_i} \qquad \forall q_i \in Q, \forall w_j \in W \quad (2.7)$$

$$\sum_{w_j \in W} x^{w_j}_{(v^t_{s_i}, v^{t+1}_{s_i})} \leq K_{s_i} \qquad \forall (v^t_{s_i}, v^{t+1}_{s_i}) \in U_2 \quad (2.8)$$

$$h^t_{w_j} \leq h^{t-1}_{w_j} + \sum_{(v^{t'-1}_{s_i}, v^{t'}_{s_i}) \in U_2 : t' = t} x^{w_j}_{(v^{t'-1}_{s_i}, v^{t'}_{s_i})} \cdot T^r - \sum_{(v^{t'}_{s_i}, v^{t''}_{s_k}) \in U_3 : t'' = t} x^{w_j}_{(v^{t'}_{s_i}, v^{t''}_{s_k})} \cdot T^r_{s_i s_k}$$
$$- \sum_{q_i \in Q : T^f_{q_i} = t} z^{w_j}_{q_i} \cdot T^r_{q_i} \qquad \forall t \in T \setminus \{1\}, \forall w_j \in W \quad (2.9)$$

$$h^t_{w_j} \geq h^{t-1}_{w_j} + \sum_{(v^{t'-1}_{s_i}, v^{t'}_{s_i}) \in U_2 : t' = t} x^{w_j}_{(v^{t'-1}_{s_i}, v^{t'}_{s_i})} \cdot T^r - \sum_{(v^{t'}_{s_i}, v^{t''}_{s_k}) \in U_3 : t'' = t} x^{w_j}_{(v^{t'}_{s_i}, v^{t''}_{s_k})} \cdot T^r_{s_i s_k}$$
$$- \sum_{u_{q_i} \in U_4 : T^f_{q_i} = t} z^{w_j}_{q_i} \cdot T^r_{q_i} - y^t_{w_j} \cdot B \qquad \forall t \in T \setminus \{1\}, \forall w_j \in W \quad (2.10)$$

$$h^t_{w_j} \leq R_{w_j} \qquad \forall t \in T, \forall w_j \in W \quad (2.11)$$

$$y^t_{w_j} > (h^{t-1}_{w_j} + \sum_{(v^{t-1}_{s_i}, v^t_{s_i}) \in U_2} x^{w_j}_{(v^{t-1}_{s_i}, v^t_{s_i})} \cdot T^r - R_{w_j}) \div B$$
$$\forall t \in T \setminus \{1\}, \forall w_j \in W \quad (2.12)$$

$$(1 - y^t_{w_j}) \geq -(h^{t-1}_{w_j} + \sum_{(v^{t-1}_{s_i}, v^t_{s_i}) \in U_2} x^{w_j}_{(v^{t-1}_{s_i}, v^t_{s_i})} \cdot T^r - R_{w_j}) \div B$$
$$\forall t \in T \setminus \{1\}, \forall w_j \in W \quad (2.13)$$

$$h^1_{w_j} = R'_{w_j} \qquad \forall w_j \in W \tag{2.14}$$

L'équation (2.5) garantit la continuité d'un chemin pour chaque véhicule en imposant que la somme des arcs entrant d'un chemin à un sommet du graphe égale celle des sortants. Les équations (2.6) et (2.7) font en sorte que, si un véhicule w_j est déplacé par un agent ou utilisé par un client, alors la variable x^{w_j} est égale à 1. L'équation (2.8) permet d'interdire que le nombre de véhicules stationnés dans une station entre deux périodes de temps dépasse la capacité de la station.

Les équations de (2.9) à (2.14) permettent de formuler la contrainte de recharge électrique *normale*. Pour chaque véhicule qui se déplace entre deux différentes stations, les contraintes (2.9) et (2.10) s'assurent que l'autonomie de chaque voiture $h^t_{w_j}$ à la période t est égale à l'autonomie de la même voiture $h^{t-1}_{w_j}$ à la période $t-1$, plus une recharge T^r si la voiture est restée dans une station entre les périodes $t-1$ et t, ou bien moins la quantité de charge consommée par le déplacement de la voiture représenté par un arc dont l'extrémité finale est le sommet $v^t_{s_i}$. La contrainte d'égalité se transforme en une contrainte d'infériorité seulement dans le cas où la voiture a atteint l'autonomie maximale R_{w_j}, ce qui correspond à $y^t_{w_j}$ égale à 1 et multiplié par un grand nombre négatif $-B$ et rend l'inégalité réalisable dans tout les cas. L'équation (2.11) force la variable $h^t_{w_j}$ à être inférieure ou égale à R_{w_j}. L'équation (2.12) force la variable $y^t_{w_j}$ à être égale à 1 lorsque la voiture w_j a atteint la charge maximale R_{w_j} et l'équation (2.13) à 0 dans le cas contraire. La dernière contrainte (2.14) permet d'initialiser à $t = 1$ la variable $h^1_{w_j}$ avec la charge initiale de la voiture w_j.

Complexité

Nous voulons prouver que le problème *ROCSP* défini avec une contrainte de recharge *normale* est *NP-difficile*. Pour ce faire, on réduit un problème *NP-difficile*, connu dans la littérature, au problème que l'on veut prouver au moins *NP-difficile*. Le problème du sac à dos ou *knapsack* prouvé *NP-difficile* (Kellerer et al., 2004) a été choisi pour faire la réduction, l'idée générale de la preuve est de prendre une instance quelconque du problème *knapsack* qu'on veut réduire, et la transformer, en temps polynomial, en une instance du *ROCSP* qu'on cherche à démontrer *NP*-difficile.

Les données du *knapsack* se résument à un sac avec une capacité W^{knp} et un ensemble d'objets O^{knp}. À chaque objet $o^{knp}_i \in O^{knp}$, sont associées, i le numéro

d'objet, un poids w_i^{knp} et un prix p_i^{knp}. L'objectif est de maximiser le prix total des objets à mettre dans le sac (fonction objectif) sans dépasser la capacité du sac. Le problème est formulé avec le modèle $PLNE^{knapsack}$ avec x_i^{knp}, la variable qui permet de sélectionner ou pas l'objet o_i.

$$\max \sum_{o_i^{knp} \in O^{knp}} p_i^{knp} \cdot x_i^{knp} \qquad (PLNE^{knapsack})$$

$$\sum_{o_i^{knp} \in O^{knp}} x_i^{knp} \cdot w_i^{knp} \leq W^{knp} \qquad (2.15)$$

Dans le but de simplifier la réduction, un Cas Particulier CP^{ROCSP} du problème $ROCSP$ est utilisé. Ce cas est défini par l'ensemble des suppositions suivantes :

— l'ensemble des stations est réduit à une seule station : $S = \{s_1\}$, avec $K_{s_1} = 1$;

— l'ensemble des voitures est réduit à une seule voiture : $W = \{w_1\}$;

— le taux de charge initial de la voiture w_1 au début de la journée R'_{w_1} est strictement inférieur à l'autonomie totale de la voiture R_{w_1} : $R'_{w_1} < R_{w_1}$;

— nous supposons que la voiture ne se recharge pas à la station s_1 : $T^r = 0$;

— les coefficients de la fonction objectif α et γ sont fixés respectivement à 0 et 1.

Pour modéliser le CP^{ROCSP} défini ci-dessus, le modèle $R\text{-}PLNE^{ROCSP}$ est simplifié en plusieurs étapes :

1. La contrainte (2.8) peut être supprimée du modèle du CP^{ROCSP}, car elle est satisfaite dans tous les cas. Cela est dû au fait qu'il n'y a qu'une seule voiture $W = \{w_1\}$ et une place de stationnement dans la station s_1 ($K_{s_1} = 1$) ;
2. D'après la contrainte (2.14), on a $h^1_{w_1} = R'_{w_1}$, ce qui implique $h^1_{w_1} < R_{w_1}$ et donc $y^2_{w_1} = 0$ selon la contrainte (2.13), vu que $T^r = 0$.
3. Selon la contrainte (2.9), on a : $h^2_{w_1} \leq h^1_{w_1} - \sum_{q_i \in Q : T^f_{q_i} = 2} z^{w_1}_{q_i} \cdot T^r_{q_i}$ car $T^r = 0$ et $U_3 = \emptyset$, cela implique $h^2_{w_1} \leq h^1_{w_1} < R_{w_1}$, ce qui fait que $y^3_{w_1} = 0$ selon la contrainte (2.13) relative à $t = 3$.
4. En répétant l'étape 2 jusqu'à T^{max}, on obtient $y^t_{w_1} = 0 \; \forall t \in T \setminus \{1\}$ et $h^t_{w_1} < R_{w_1} \; \forall t \in T$;

5. Comme $h^t_{w_1} < R_{w_1} \forall t \in T$, la contrainte (2.11) peut être supprimée du modèle CP^{ROCSP} car elle est satisfaite dans tous les cas. Même chose pour les contraintes (2.12) et (2.13) car $(h^{t-1}_{w_1} - R_{w_1}) < 0$ et $y^t_{w_1} = 0 \; \forall t \in T \setminus \{1\}$;
6. On a $y^t_{w_1} = 0 \; \forall t \in T \setminus \{1\}$, $T^r = 0$ et $U_3 = \emptyset$, ce qui implique que la contrainte (2.9) est équivalente à la contrainte (2.16), la contrainte (2.10) est équivalente à la contrainte (2.17). Les deux contraintes (2.16) et (2.17) peuvent être remplacées par la contrainte équivalente (2.18) ;

$$h^t_{w_1} \leq h^{t-1}_{w_1} - \sum_{q_i \in Q : T^f_{q_i} = t} z^{w_1}_{q_i} \cdot T^r_{q_i} \qquad \forall t \in T \setminus \{1\} \tag{2.16}$$

$$h^t_{w_1} \geq h^{t-1}_{w_1} - \sum_{q_i \in Q : T^f_{q_i} = t} z^{w_1}_{q_i} \cdot T^r_{q_i} \qquad \forall t \in T \setminus \{1\} \tag{2.17}$$

$$h^t_{w_1} = h^{t-1}_{w_1} - \sum_{q_i \in Q : T^f_{q_i} = t} z^{w_1}_{q_i} \cdot T^r_{q_i} \qquad \forall t \in T \setminus \{1\} \tag{2.18}$$

7. La contrainte (2.18) implique la contrainte (2.19) qui est obtenue en remplaçant à chaque fois chaque $h^t_{w_1} : t \in T \setminus \{1\}$ par $h^{t-1}_{w_1} - \sum_{q_i \in Q : T^f_{q_i} = t} z^{w_j}_{q_i} \cdot T^r_{q_i}$ dans la contrainte (2.18) relative à T^{max}. La simplification de la contrainte (2.19) donne la contrainte (2.20). Cette dernière implique la contrainte (2.21) car $h^{T^{max}}_{w_1} \geq 0$ et $h^1_{w_1} = R'_{w_1}$;

$$h^{T^{max}}_{w_1} = h^1_{w_1} - \sum_{t' \in T \setminus \{1\}} \sum_{q_i \in Q : T^f_{q_i} = t'} z^{w_1}_{q_i} \cdot T^r_{q_i} \tag{2.19}$$

$$\sum_{q_i \in Q} z^{w_1}_{q_i} \cdot T^r_{q_i} = h^1_{w_1} - h^{T^{max}}_{w_1} \tag{2.20}$$

$$\sum_{q_i \in Q} z^{w_1}_{q_i} \cdot T^r_{q_i} \leq R'_{w_1} \tag{2.21}$$

8. Les coefficients $\alpha = 0$ et $\gamma = 1$ sont remplacés dans la fonction objectif par leurs valeurs ce qui donne : $\min Ct = -\sum_{q_i \in Q} G_{q_i} \cdot z^{w_1}_{q_i}$ car $U_3 = \emptyset$, ensuite la fonction objectif est transformée en une fonction de maximisation en la multipliant par -1 ce qui donne : $\max \sum_{q_i \in Q} G_{q_i} \cdot z^{w_1}_{q_i}$.

L'ensemble des transformations permet de modéliser le CP^{ROCSP} à l'aide du modèle $PLNE - CP^{ROCSP}$.

$$\max \sum_{q_i \in Q} G_{q_i} \cdot z^{w_1}_{q_i} \qquad (PLNE - CP^{ROCSP})$$

sous contraintes de (2.22) à (2.27).

$$\sum_{q_i \in Q} z_{q_i}^{w_1} \cdot T_{q_i}^r \leq R'_{w_1} \tag{2.22}$$

$$\sum_{v_{s_1}^t \in V} x_{(v_{s_1}^t, v_{s_1}^{t'})}^{w_1} + \sum_{q_i \in Q, T_{q_i}^f = t'} z_{q_i}^{w_1} = \sum_{v_{s_1}^{t''} \in V} x_{(v_{s_1}^{t'}, v_{s_1}^{t''})}^{w_1} + \sum_{q_i \in Q, T_{q_i}^d = t'} z_{q_i}^{w_1}$$

$$\forall v_{s_1}^{t'} \in V \setminus \{V^d, V^f\} \tag{2.23}$$

$$x^{w_1} \geq x_{(v_{s_1}^t, v_{s_1}^{t'})}^{w_1} \qquad \forall (v_{s_1}^t, v_{s_1}^{t'}) \in U \tag{2.24}$$

$$x^{w_1} \geq z_{q_i}^{w_1} \qquad \forall q_i \in Q \tag{2.25}$$

$$h_{w_1}^t = h_{w_1}^{t-1} - \sum_{q_i \in Q : T_{q_i}^f = t} z_{q_i}^{w_1} \cdot T_{q_i}^r \qquad \forall t \in T \setminus \{1\} \tag{2.26}$$

$$h_{w_1}^1 = R'_{w_1} \tag{2.27}$$

Pour réduire le modèle $PLNE^{knapsack}$ au modèle $PLNE - CP^{ROCSP}$, des équivalences entre les variables et les données des deux modèles sont définies : $z_{q_i}^{w_1} = x_i^{knp}$; $R'_{w_1} = W^{knp}$; $T^{max} = \sum_{o_i^{knp} \in O^{knp}} w_i^{knp}$; $|Q| = |O^{knp}|$; $G_{q_i} = p_i^{knp}$; $T_{q_i}^r = w_i^{knp}$; $T_{q_i}^d = \sum_{o_k^{knp} \in O^{knp} : k < i} w_k^{knp}$; $T_{q_i}^f = \sum_{o_k^{knp} \in O^{knp} : k \leq i} w_k^{knp}$; $S_q^d = 1$; $S_q^f = s_1$.

Remplacer les variables et les données du modèle $PLNE - CP^{ROCSP}$ par celles du modèle $PLNE^{knapsack}$, permet de conclure que la solution optimale des modèles est la même car :

— les fonctions objectif des deux modèles sont exactement les mêmes ;

— la contrainte (2.22) est exactement la même que la contrainte (2.15) ;

— il suffit que $x^{w_1} = 1$ pour que les contraintes (2.24) et (2.25) soient satisfaites dans tous les cas et comme x^{w_1} n'intervient ni dans la fonction objectif, ni dans une autre contrainte, alors ça ne change rien à la solution optimale.

— pour que la contrainte (2.23) soit satisfaites dans tous les cas, il suffit que :

$$\forall q_i \in Q : z_{q_i}^{w_1} = 0 \quad \Rightarrow \forall (v_{s_1}^t, v_{s_1}^{t+1}) \in U_2 : T_{q_i}^d \leq t < T_{q_i}^f \quad x_{(v_{s_1}^t, v_{s_1}^{t+1})}^{w_1} = 1$$

et

$$\forall q_i \in Q : z_{q_i}^{w_1} = 1 \quad \Rightarrow \forall (v_{s_1}^t, v_{s_1}^{t+1}) \in U_2 : T_{q_i}^d \leq t < T_{q_i}^f \quad x_{(v_{s_1}^t, v_{s_1}^{t+1})}^{w_1} = 0.$$

et

$$x_{(V^d, v_{s_1}^1)}^{w_1} = 1 \wedge x_{(v_{s_1}^{T^{max}}, V^f)}^{w_1} = 1.$$

Comme les $x^{w_1}_{(v^t_{s_i}, v^{t'}_{s_k})}$ n'intervient ni dans la fonction objectif ni dans une autre contrainte alors ça ne change rien à la solution optimale ;

— pour que la contrainte (2.26) soit satisfaite dans tous les cas, il suffit que $h^t_{w_1} = R'_{w_1} - \sum_{q_i \in Q : T^f_{q_i} \leq t} z^{w_1}_{q_i} \cdot T^r_{q_i} \quad \forall t \in T \setminus \{1\}$, comme $h^t_{w_1}$ n'intervient ni dans la fonction objectif ni dans une autre contrainte alors ça ne change rien à la solution optimale ;

Pour résumer, les deux modèles $PLNE^{knapsack}$ et $PLNE - CP^{ROCSP}$ ont la même fonction objectif et la contrainte (2.22) est équivalente à la contrainte (2.15). Quant aux contraintes de (2.23) à (2.27), elles ne contraignent pas la solution optimale du $PLNE - CP^{ROCSP}$, ce qui fait que les deux modèles donnent la même solution optimale. Par conséquent, n'importe quelle instance du problème du sac à dos peut être réduite à une instance du *ROCSP* et résolue avec le modèle $PLNE - CP^{ROCSP}$. L'algorithme 1 permet de faire la réduction en un temps polynomial. En effet, l'algorithme contient une seule boucle qui a $|O^{knp}|$ comme nombre d'itérations, ce qui fait que l'algorithme a une complexité polynomiale.

Algorithm 1: Algorithme de réduction

Input: Une instance du sac à dos (*knapsack*)
Output: Une instance du *ROCSP*

$S \leftarrow \{s_1\}$; $K_{s_1} = 1$; $\alpha = 0$; $\gamma = 1$; $T^r = 0$; $R'_{w_1} = W^{knp}$;
$T^{max} \leftarrow 0$;
for $o^{knp}_i \in O^{knp}$ **do**
    $Q \leftarrow Q \cup \{q\} : S^d_{q_i} = 1 S^f_{q_i} = 1, T^d_{q_i} = T^{max}, T^f_{q_i} = T^{max} + w^{knp}_i, T^r_{q_i} = w^{knp}_i, G_{q_i} = p^{knp}_i$;
    $T^{max} = T^{max} + w^{knp}_i$;

Proposition 2.4.1. *ROCSP avec contrainte de recharge normale est de complexité NP-difficile.*

Démonstration. N'importe quelle instance du problème du sac à dos est transformable en un temps polynomial en une instance du problème *ROCSP* en utilisant l'algorithme 1. □

2.4.4 Comparaison des deux modèles (*flots vs chemins*)

La résolution des deux modèles est un problème de complexité *NP-difficile*. En revanche, le modèle *F-PLNE*^{ROCSP} a un nombre beaucoup moins important de variables de décision. En effet, ce dernier compte $|U|$ variables alors que le modèle *R-PLNE*^{ROCSP} compte $(|U| + T^{max} \cdot 2) \cdot W^{max}$ variables, ce qui fait qu'il est potentiellement plus difficile à résoudre. Cependant, le modèle à base de chemins permet une abstraction moins grossière de la réalité des contraintes de recharge, surtout quand la flotte de voitures n'est pas hétérogène avec des consommations et autonomies électriques qui peuvent varier d'une voiture à une autre. En effet, la prise en compte de l'hétérogénéité de la flotte représente un paramètre important dans les problèmes de routage dans le transport, par exemple, Liu et al. (2017) abordent le problème d'horaires dans un réseau de transport en commun, en intégrant une contrainte de consommation de carburant qui varie selon les véhicules. Ajoutons à cela que dans certains services d'auto-partage, le client peut demander un type de voiture particulier comme par exemple des voitures pour handicapés ou des voitures pour le transport de marchandises.

2.5 Modélisation PLNE du *ESRP*

Le problème de planning et de routage des agents de relocalisation *ESRP* (Employee Scheduling and Routing Problem) consiste à affecter de manière optimale l'ensemble des opérations de relocalisation des voitures à un ensemble d'agents, en respectant leurs horaires de travail. Le problème est *NP-difficile* car il peut être vu comme une généralisation d'un autre problème *NP-difficile*, à savoir le *Airline crew scheduling problem* (Deng et Lin, 2011), où il s'agit d'affecter des vols aux pilotes tout en respectant leurs contraintes horaires. Les agents de relocalisation peuvent être vus comme les pilotes et les opérations de relocalisation comme les vols, sauf que les pilotes ne peuvent pas se déplacer entre les aéroports sans effectuer un vol alors que les agents peuvent prendre les transports publics, par exemple, pour se déplacer entre deux stations sans déplacer des voitures.

2.5.1 Graphe modélisant l'*ESRP*

La modélisation graphique a pour but de transformer l'*ESRP* du problème de planning des employés en un problème de routage d'agents de relocalisation

en contractant chaque opération de relocalisation en un sommet du graphe. Soit $G^{ESRP}(V', U', R')$, le graphe modélisant le problème *ESRP*, où :

— L'ensemble des sommets $V' = \{v_{o_i} : o_i \in O\} \cup \{V^{d'} = v_{o_0}, V^{f'} = v_{o_{O^{max}+1}}\}$ où chaque sommet v_{o_i} est tel que o_i représente une opération de relocalisation ; les sommets $V^{d'}$ et $V^{f'}$ des opérations fictives qui servent de source et de destination dans le graphe et qui représentent respectivement le début et la fin de la journée de travail de l'agent.

— L'ensemble des arcs $U' = U'_1 \cup U'_2$ où :

- $U'_1 = \{(V^{d'}, v_{o_i}), (v_{o_i}, V^{f'}) : v_{o_i} \in V'\}$ où chaque arc dans U'_1 représente un lien entre les sommets fictifs $V^{d'}$ et $V^{f'}$ et les autres sommets représentent les opérations de relocalisation ;
- $U'_2 = \{(v_{o_i}, v_{o_k}) : v_{o_i}, v_{o_k} \in V', T^d_{o_k} \geq T^f_{o_i} + T'_{S^f_{o_i} S^d_{o_k}}\}$ où chaque arc est un lien entre deux opérations de relocalisation qui peuvent être planifiées dans la tournée du même agent. En effet, pour que deux opérations o_i et o_k puissent être affectées au même agent dans l'ordre, il faut que le temps de début $T^d_{o_k}$ de l'opération o_k soit supérieur ou égal au temps d'arrivée $T^f_{o_i}$ de l'opération o_i, plus la distance en temps qui sépare les deux opérations.

— L'ensemble des consommations de ressource des arcs R' contient le nombre de périodes de temps que consomme chaque opération de relocalisation du temps de travail de l'agent de relocalisation. En effet, chaque agent a un nombre maximal d'heures de travail par jour. de ce fait, nous définissons une consommation de ressources dans chaque arc comme le temps nécessaire pour se rendre à la station de l'opération de relocalisation en plus de la durée de l'opération de relocalisation et le temps d'attente entre les deux. L'ensemble des consommations de ressources $R' = R'_1 \cup R'_2$ où :

- $R'_1 = \{r_{(V^{d'}, v_{o_i})} = T^f_{o_i} - T^d_{o_i}, r_{(v_{o_i}, V^{f'})} = 0 : (V^{d'}, v_{o_i}), (v_{o_i}, V^{f'}) \in U'_1\}$;
- $R'_2 = \{r_{(v_{o_i}, v_{o_k})} = T^f_{o_k} - T^f_{o_i} : (v_{o_i}, V^{f'}) \in U'_2\}$.

2.5.2 Modèle PLNE pour l'*ESRP*

En se basant sur le graphe G^{ESRP}, nous proposons un modèle de programmation linéaire $PLNE^{ESRP}$ pour le problème de planning des employés, avec deux vecteurs de variables de décision y_{a_j} et $x^{a_j}_{(v_{o_i}, v_{o_k})}$:

— $y_{a_j} \in \{0, 1\}$: égale à 1, si l'agent a_j est déployé, 0 sinon.

— $x^{a_j}_{(v_{o_i},v_{o_k})} \in \{0,1\}$: égale à 1, si l'agent a_j exécute l'opération de relocalisation o_k, juste après avoir exécuté l'opération de relocalisation o_i, 0 sinon.

La fonction objectif est formulée comme suit :

$$\min \quad \{Ct' = \delta \cdot C_A + \omega \cdot C_D\} \qquad (PLNE^{ESRP})$$

avec :

— $C_A = \sum_{a_j \in A} y_{a_j}$

— $C_D = \sum_{a_j \in A} \sum_{(v_{o_i},v_{o_k}) \in U'} r_{(v_{o_i},v_{o_k})} \cdot x^{a_j}_{(v_{o_i},v_{o_k})}$

sous les contraintes (2.28) à (2.31).

$$\sum_{a_j \in A} \sum_{v_{o_k} \in V'} x^{a_j}_{(v_{o_k},v_{o_i})} = 1 \qquad \forall v_{o_i} \in V' \setminus \{V^{d'}, V^{f'}\} \qquad (2.28)$$

$$\sum_{v_{o_k} \in V'} x^{a_j}_{(v_{o_k},v_{o_i})} - \sum_{v_{o_k} \in V'} x^{a_j}_{(v_{o_i},v_{o_k})} = 0$$

$$\forall v_{o_i} \in V' \setminus \{V^{d'}, V^{f'}\}, \forall a_j \in A \qquad (2.29)$$

$$\sum_{(V^{d'},v_{o_i}) \in U'} x^{a_j}_{(V^{d'},v_{o_i})} \leq y_{a_j} \qquad \forall a_j \in A \qquad (2.30)$$

$$\sum_{(v_{o_i},v_{o_k}) \in U'} r_{(v_{o_i},v_{o_k})} \cdot x^{a_j}_{(v_{o_i},v_{o_k})} \leq R_{a_j} \qquad \forall a_j \in A \qquad (2.31)$$

On peut observer que les contraintes du $ESRP$ sont assez similaires à l'ensemble des contraintes du VRP classique, où l'équation (2.28) permet la satisfaction de toutes les opérations de relocalisation. L'équation (2.29) assure la continuité du chemin en imposant que la somme des arcs entrant à un sommet soit égale à la somme des arcs sortants. L'équation (2.30) vérifie qu'un seul chemin au maximum est associé à chaque agent en imposant que la somme des chemins qui sortent du sommet source $V^{d'}$ correspondant au même agent soit inférieure ou égale à la variable indiquant si l'agent est utilisé ou pas. L'équation (2.31) vérifie que chaque agent ne travaille pas plus que le nombre maximal d'heures de travail.

2.6 Instances et modèles pour la génération des données

Après avoir décrit et modélisé le problème d'optimisation dans l'auto-partage *à un seul sens*, et avant de passer à sa résolution, ce chapitre nous indique comment sont générées les instances avec lesquelles les simulations sont effectuées dans le cadre de cette thèse. Comme principal cas d'étude, le service d'auto-partage de la ville de Nice (64 stations) sera utilisé pour la génération d'un premier paquet d'instances théoriques, afin d'évaluer les performances du système. En complément, pour évaluer nos algorithmes, d'autres instances théoriques d'auto-partage générées à partir d'instances de vélopartage avec des villes allant de 13 à 116 stations seront aussi utilisées.

2.6.1 Simulation d'auto-partage : instances pour le site de Nice (Auto Bleue)

L'auto-partage Auto Bleue de la métropole de Nice permet de louer des véhicules en libre service à Nice et dans ses proches environs. Tous les véhicules sont électriques, disposent donc d'une autonomie limitée et nécessitent des rechargements fréquents. 64 stations (figure 2.5) sont actuellement localisées sur ce territoire par le service Auto Bleue (Auto-bleue, 2017).

Il nous est actuellement impossible de disposer des informations clients sur le service Auto Bleue. De ce fait, et également afin d'étalonner et d'évaluer la méthode proposée, avec l'aide de géomaticiens, nous avons généré des instances aléatoires pour simuler les flux origine-destination à partir de la localisation des stations du service niçois. La matrice des distances en temps entre les stations a été calculée sur le réseau avec un système d'information géographique, en prenant en compte l'orientation (topologie viaire avec sens uniques) et la valuation (frictions inverses aux vitesses autorisées sur les différentes sections de route) des arêtes du graphe. Pour créer des flux origine-destination, nous procédons en deux étapes.

Première étape : génération des attractivités des stations en fonction de leur bassin de chalandise local

Tout d'abord, nous créons une partition à partir de polygones de Voronoï, où chaque station possède une aire de chalandise bornée par celles de ses voisines (figure 2.6). Un autre intérêt est l'obtention d'une partition spatiale où

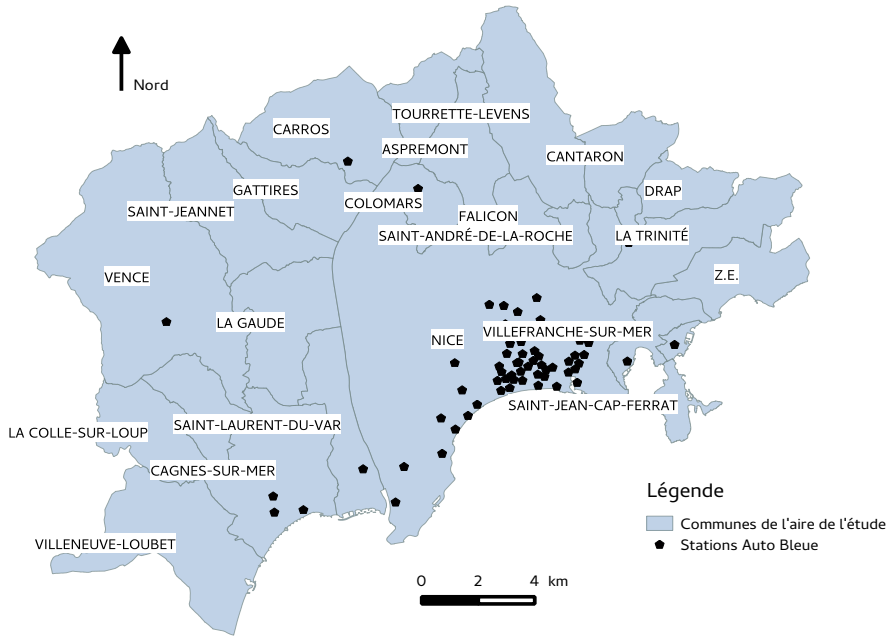


FIGURE 2.5 – *Localisation des stations d'auto-partage Auto Bleue dans les environs de Nice ; sources : IGN, Auto Bleue*

ne persiste aucun trou, ni aucune superposition d'aires de chalandise contiguës. Considérant ensuite deux distances d'accès aux stations à pied (300 et 500 mètres), nous avons réalisé des zones tampons à vol d'oiseau autour des stations (figure 2.7). Avec cette approche, nous simulons seulement un usage de proximité du service, faisant abstraction d'un éventuel usage de modes intermodaux (véhicule personnel, bus, etc.) pour rejoindre les stations.

L'intersection des zones tampons avec les polygones de Voronoï nous fournit une zone de chalandise par station qui pour l'instant ne comptabilise pas de quantité de population (figure 2.8). Chacune de ces aires est alors croisée avec les données de population issues du carroyage de l'INSEE (maille de côté de 200 m, *cf.* figure 2.9 et figure 2.10). Finalement, par une requête spatiale à chaque station, nous calculons la somme des populations incluses dans l'aire de chalandise (figure 2.11). Cette valeur d'attractivité permet d'estimer une probabilité de départ et d'arrivée de véhicules d'auto-partage pour une station donnée et sert de base à la création des instances.

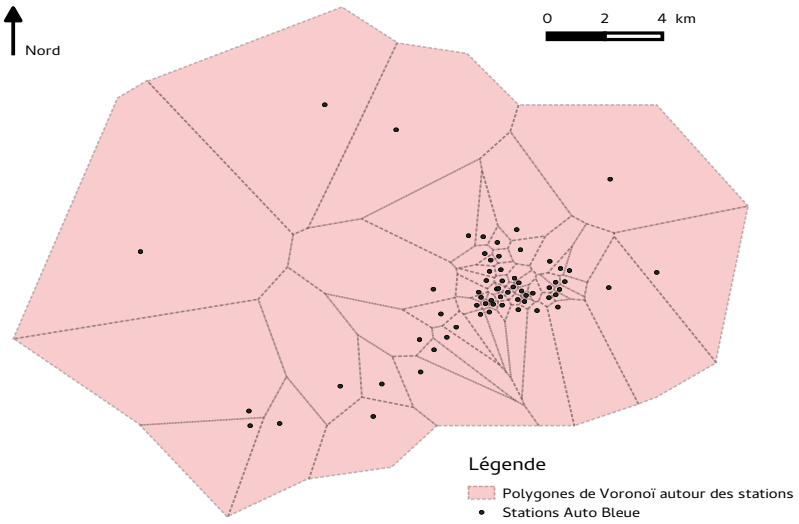


FIGURE 2.6 – *Partitionnement spatial basé sur des polygones de Voronoï autour des stations : chaque polygone ne contient qu'une station. Source : (Ait-Ouahmed et al., 2018)*

Seconde étape : génération des flux en fonction de l'attractivité des stations suivant un modèle gravitaire

Pour créer nos instances aléatoires sous contrainte de probabilité, nous prenons tout d'abord en compte les attractivités des stations. Plus l'attractivité de la station est grande et plus la probabilité que ses véhicules soient utilisés est importante. À partir de cette attractivité, nous générons une série d'instances en utilisant un modèle gravitaire qui est destiné à prévoir la géographie des flux ou des interactions (Camagny, 1996; Josselin et Nicot, 2003). Le modèle gravitaire utilisé est une généralisation de la loi de la gravitation universelle de Newton très utilisée en géographie : deux corps s'attirent selon leurs masses respectives et inversement à la distance (au carré) qui les sépare. Dans notre cas, les masses (points d'origine ou de destination) et la distance sont affectées d'un exposant paramétrable. La répartition des interactions entre les stations dépend de la force d'attraction de chacune et de la difficulté à se déplacer de l'une à l'autre. Dans notre cas, ce modèle est utilisé pour estimer les flux en fonction des distances relatives en temps entre les stations. Ces distances ont été calculées sur le réseau en fonction des vitesses commerciales autorisées. Ce modèle a été choisi sur la base de deux hypothèses :

— au vu des informations dont nous disposions sur les courtes distances

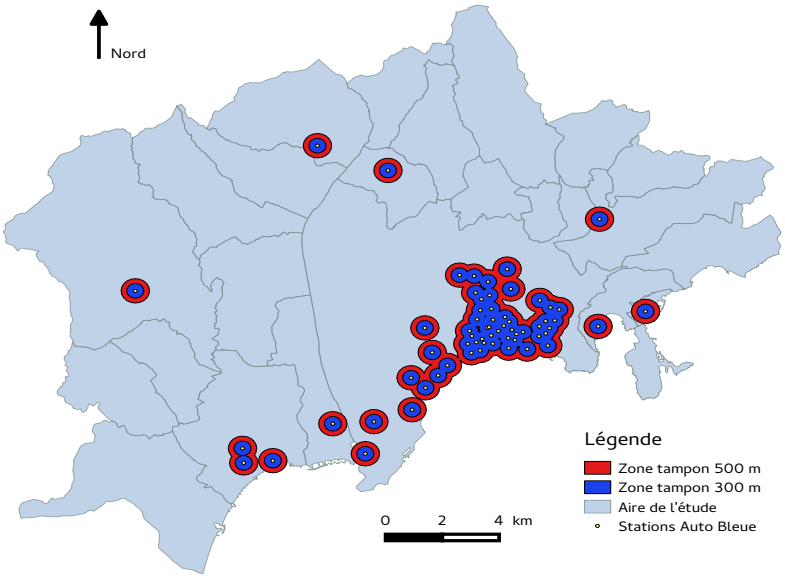


FIGURE 2.7 – *Zones tampons de 300 et 500 mètres autour des stations. Source : (Ait-Ouahmed et al., 2018)*

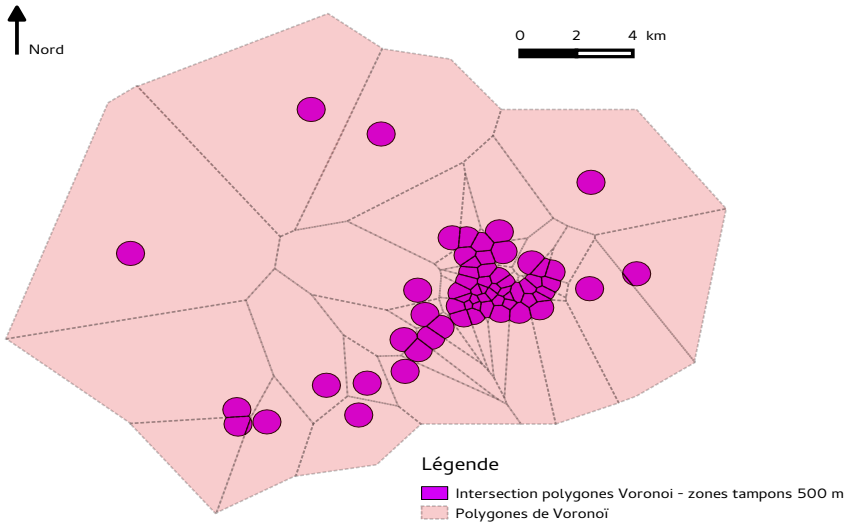


FIGURE 2.8 – *Intersection entre la partition de Voronoï et la zone tampon de 500 mètres autour des stations. Source : (Ait-Ouahmed et al., 2018)*

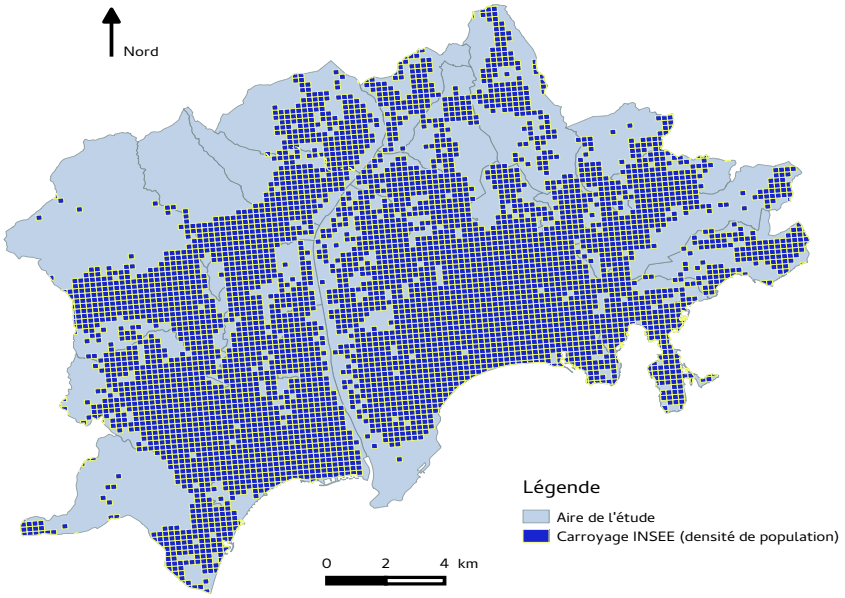


FIGURE 2.9 – *Données INSEE de population par carreaux de 200 m par 200 m ; source : INSEE*

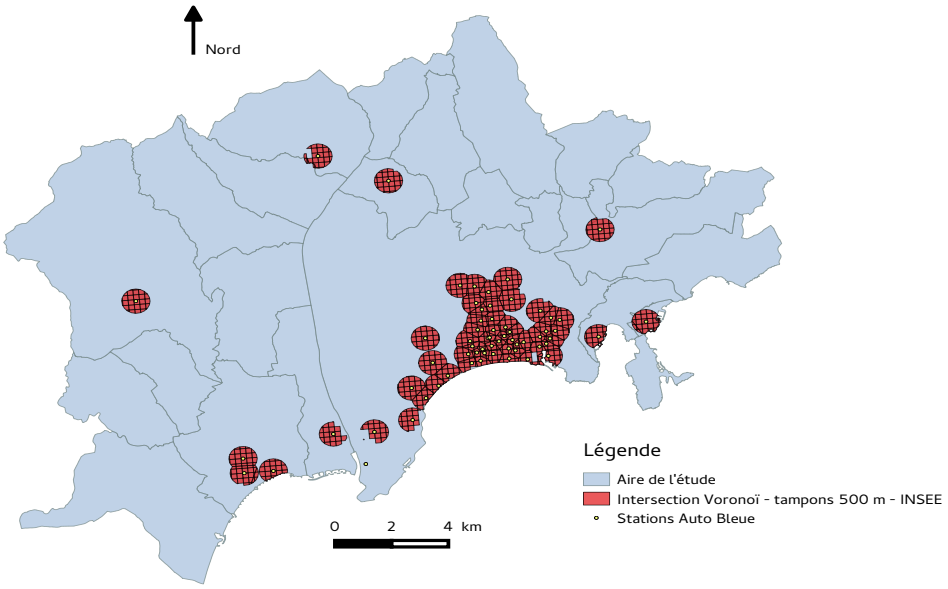


FIGURE 2.10 – *Intersection entre la partition de Voronoï, la zone tampon de 500 mètres autour des stations et le carroyage de l'INSEE. Source : (Ait-Ouahmed et al., 2018)*

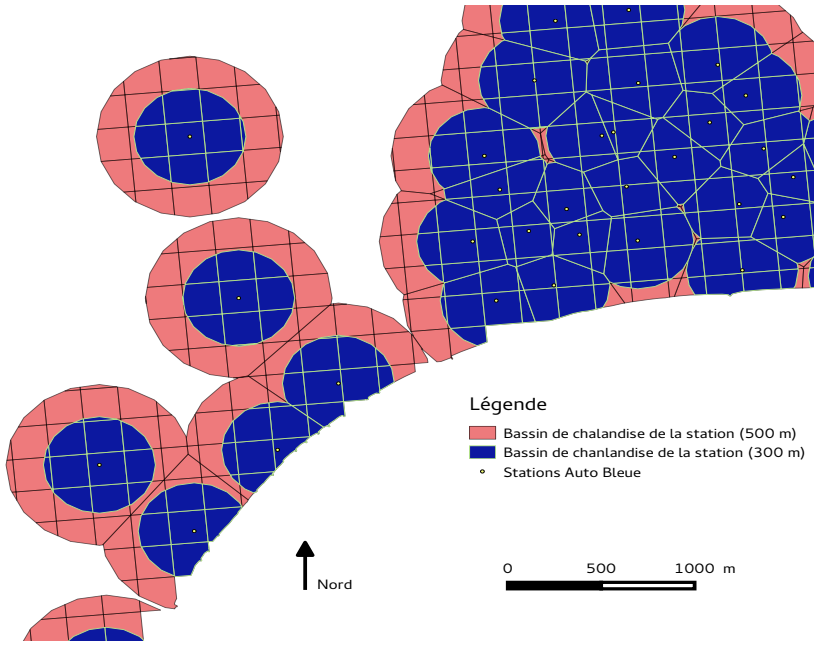


FIGURE 2.11 – *Zoom sur quelques zones de chalandise obtenues à 300 et 500 mètres des stations : la population recensée par l'INSEE dans chaque bassin de chalandise est agrégée et rapportée à chaque station. Source : (Ait-Ouahmed et al., 2018)*

moyennes de location de véhicules d'auto-partage (moins de 10 km), faisant de l'auto-partage un service «de proximité» ;

— du fait qu'il est plus probable que le client rapporte la voiture là où il l'a prise.

Il s'agit donc de simuler des flux, compte tenu des bassins de chalandise des stations. Le cas de l'intermodalité n'est pas considéré mais peut constituer des travaux ultérieurs. Ainsi, le flux F_{ij} entre deux stations i et j est directement proportionnel au produit des masses M_i et M_j de chaque station et inversement proportionnel à la distance d_{ij} qui les sépare :

$$F_{i,j} = \frac{M_i^\theta M_j^\chi}{d_{i,j}^\phi}$$

Dans notre cas, la masse est estimée par la population captable dans l'aire de chalandise de la station. Les valeurs des exposants θ, χ et ϕ sont respectivement fixées à 1, 1 et 2, comme c'est souvent le cas pour ce type de modèle. Le flux est donc symétrique et l'effet inverse de la distance est important. Le tirage aléatoire se fait alors sur un échantillon d'Origines-Destinations, dont les valeurs

de probabilité sont issues du calcul des flux théoriques gravitaires. Avec cette méthode, on suppose que pour les demandes de location des clients, plus deux stations sont proches et plus la probabilité qu'elles soient un couple (origine, destination) est grande. Pour une même station, on considère une distance arbitraire de 200 m qui donne une probabilité importante de retour à la même station. Cette manipulation est nécessaire à cause de l'impossibilité de division par zéro dans la formule du modèle gravitaire, qui constitue un des défauts majeurs du modèle. Notons que cette méthode produit une distribution des attractivités très dissymétrique, avec quelques stations grandement utilisées et bon nombre d'entre elles relativement peu sollicitées. En dépit des hypothèses fortes de ce modèle et des distances considérées, la distribution des distances peut correspondre à une certaine réalité (cependant non vérifiable concrètement), car ce sont dans les quartiers peuplés que se trouve la plus grande densité de stations proches les unes des autres. Ce n'est que lorsque nous pourrons disposer de données observées que notre modèle pourra être testé de façon plus fine et plus juste, pour ajuster éventuellement le poids de la distance (valeur des exposants des stations d'origine et de destination et de la distance).

2.6.2 Instances générées à partir d'instances de vélo-partage

En plus des instances de Nice, des instances d'auto-partage sont générées en exploitant des données d'instances de *BikeSharing* (Group, 2017) de vingt-deux villes offrant des services de partage de vélos, ces villes ont un nombre différent de stations, comme indiqué dans le tableau 2.4 . Les données utilisées pour la transformation sont :

— la matrice de distance entre les stations de chaque instance ;

— le déficit et le surplus de vélos enregistrés en fin de journée pour chaque station, qui nous servent pour générer une probabilité d'attractivité pour chaque station et donc générer les flux origine-destination théoriques entres les stations.

2.6.3 Paramètres des instances générées

En plus de l'ensemble des stations et les flux théorique entre les stations, d'autres paramètre sont fixés pour générer les instances de simulations :

— la journée de service : elle commence à 6 h du matin et se termine à minuit, ce qui représente 72 périodes de temps avec 15 min par chaque période de temps ;

— la capacité des stations k_{s_i} : pour les instances de Nice, nous disposons des capacités réelles des stations du service Auto Bleue qui sont de 5 places

TABLE 2.4 – *Nombre de stations de vélo partage (BikeSharing) par ville*

Ville	Nombre de stations
Bari	13 stations
Reggio d'Émilie	14 stations
Bergame	15 stations
Parme	15 stations
Trévise	18 stations
La Spezia	20 stations
Buenos Aires	21 stations
Ottawa	21 stations
San Antonio	23 stations
Brescia	27 stations
Rome	28 stations
Madison	28 stations
Guadalajara	41 stations
Dublin	45 stations
Denver	51 stations
Rio de Janeiro	55 stations
Boston	59 stations
Turin	75 stations
Toronto	80 stations
Miami	82 stations
Mexico	90 stations
Minneapolis	116 stations

par station mais dont 2 qui peuvent être également utilisées par des particuliers, ce qui a fait qu'on a généré des capacités aléatoires comprises entre 3 et 5 places pour les instances de Nice, ainsi que pour les autres instances issues du *BikeSharing* ;

— le nombre de voitures : dans la version statique du *ROCSP*, le nombre de voitures est un critère qu'on cherche à minimiser et donc non fixé, toutefois le nombre maximal est fixé à la somme des places des stations de l'instance. Par contre, pour la version dynamique du *ROCSP*, il ne constitue pas un critère d'optimisation mais un paramètre d'entrée fixé à 3 voitures dans les stations de 5 places et 2 voitures dans les stations de 3 à 4 places ;

— le nombre de requêtes par journée : pour les instances de Nice qui représentent notre principal cas d'étude, une gamme de demandes de clients (nombre de requêtes) allant de 50 jusqu'à 1000 requêtes a été générée. Pour les autres instances issues du *BikeSharing*, le nombre de requêtes générées est fixé à dix fois le nombre de stations dans l'instance, ce qui correspond à une moyenne de 10 requêtes par station et par jour. Le nombre de requêtes généré entre chaque deux stations est proportionnel au flux théorique calculé précédemment ;

— génération des horaires de départs des requêtes : pour générer des horaires aléatoires, on utilise une courbe de distribution théorique de la demande basée sur les pics de flux de «navetteurs» (trajets domicile-travail, voir figure 2.12). Même si dans la pratique, l'auto-partage peut-être utilisé pour d'autres motifs, cette courbe fournit une estimation des flux à

certains moments de la journée. On y identifie deux gros pics (un le matin et le second le soir), et deux petits pics en fin de matinée et en début d'après-midi. Outre qu'elle est plus réaliste qu'une distribution aléatoire uniforme, cette répartition permet de tester la robustesse du modèle, puisque les demandes sont concurrentes dans le temps (pics). De ce point de vue, les simulations sont réalisées dans des conditions peu favorables à l'optimisation. Quant à la durée de location des véhicules, elle est générée avec une moyenne d'une heure par requête, en se basant sur le rapport d'activité d'Autolib (Autolib, 2014), qui stipule que la durée moyenne est d'environ 45 min et qui a été arrondie à la hausse à 1 heure, soit 4 périodes de temps ;

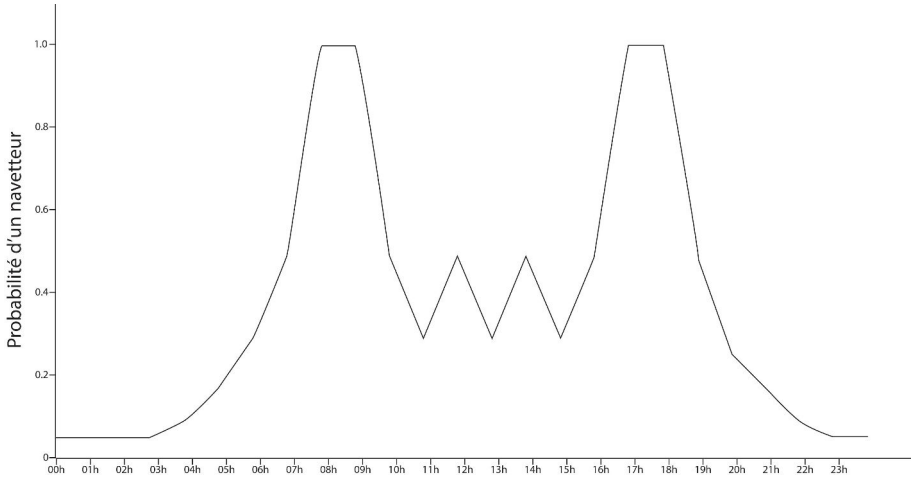


FIGURE 2.12 – *Fréquence horaire théorique des flux de navetteurs (sur une journée)*

— G_{q_i} : le gain que rapporte la satisfaction d'une requête q_i est fixé à la durée de location de la requête q_i, car nous supposons que le prix que paye le client est proportionnel au temps de location ; toutefois d'autres modèles de tarification peuvent être envisagés, comme par exemple fixer des prix plus intéressants aux clients, dont les requêtes correspondent aux relocalisations envisagées ;

— $T_{s_i s_k}$: le temps de trajet entre deux stations est calculé en utilisant la distance entre les deux stations et une vitesse de transformation de 30 km/h, qui correspond à la vitesse moyenne d'une voiture dans la ville de Paris quand la circulation est fluide ;

— $T'_{s_i s_k}$: le temps de trajet pour un agent qui se déplace entre deux stations en dehors des opérations de relocalisation est calculé en utilisant une vitesse de transformation de 25 km/h qui correspond à la vitesse moyenne dans les transports publics de la ville de Paris quand la circulation est fluide ; cette vitesse est réduite à 15 km/h pour prendre en compte les éventuels temps d'attente ;

— R_{w_j}, R'_{w_j} : l'autonomie électrique maximale ainsi que l'autonomie initiale électrique de toutes les voitures sont fixées à 8 heures, ce qui correspond à 240 km d'autonomie en utilisant la vitesse de transformation de 30 km/h ;
— $T^r_{q_i}$: la consommation en autonomie électrique de chaque requête est aussi fixée à la durée de location de la requête q_i, car on suppose le pire cas à savoir que le client roule durant toute sa durée de location, par exemple pour 1 heure de location il va rouler 30 km selon la vitesse moyenne, et 30 km correspondent aussi à une heure d'autonomie en moins ;
— $T^r_{s_i s_k}$: la consommation en autonomie électrique du trajet de relocalisation est aussi fixée à la durée du trajet $T_{s_i s_k}$ pour les mêmes raisons ;
— T^r : une voiture qui reste dans la même station pendant une période de temps augmente son autonomie électrique d'une période de temps, car il faut environ 8 heures pour qu'une voiture se recharge complètement dans une borne *normale* et l'autonomie des voitures est fixée à 8 heures, ce qui fait que le temps de recharge est égal au temps d'autonomie ;
— R_{a_j} : le temps de travail maximal des agents est aussi fixé à 8 heures.

2.6.4 Conditions de simulations

Toutes les simulations sont effectuées sur un PC avec CPU Intel Core i5 (7ème génération) 7200 U - 2.5 GHz et de 8 Go de RAM. Tous les algorithmes ont été implémentés en Java. L'outil d'optimisation CPLEX 12.2 est utilisé pour résoudre les formulations PLNE proposées.

2.7 Conclusion

L'auto-partage avec des véhicules électriques pour le transport dans un environnement urbain a été le sujet de ce chapitre. Les objectifs d'optimisation et les règles de fonctionnement d'un tel système ont été définis. Puis, des modélisations qui prennent en compte de nouvelles contraintes non modélisées jusqu'à présent dans des problèmes d'auto-partage, comme les contraintes de rechargement électrique, ont été proposées. La modélisation de ces nouvelles contraintes a donné lieu à une nouvelle variante du problème de routage de véhicules, prouvée comme *NP*-difficile et donc délicat à résoudre. Ce chapitre nous a aussi informé sur les instances d'auto-portage et les conditions de simulations qui seront déployées dans la suite de nos travaux. Le chapitre qui suit exposera des méthodes de résolution efficaces se confrontant à la complexité des modèles proposés. Elles seront testées et validées par une batterie de simulations effectuées sur les instances proposées dans ce chapitre.

Chapitre 3

Résolution du problème de relocalisation dans l'auto-partage *à un seul sens* avec réservations statiques

Sommaire

3.1 Introduction

Dans le chapitre précédent, des modèles PLNE ont été proposés pour la résolution exacte des problèmes *ROCSP* et *ESRP*. Ces modèles seront implémentés à l'aide du solveur CPLEX pour essayer de résoudre de manière exacte les instances d'auto-partage présentées dans 2.6. Mais vue la complexité des problèmes traités, nous nous attendons à être limités à des petites instances dans la résolution exacte. Pour espérer résoudre des instances de taille réelle, des heuristiques sont proposées dans ce chapitre pour une résolution approchée avec des temps raisonnables. Les solutions approchées sont évaluées par la suite, en les comparant avec les solutions exactes des modèles PLNE sur des petites instances. Pour finir, ces heuristiques sont utilisées pour étudier l'auto-partage dans le cas de la ville de Nice.

3.2 Résolution heuristique du *ROCSP*

Nous proposons une heuristique constructive qui permet de trouver une solution au *ROCSP*. Cette solution est construite de manière itérative en routant voiture par voiture et en utilisant un graphe résiduel pour soustraire les ressources (places de stationnement) correspondant aux voitures déjà routées dans une solution partielle. Aussi sont soustraits du graphe résiduel les arcs correspondant aux clients déjà satisfaits dans la solution partielle. Puis, un algorithme génétique basé sur cette heuristique constructive est proposé dans le but d'améliorer la qualité des solutions.

3.2.1 Heuristique Constructive : HC^{ROCSP}

Une Heuristique Constructive (HC^{ROCSP}) spécifique au *ROCSP* est proposée, tenant compte des objectifs, des règles et des contraintes évoqués précédemment dans le chapitre 2. Le but est de trouver des solutions approchées et réalisables dans un temps raisonnable de calcul. Nous commençons par présenter le principe général. Ensuite, nous détaillons un algorithme de plus court chemin avec contraintes de ressource utilisée à chaque itération de l'heuristique.

Principe et algorithme

Le principe de la méthode proposée consiste à utiliser le graphe spatio-temporel $G^{ROCSP}(V, U, R)$ présenté dans 2.4.1 pour router les voitures à travers la journée de service les unes après les autres. Pour ce faire, deux ensembles supplémentaires C et K sont associés au graphe spatio-temporel, ce qui donne le graphe $G^{ROCSP}(V, U, R, C, K)$ de telle sorte que :

— L'ensemble C des coûts associés aux arcs est calculé en utilisant les coefficients α, β et γ de la fonction objectif *Ct* du *ROCSP*, l'ensemble C se décompose en 4 sous ensembles,$C = C_1 \cup C_2 \cup C_3 \cup C_4$ où :

- $C_1 = \{c_{(V^d, v^1_{s_i})} = 0, c_{(v^{T^{max}}_{s_i}, V^f)} = 0 : (V^d, v^1_{s_i}), (v^{T^{max}}_{s_i}, V^f) \in U_1\}$; tous les coûts des arcs dans U_1 sont égaux à zéro, car chaque voiture est supposée pouvoir débuter la journée dans n'importe quelle station, dans la mesure où des relocalisations sont possibles la nuit pour démarrer la journée de service avec la distribution la plus optimale. Toutefois, ces coûts peuvent prendre des valeurs différentes de zéro, si on vise à optimiser aussi les relocalisations de la nuit, ce qui n'est pas le cas dans nos travaux ;
- $C_2 = \{c_{(v^t_{s_i}, v^{t+1}_{s_i})} = 0 : (v^t_{s_i}, v^{t+1}_{s_i}) \in U_2\}$, un coût égal à zéro est aussi affecté à chaque arc représentant une voiture qui reste dans la même station ;
- $C_3 = \{c_{(v^t_{s_i}, v^{t'}_{s_k})} = \beta \cdot T_{s_i s_k} : (v^t_{s_i}, v^{t'}_{s_k}) \in U_3\}$, chaque coût $c_{(v^t_{s_i}, v^{t'}_{s_k})}$ représente le coût d'une relocalisation entre la station s_i et s_k ;
- $C_4 = \{c_{u_{q_i}} = -\gamma \cdot G_{q_i} : u_{q_i} \in U_4\}$, chaque coût $c_{u_{q_i}}$ a une valeur négative et permet de quantifier le gain effectué pour la satisfaction d'une requête q_i.

— L'ensemble $K = \{k^t_{s_i} : (v^t_{s_i}, v^{t+1}_{s_i}) \in U_2\}$: chaque $k^t_{s_i}$ contient la capacité d'accueil résiduelle de la station s_i à chaque période de temps $t \in T$. Chaque capacité $k^t_{s_i}$ est initialisée à k_{s_i} (la capacité de la station s_i), et à chaque itération de notre heuristique, les capacités d'accueil résiduelles $k^t_{s_i} \in K$ sont mises à jour.

Pour chaque voiture, on cherche le meilleur chemin, à savoir celui qui satisfait un nombre maximal de clients avec un minimum de relocalisations et qui peut être ajouté à la solution partielle, sans violer les contraintes de capacité des stations. Un tel chemin correspond au plus court chemin dans le graphe $G^{ROCSP}(V, U, R, C, K)$ entre V^d et V^f. Plus précisément, pour chaque véhicule $w_j \in W$, nous calculons le meilleur chemin $Path_j$ à partir du sommet source V^d vers le sommet destination V^f, en utilisant l'algorithme de plus court chemin avec contraintes de ressource, présenté quelques lignes plus bas. À noter,

que le graphe spatio-temporel utilisé ne contient pas de cycle, ce qui nous permet d'utiliser des arcs de coûts négatifs sans avoir de cycle absorbant. Après le calcul de chaque chemin, on construit un graphe résiduel en mettant à jour l'ensemble U et en supprimant les arcs $(v^t_{s_i}, v^{t+1}_{s_i}) \in U_2$ tel que $k^t_{s_i} = 0$ (station pleine) ainsi que les arcs $u_{q_i} \in U_4$ représentant les demandes des utilisateurs desservis par la voiture w_j. Le pseudo code de l'heuristique proposée est décrit dans l'algorithme 2.

Algorithm 2: Pseudo code du HC^{ROCSP}

Data: $G^{ROCSP}(V, U, R, C, K)$;
Result: $Paths_{Cars}$, $Relocation$, $AcceptedQueries$
initialisation;
$Paths^{Cars} \leftarrow \emptyset$ /* l'ensemble de chemins qui constituent les trajectoires des voitures */ ;
$AcceptedQueries \leftarrow \emptyset$ /* l'ensemble de demandes satisfaites*/ ;
$Relocation \leftarrow \emptyset$ /*l'ensemble des relocalisations des voitures */ ;
$newAQuerie \leftarrow true$;
while $newAQuerie$ **do**
    $path \leftarrow ESPPRC(G^{ROCSP}(V, U, R, C, K))$;
    $newAQuerie \leftarrow false$;
    forall $arc \in path$ **do**
        if $arc = u_{q_i} \in U_4$ **then**
            $U_4 \leftarrow U_4 \setminus u_{q_i}$;
            $AcceptedQueries \leftarrow AcceptedQueries \cup \{q_i\}$;
            $newAQuerie \leftarrow true$;
        if $arc = (v^t_{s_i}, v^{t'}_{s_k}) \in U_3$ **then**
            $Relocation \leftarrow Relocation \cup (v^t_{s_i}, v^{t'}_{s_k})$;
        if $arc = (v^t_{s_i}, v^{t+1}_{s_i}) \in U_2$ **then**
            $k^t_{s_i} \leftarrow k^t_{s_i} - 1$;
            if $k^t_{s_i} = 0$ **then**
                $U_2 \leftarrow U_2 \setminus (v^t_{s_i}, v^{t'}_{s_k})$;
    if $newAQuerie = true$ **then**
        $Paths^{Cars} \leftarrow Paths^{Cars} \cup path$;

Plus court chemin avec contraintes de ressource

Après avoir présenté l'heuristique constructive, qui utilise un algorithme de résolution du problème du plus court chemin avec contraintes de ressource dans un graphe, nous allons détailler l'algorithme développé qui est une adaptation de l'algorithme *Elementary Shortest Path Problem with Resource Constraints (ESPPRC)* présenté dans l'article (D. Feillet et Gueguen, 2004). En effet, cet algorithme est adapté au calcul des trajets de voitures dans le $G^{ROCSP}(V, U, R, C, K)$, en considérant une seule ressource à savoir l'autonomie électrique maximale des voitures (R_{w_j}). On associe à chaque chemin $Path_j$ de la voiture w_j du sommet source V^d vers un sommet intermédiaire $v^t_{s_i}$, une étiquette $Label^j_{v^t_{s_i}} = (cost^j_{v^t_{s_i}}, charge^j_{v^t_{s_i}})$ où $cost^j_{v^t_{s_i}}$ indique le coût du chemin partiel et $charge^j_{v^t_{s_i}}$, le taux de charge de la batterie du chemin partiel au sommet $v^t_{s_i}$. Les labels d'un chemin à chaque sommet sont le résultat de l'extension des labels à partir des sommets prédécesseurs dans le graphe. Dans ce qui suit, nous allons expliquer comment les chemins sont étendus d'un sommet vers un autre et de quelle façon une règle de dominance est utilisée pour éliminer les chemins partiaux dominés. Pour finir, nous allons voir que certains plus courts chemins ne sont pas calculés à partir du sommet origine mais à partir d'un sommet intermédiaire, grâce à l'exploitation de certains labels générés par le calcul du chemin précédent dans l'algorithme 2.

Algorithm 3: ESPPRC

Data: $G^{ROCSP}(V, U, R, C, K)$, w_j ;
Result: $LabelsList_{v^t_{s_i}} \forall v \in V$

initialisation ;
$LabelsList_{V^d} = \{(0, R'_{w_j})\}$;
forall $v^t_{s_i} \in V \setminus \{V^d\}$ **do**
    $LabelsList_{v^t_{s_i}} \leftarrow \emptyset$;
$OrderedList \leftarrow Order(V)$;
forall $v^t_{s_i} \leftarrow OrderedList.next()$ **do**
    forall $v^{t'}_{s_k} \in Succ(v^t_{s_i})$ **do**
        forall $Label_{v^t_{s_i}} \in LabelsList^j_{v^t_{s_i}}$ **do**
            $LabelsList^j_{v^{t'}_{s_k}} \leftarrow LabelsList^j_{v^{t'}_{s_k}} \cup Extend(Label^j_{v^t_{s_i}}, v^{t'}_{s_k})$;
        $LabelsList^j_{v^{t'}_{s_k}} \leftarrow Clear(LabelsList^j_{v^{t'}_{s_k}})$;

L'algorithme 3 illustre l'adaptation du *ESPPRC* qui détermine le chemin le plus court à partir d'un sommet source V^d vers un sommet destination V^f et qui fait appel aux fonctions suivantes :

— $Extend(v^t_{s_i}, v^{t'}_{s_k})$: permet l'extension d'un chemin partiel dont l'extrémité terminale est le sommet $v^t_{s_i}$ vers le sommet $v^{t'}_{s_k}$; elle retourne l'étiquette $Label^j_{v^{t'}_{s_k}}$ résultante de l'extension de $Label^j_{v^t_{s_i}}$ lorsque l'extension est possible, rien sinon. La fonction est utilisée pour calculer à chaque nouveau sommet visité, le taux de charge électrique et le coût (défini comme la somme des coûts des arcs composant le chemin partiel). Lorsque le chemin partiel, dont l'extrémité terminale est $v^t_{s_i}$, est étendu vers le sommet $v^{t'}_{s_k}$, le coût est augmenté du coût de l'arc $(v^t_{s_i}, v^{t'}_{s_k})$ et le taux de charge est augmenté ou diminué selon l'un des deux cas suivants :

- Cas 1 : si $(v^t_{s_i}, v^{t'}_{s_k}) \in U_1 \cup U_3 \cup U_4$ alors $charge^j_{v^{t'}_{s_k}} = charge^j_{v^t_{s_i}} - r_{(v^t_{s_i}, v^{t'}_{s_k})}$, on soustrait à la charge la consommation $r_{(v^t_{s_i}, v^{t'}_{s_k})}$;
- Cas 2 : si $(v^t_{s_i}, v^{t'}_{s_k}) \in U_2$ donc $i = k$ et $t' = t + 1$, ce qui veut dire que la voiture reste dans la station pour se recharger : alors, on a $charge^j_{v^{t+1}_{s_i}} = charge^j_{v^t_{s_i}} + r_{(v^t_{s_i}, v^{t+1}_{s_i})}$; la charge est augmentée de $r_{(v^t_{s_i}, v^{t+1}_{s_i})}$ mais seulement dans le cas où la nouvelle charge $(charge^j_{v^t_{s_i}} + r_{(v^t_{s_i}, v^{t+1}_{s_i})})$ est inférieure ou égale à la capacité de la batterie R_{w_j} ; sinon elle reste égale à R_{w_j} même si la voiture est garée dans une station, mais elle ne peut plus se recharger au dessus de l'autonomie maximale R_{w_j}.

Une extension vers le sommet $v^{t'}_{s_k}$ est donc considérée uniquement dans le cas où, une fois le sommet $v^{t'}_{s_k}$ ajouté, le taux de charge est supérieur au seuil de charge minimale fixé à zéro dans nos travaux. À l'état initial, le label associé à la source V^d est caractérisé par le taux de charge initial $charge^j_{V^d} = R'_{w_j}$ et un coût nul $cost^j_{V^d} = 0$.

— $Order()$: permet de définir un ordre de parcours des sommets du graphe $G^{ROCSP}(V, U, R, C, K)$, de telle sorte que si un sommet $v^{t'}_{s_k}$ dans le graphe est plus grand qu'un sommet $v^t_{s_i}$ dans l'ordre défini, alors il n'existe pas de chemin dans le graphe qui mène de $v^{t'}_{s_k}$ vers $v^t_{s_i}$. Dans le cas d'un graphe acyclique, cela permet à l'algorithme *ESPPRC* de parcourir les sommets du graphe dans l'ordre et de ne pas revisiter un sommet, comme dans le cas général. Comme le graphe spatio-temporel est acyclique, nous définissons alors l'ordre des sommets, tel que $v^{t'}_{s_k} > v^t_{s_i}$ si et seulement si $v^{t'}_{s_k}$

représente une station à une période de temps supérieure à celle de $v^t_{s_i}$, donc $t' > t$. Dans le cas où les deux sommets ont la même période de temps $t' = t$, alors arbitrairement $v^{t'}_{s_k} > v^t_{s_i}$ si et seulement si $k > i$.

— $Clear()$: permet de conserver uniquement les labels non dominés dans la liste des labels à chaque sommet. En effet, pour établir qu'un chemin partiel $path_j$ domine un autre chemin $path_{j'}$, il faut d'abord que les deux chemins aient la même extrémité terminale $v^t_{s_i}$; ensuite il faut que le taux de charge du premier chemin soit supérieur ou égal au taux du deuxième chemin $charge^j_{v^t_{s_i}} \geq charge^{j'}_{v^t_{s_i}}$ et que le coût du premier soit inférieur ou égal au coût du deuxième $cost^j_{v^t_{s_i}} \leq cost^{j'}_{v^t_{s_i}}$. Les règles de dominance sont importantes afin d'éliminer un grand nombre de chemins partiels. Pour obtenir la solution optimale du problème du plus court chemin, il suffit de considérer les labels non dominés.

Nous définissons aussi quelques notations utilisées dans l'algorithme comme suit :

— $LabelsList_{v^t_{s_i}}$: liste des labels du sommet $v^t_{s_i}$;

— $Succ(v^t_{s_i})$: ensemble de successeurs du sommet $v^t_{s_i}$;

— *OrderedList* : liste de sommets ordonnés dans un ordre croissant selon l'ordre défini.

Exploitation d'anciens labels sauvegardés

L'*ESPPRC* est utilisé plusieurs fois dans l'heuristique pour le calcul du trajet de chaque voiture, d'où l'importance de réduire sa complexité. Comme dans l'heuristique, on recalcule un nouveau chemin pour la voiture après avoir supprimé les arcs correspondants aux clients satisfaits par les voitures précédentes et les stations avec un taux de remplissage égal à la capacité maximale. Au lieu de calculer un plus court chemin à chaque fois depuis la source et donc réinitialiser tous les labels pour chaque nouveau chemin, on garde les labels qui restent exploitables pour le nouveau chemin et on calcule le plus court chemin à partir d'un sommet intermédiaire. Ce dernier est choisi de telle sorte qu'on revisite tous les sommets touchés par les suppressions des arcs du chemin précédent. Plus formellement, le calcul du chemin $Path_j$ de la voiture w_j se fait de la manière suivante :

1. l'ensemble U^{j-1}_{sup} est défini comme l'ensemble des arcs supprimés dans l'itération précédente de l'heuristique lors du calcul du chemin de la voiture w_{j-1} ;

2. le sommet $v_{s_i}^{t*}$ choisi est le plus petit sommet d'extrémité initiale des arcs contenus dans U_{sup}^{j-1} ;
3. l'ensemble $U_{supAffect}^{j-1} = \{(v_{s_l}^t, v_{s_k}^{t'}) : t' > t^*\}$ est défini comme l'ensemble des arcs affectés par la suppression ;
4. la source à partir de laquelle on va reprendre le calcul du plus court chemin est le plus petit sommet $v_{s_l}^t$ d'extrémité initiale des arcs dans $U_{supAffect}^{j-1}$.

Nb : un sommet du graphe est dit plus grand qu'un autre sur la base de l'ordre de parcours définit dans la fonction $Order()$.

3.2.2 Algorithme Génétique : AG^{ROCSP}

Une adaptation de l'Algorithme Génétique basé sur l'heuristique constructive est proposée pour le $ROCSP$ (AG^{ROCSP}). Le principe de l'algorithme génétique proposé est de modifier l'ensemble des coûts C_4 des arcs représentant des clients pour la construction de chaque solution. Rappelons que l'ensemble C_4 représente un paramètre d'entrée de l'heuristique constructive qui se base sur le graphe $G^{ROCSP}(V, U, R, C, K)$ pour construire une solution. Le changement de ces coûts permet de modifier les parcours des voitures calculés par l'algorithme de plus court chemin utilisé dans l'heuristique constructive pour router les voitures. Ces changements de parcours permettent à l'heuristique de trouver des solutions différentes selon les paramètres d'entrée. Le fonctionnement général est introduit dans la figure 3.1 et ses différentes étapes sont détaillées dans le reste de cette sous-section.

Modélisation du chromosome (solution). Toute instance du problème traité est associée à un ensemble fini de solutions réalisables, dont chacune peut être caractérisée par une modélisation sous forme de chromosome, ce qui permet la distinction entre les différentes solutions. Dans l'algorithme génétique proposé, chaque solution Sol est construite en utilisant l'algorithme de l'heuristique constructive avec comme entrée le graphe spatio-temporel $G^{ROCSP}(V, U, R, C^{Sol}, K)$, avec $C^{Sol} = C_1 \cup C_2 \cup C_3 \cup C_4^{Sol}$. L'ensemble C_4^{Sol} est généré en multipliant aléatoirement chaque valeur dans C_4 par un nombre compris entre 0.5 et 2, ce qui veut dire que soit on divise au maximum par 2 le coût (gain) de la satisfaction d'une requête, soit on le multiplie au maximum par 2. Chaque ensemble C_4^{Sol} généré permet de donner un ordre d'importance entre les requêtes des clients spécifique à la solution Sol, ce qui donne des solutions différentes selon les ordres d'importance générés. Une modélisation sous forme de chromosomes de la solution Sol est une représentation vectorielle de l'ensemble C_4^{Sol}.

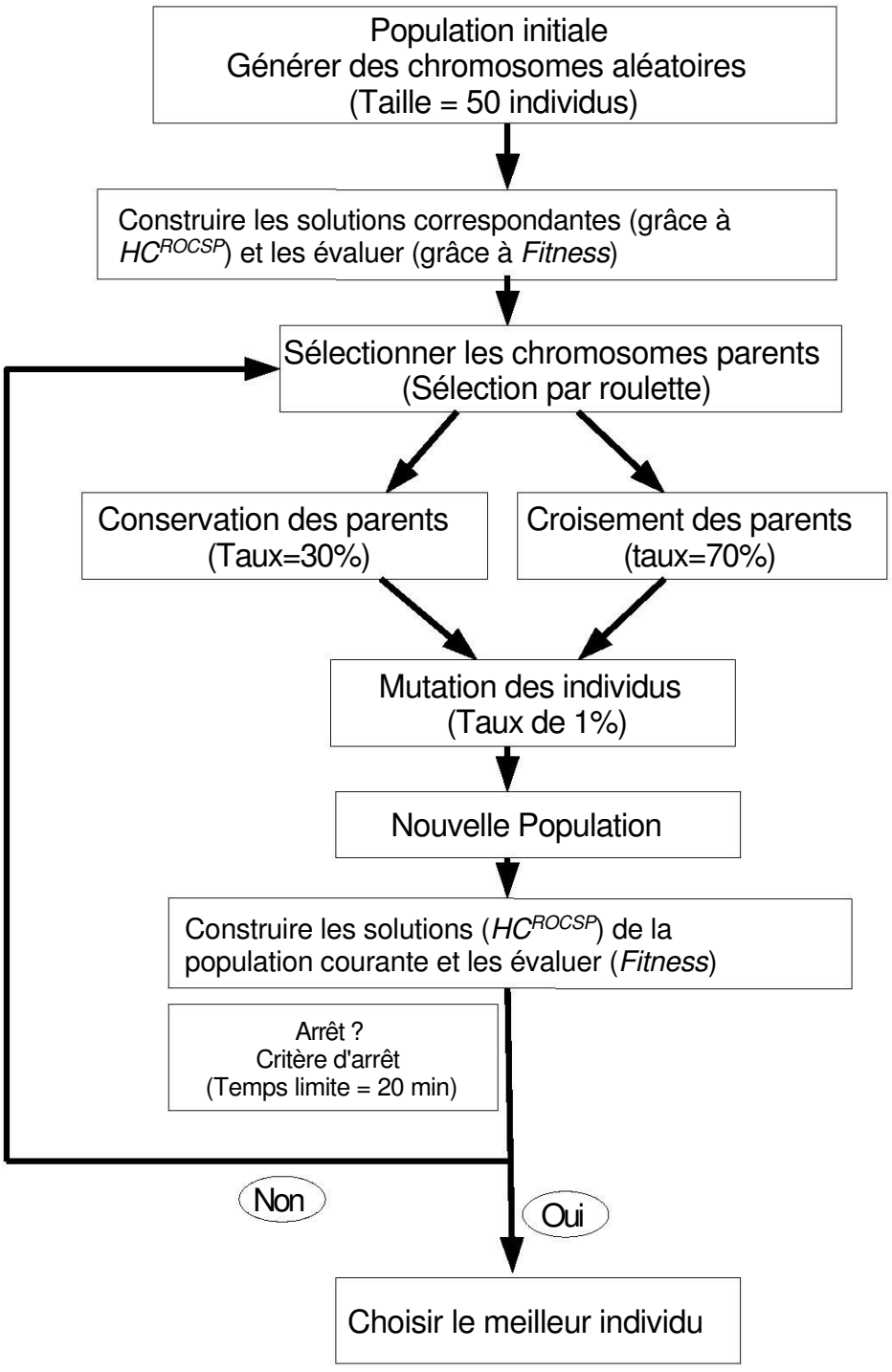


FIGURE 3.1 – *Utilisation de l'algorithme génétique pour améliorer les solutions*

Choix techniques et paramètres de l'algorithme génétique. Pour calculer la valeur *Fitness* d'une solution, la fonction objectif Ct définie précédemment est

utilisée. La sélection des solutions parentes pour la reproduction se fait de manière stochastique, par roulette sous contrainte de probabilités. Dans ce schéma de sélection, la probabilité de sélectionner un individu est proportionnelle à la valeur de sa fonction *Fitness*, ce qui veut dire que meilleure est la solution, plus elle a des chances d'être sélectionnée.

L'opérateur de croisement utilisé est un processus d'échange de parties de chromosomes choisis aléatoirement entre les deux parents. Un opérateur de mutation simple est également utilisé pour modifier aléatoirement la valeur d'une section du chromosome. Les taux de croisement et de mutation ont été fixés respectivement à 70 % et 1 % par rapport à la taille de la population, fixée elle à 50 individus.

Illustration par un exemple

Nous prenons un exemple simplifié de *ROCSP* dans le but d'illustrer le principe d'optimisation de l'heuristique constructive et l'algorithme génétique. Dans cet exemple simplifié, on dispose de 3 stations. La journée de service est réduite à un nombre de périodes de temps égal à 5. Le nombre de clients est égal à 4 et le nombre de voitures est fixé à 2. Le graphe simplifié représentant cet exemple est donné dans la figure 3.2.

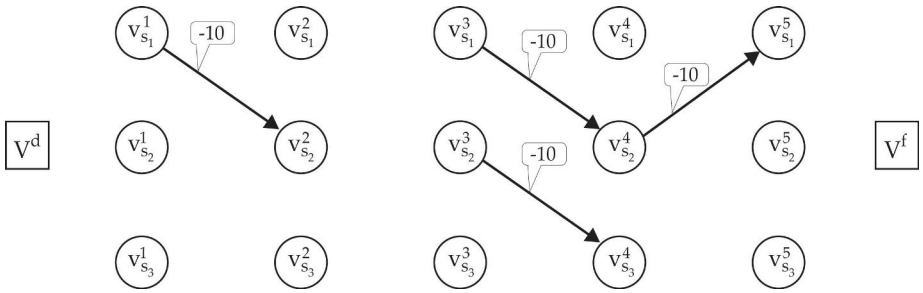


FIGURE 3.2 – *Graphe simplifié modélisant un exemple d'auto-partage avec 4 clients*

Dans l'algorithme de l'heuristique constructive, le calcul du plus court chemin pour chacune des deux voitures est fait de manière itérative (l'un après l'autre), ce qui donne le résultat suivant (cf. figure 3.3) :

— Chemin voiture 1 : $V^d \rightarrow v^1_{s_1} \rightarrow v^2_{s_2} \rightarrow v^3_{s_1} \rightarrow v^4_{s_2} \rightarrow v^5_{s_1} \rightarrow V^f$ (coût=-29) ;

— Chemin voiture 2 : $V^d \rightarrow v^1_{s_2} \rightarrow v^2_{s_2} \rightarrow v^3_{s_2} \rightarrow v^4_{s_3} \rightarrow v^5_3 \rightarrow V^f$ (coût=-10).

Le coût du chemin de la voiture 1 est égal à -30 (la somme des coûts négatifs des trois clients) plus un coût d'arc positif égal à 1 qui représente une opération de relocalisation de la voiture par un agent entre la station 2 et 1 à la période de temps numéro 2.

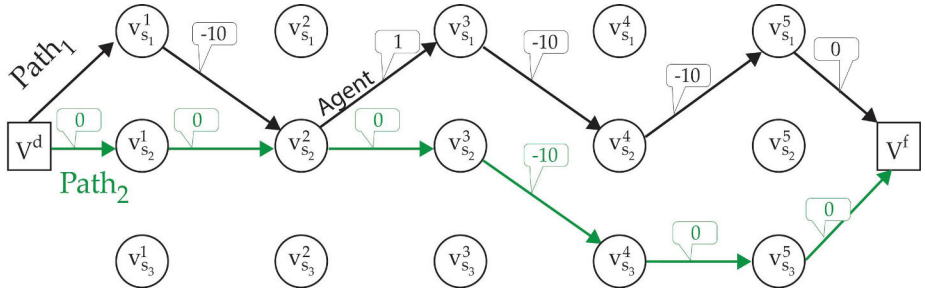


FIGURE 3.3 – *Solution d'un exemple d'auto-partage obtenue par l'heuristique constructive*

L'algorithme génétique permet de générer différents coûts négatifs des arcs représentant les clients pour la même instance du problème *ROCSP*. La figure 3.4 représente le graphe modélisant notre exemple simplifié avec une hypothèse d'un nouvel ensemble C_4^{Sol}, qui peut être généré avec l'algorithme génétique.

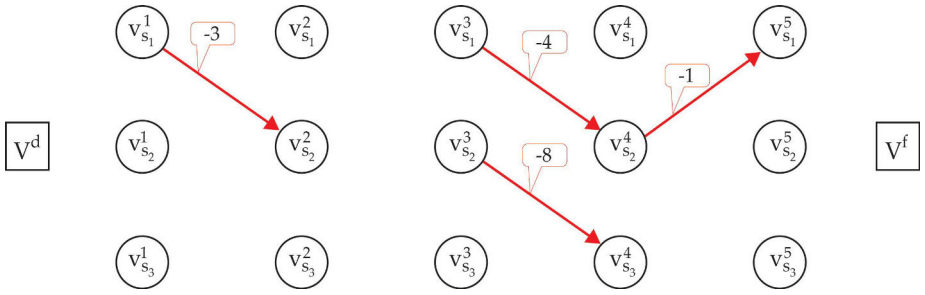


FIGURE 3.4 – *Utilisation de l'algorithme génétique pour modifier les coûts des arcs représentant les clients*

En déroulant l'heuristique constructive avec les nouveaux coûts de l'hypothèse précédente, on obtient le résultat suivant (cf. figure 3.5) :

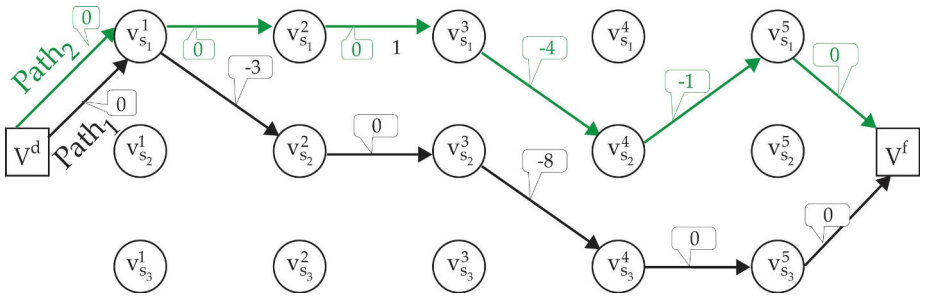


FIGURE 3.5 – *Solution obtenue après les changements de coûts des arcs représentant les clients*

— Chemin voiture 1 : $V^d \rightarrow v_{s_1}^1 \rightarrow v_{s_2}^2 \rightarrow v_{s_2}^3 \rightarrow v_{s_3}^4 \rightarrow v_{s_3}^5 \rightarrow V^f$ (coût=-11) ;

— Chemin voiture 2 : $V^d \rightarrow v^1_{s_1} \rightarrow v^2_{s_1} \rightarrow v^3_{s_1}$ -> $v^4_{s_2} \rightarrow v^5_{s_1} \rightarrow V^f$ (coût=-5).

On remarque que cette solution est meilleure, vu que le même nombre de clients a été servi, mais sans utiliser de relocalisations et donc d'agents. Cette amélioration est due au fait que l'algorithme génétique a modifié les coûts négatifs des arcs des clients et donc leurs pouvoirs d'attraction dans le processus de calcul du plus court chemin. Le résultat de cette modification est un ensemble de trajectoires différentes pour les voitures. Dans cet exemple, la solution est donc différente et meilleure que la solution initiale.

3.2.3 Évaluation des différents algorithmes

Pour évaluer l'efficacité de l'algorithme génétique et de l'heuristique constructive, nous utilisons une fonction objectif économique, dans le sens où la fonction objectif Ct est pondérée en se basant sur des estimations économiques des différents coûts et bénéfices de l'auto-partage à Nice réalisées par (Boyacı et al., 2015). En effet, dans ces travaux, le coût du véhicule est fixé à 20 euros par jour, le coût de fonctionnement du véhicule électrique à 0,01 euro par kilomètre et le personnel de relocalisation coûte 18 euros par heure. Le revenu par unité de temps de la location de voiture de partage est de 8 euros par heure. En considérant les coûts économiques et le fait que les distances de relocalisation (C_R) et les durées des requêtes clients (C_C) sont exprimées en nombre de périodes de temps de 15 min, nous obtenons la fonction objectif suivante :

$$Ct = 20 \cdot C_V + 4,57 \cdot C_R - 2 \cdot C_C.$$

Pour les petites instances (pas plus de 15 stations), nous sommes en mesure d'obtenir une solution optimale ou quasi optimale (gap entre 0% et 4%) en résolvant le modèle $R\text{-}PLNE^{ROCSP}$ via le solveur *CPLEX*. Pour ces problèmes, nous montrons que les différences de performance entre la solution optimale du PLNE et la solution de l'algorithme génétique dans la Table 3.1 sont minimes. En effet, les résultats de la Table 3.1 montrent que l'algorithme génétique permet de trouver des solutions avec un écart maximum d'environ 7 % par rapport à la solution optimale et permet dans tous les cas d'améliorer la solution trouvée par l'heuristique constructive. Cependant, pour les gros problèmes, lorsqu'il y a plus de 15 stations et 150 requêtes de clients, le modèle PLNE nécessite une quantité de mémoire et de temps CPU trop importante. De ce fait, l'exécution s'arrête avec des gap d'environ 10% pour les instances entre 18 et 21 stations et ne donne aucun résultat pour les instances de plus de 21 stations pour cause de "Out of Memory".

Donc, pour les instances de Trévise (18 stations), La Spezia (18 stations) et Buenos Aires (21 stations), nos heuristiques trouvent de meilleures solutions

TABLE 3.1 – *Résultats des simulations (ObjValue : valeur de la fonction objectif, nbS : nombre de stations, nbQ : nombre de requêtes clients, nbQA : nombre de requêtes satisfaites, nbV : nombre de voitures déployées, nbO : nombre d'opérations de relocalisation effectué, Gap : gap relatif en la valeur de la fonction objectif de la solution trouvée et celle donnée par le PLNE*

Instances			Comparaison des Algorithmes						
Ville	nbS	nbQ	Algorithme	*ObjValue*	*nbQA*	*nbV*	*nbO*	*Gap*	*Time*
Bari	13	130	$R\text{-}PLNE^{ROCSP}$	422 Euro	100	17	14	0%	33.3 heures
			HC^{ROCSP}	408 Euro	97	16	16	3.3%	1.1 sec
			AG^{ROCSP}	417 Euro	100	17	15	1.1%	2.2 min
Reggio d'Émilie	14	140	$R\text{-}PLNE^{ROCSP}$	419 Euro	100	17	20	0%	26 jours
			HC^{ROCSP}	392 Euro	102	18	22	6.4%	1.4 sec
			AG^{ROCSP}	396 Euro	104	18	21	5.4%	3.1 min
Bergame	15	150	$R\text{-}PLNE^{ROCSP}$	569 Euro	-	-	-	3.69%	34.3 heures
			HC^{ROCSP}	533 Euro	112	17	24	6.3%	1.9 sec
			AG^{ROCSP}	553 Euro	113	18	21	2.8%	4.08 min
Parme	15	150	$R\text{-}PLNE^{ROCSP}$	581 Euro	-	-	-	0.17%	36 jours
			HC^{ROCSP}	539 Euro	120	18	24	7.2%	2.2 sec
			AG^{ROCSP}	543 Euro	121	18	24	6.5%	1.2 min
Trévise	18	180	$R\text{-}PLNE^{ROCSP}$	684 Euro	-	-	-	10.95 %	15 jours
			HC^{ROCSP}	694 Euro	133	20	28	-1.4 %	3.8 sec
			AG^{ROCSP}	707 Euro	140	21	30	-3.4%	8.6 min
La Spezia	18	180	$R\text{-}PLNE^{ROCSP}$	676 Euro	-	-	-	11.35 %	18 jours
			HC^{ROCSP}	770 Euro	161	24	32	-13.9%	4.1 sec
			AG^{ROCSP}	778 Euro	162	24	32	-15%	4.7 min
Buenos Aires	21	210	$R\text{-}PLNE^{ROCSP}$	559 Euro	-	-	-	12.88 %	4.2 jours
			HC^{ROCSP}	584 Euro	142	23	46	-4.4%	4.1 sec
			AG^{ROCSP}	598 Euro	143	23	45	-6.9%	7.25 min

(3.4%, 15 % et 6.9%) en beaucoup moins de temps. Cela conforte le choix d'utilisation de ces algorithmes heuristiques capables de trouver un bon compromis entre la qualité de la solution et le temps de calcul. Avec un temps de traitement limité à 20 minutes, l'algorithme est évolutif et plus adéquat pour l'étude des grands systèmes d'auto-partage, notamment en condition réelle, comme à Nice, sujet de la dernière section de ce chapitre.

3.3 Résolution heuristique du *ESRP*

Rappelons d'abord que la résolution du *ROCSP* n'était que la première partie de notre problème intégral, car même si nous trouvons une solution du *ROCSP* avec l'ensemble des opérations de relocalisation à effectuer, nous devons encore calculer une solution pour le problème de planning des agents, afin d'assigner de façon optimale les voitures au personnel chargé de la relocalisation. Nous rappelons aussi les notations suivantes définies précédemment dans 2.5 et propres au *ESRP* :

— O : ensemble des opérations de relocalisation, à chaque $o_i \in O$ sont associées : $S^d_{o_i}$, $T^d_{o_i}$, $S^f_{o_i}$ et $T^f_{o_i}$ correspondant à la station de départ de l'opération o_i, l'heure de début de l'opération o_i, la station d'arrivée de l'opération o_i et l'heure d'arrivée de l'opération o_i, respectivement.

— $G^{ESRP}(V', U', R')$: graphe modélisant l'*ESRP* présenté dans 2.5.1.

Nous définissons une heuristique constructive pour obtenir une solution initiale faisable pour l'*ESRP*. Par la suite, dans le but d'améliorer la qualité des solutions, un algorithme de *Branch and Price* est proposé et adapté pour une résolution heuristique et non exacte.

3.3.1 Heuristique Constructive : HC^{ESRP}

Le principe du HC^{ESRP} (Heuristique Constructive pour l'*ESRP*) est semblable à celui du HC^{ROCSP}, sauf qu'il s'agit de router les agents l'un après l'autre au lieu des voitures à travers le graphe $G^{ESRP}(V', U', R')$, en calculant à chaque itération de l'heuristique le chemin le plus court avec des contraintes de ressource de $V^{d'}$ à $V^{f'}$.

Pour pouvoir calculer un tel plus court chemin, des coût négatifs sont affectés aux arcs du graphe, ce qui donne le graphe $G^{ESRP}(V', U', R', C')$ avec

$$C' = \{c_{(v_{o_i},v_{o_k})} = (T^d_{o_k} - T^f_{o_i}) - B : (v_{o_i}, v_{o_k}) \in U', B > \sum_{(v_{o_i},v_{o_k})\in U'} (T^d_{o_k} - T^f_{o_i})\}.$$

Le coût de chaque arc est négatif car la constante B utilisée est strictement supérieure à la somme des temps qui séparent les opérations de relocalisation. Cela signifie aussi que le plus court chemin est le chemin qui maximise le nombre d'arcs, et donc celui qui prend en charge le plus grand nombre d'opérations de relocalisation, tout en minimisant en second lieu la somme des distances parcourues en dehors des opérations de relocalisation, plus les temps d'attentes de l'agent.

Le fait que le graphe $G^{ESRP}(V', U', R', C')$ soit acyclique, nous permet d'exploiter à nouveau le *ESPPRC* (algorithme 3) utilisé pour HC^{ROCSP} pour calculer le plus court chemin de $V^{d'}$ à $V^{f'}$; cela en définissant comme ressource le temps de travail maximal pour chaque agent $R_{a_j} : a_j \in A$ ainsi qu'un ordre de parcours sur les sommets tel que v_{o_i} est supérieur à v_{o_k} si :

— $T^d_{o_i} >= T^f_{o_k}$, c'est à dire que l'opération o_i commence après la fin de o_k et donc il n'existe pas de chemin entre v_{o_i} et v_{o_k}) ;
Ou

— $T^d_{o_i} >= T^d_{o_k}$ et les intervalles de temps des deux opérations se chevauchent de telle sorte que $]T^d_{o_i}, T^f_{o_i}[\cap]T^d_{o_k}, T^f_{o_k}[\neq \emptyset$, ce qui veut dire qu'il n'existe pas de chemin entre les deux sommets car il est impossible de les affecter au même agent ; v_{o_i} est choisi alors arbitrairement comme plus grand si $T^d_{o_i} >= T^d_{o_k}$.

À la fin de chaque itération, les sommets correspondant aux opérations de relocalisation associés au dernier chemin calculé sont supprimés de l'ensemble V'. On arrête l'heuristique lorsque toutes les opérations d'équilibrage sont affectées aux agents, ce qui signifie que l'ensemble V' est égal à $\{V^{d'}, V^{f'}\}$. Le pseudo code de l'heuristique est décrit à l'aide de l'algorithme 5.

Pour améliorer la qualité de la solution obtenue par l'heuristique, on exécute l'algorithme HC^{ESRP} plusieurs fois, en modifiant à chaque fois le graphe $G^{ESRP}(V', A', R', C')$ comme suit :

1. Fixer un seuil R^{seuil} sur la consommation des ressources d'arcs, qui nous permet de diviser chaque chemin contenant un arc (v_{o_i}, v_{o_k}) avec $r_{(v_{o_i},v_{o_k})} \geq R^{seuil}$. Le seuil R^{seuil} a été fixé dans nos simulations à la moyenne des consommations de ressource de tous les arcs ;
2. Diviser chaque $path_j \in Paths_{Agents}$ en deux chemins : $Path'_j$ et $Path''_j$, en supprimant (v_{o_i}, v_{o_k}) avec la plus grande consommation de ressource $r_{(v_{o_i},v_{o_k})} \geq R^{seuil}$;
3. $Paths_{Agents} \leftarrow Paths_{Agents} \setminus path_j \cup \{Path'_j, Path''_j\})$;

Algorithm 4: Pseudo code du HC^{ESRP}

Data: $G^{ESRP}(V', U', R', C')$
Result: $Paths_{Agents}$

```
Initialization ;
Paths_Agents ← ∅ ;
j ← 0 ;
while (V' ≠ {V^d', V^f'}) do
    path_j ←ESPPRC(G(V', U', R', C')) ;
    forall (v_oi, v_ok) ∈ path_j do
        if (v_oi ≠ V^d') then
            V' ← V' \ v_oi ;
        if (v_ok ≠ V^f') then
            V' ← V' \ v_ok ;
    Paths_Agents ← path_j ;
    j ← j + 1 ;
```

4. Répéter les deux dernières étapes jusqu'à ce que $Paths_{Agents}$ ne contienne plus de chemin avec un arc dont la consommation de ressource dépasse R^{seuil} ;
5. Contracter chaque $path_j$ dans $Paths_{Agents}$ en un super-sommet qui possède le même ensemble d'arcs entrants que le premier sommet de $path_j$ et qui a également le même ensemble d'arcs sortants que le dernier sommet $path_j$ et l'ajouter à V', en supprimant tous les sommets contenus dans $path_j$ ainsi que les arcs relatifs à ces sommets de U'. Avec ces transformations, nous obtenons un nouveau graphe sur lequel on va exécuter de nouveau l'HC^{ESRP} jusqu'à atteindre le critère d'arrêt, fixé dans nos simulations à 10 itérations successives sans amélioration de la solution.

3.3.2 Branch and price : BP^{ESRP}

Un algorithme de *Branch and Price* BP^{ESRP} est proposé pour résoudre l'*ESRP*, algorithme dont le principe général de la méthode est expliqué en détail dans 1.3.1. Une génération de colonnes est appliquée à chaque nœud de l'arbre de recherche de solutions du *Branch and Price*. Comme la génération de colonnes ne fournit pas forcément des solutions entières, à chaque nœud sont ajoutées des contraintes (règles de branchement) jusqu'à forcer l'entièreté des solutions au niveau des feuilles de l'arbre. Toutefois, nous ne visons pas à explorer tout

l'arbre de recherche, mais nous effectuons une recherche en profondeur d'abord et nous nous arrêtons à la première solution entière trouvée. L'heuristique de génération de colonnes divise le problème en deux sous-problèmes : le problème *maître* et le problème du *Pricing*. Nous proposons une décomposition via une génération de colonnes, de telle sorte que le problème maître se charge de vérifier que toutes les opérations de relocalisation sont bien satisfaites. Dans le problème du *Pricing*, on cherche le meilleur chemin d'agents (colonne) qui peut améliorer la solution.

Problème maître

Nous utilisons la solution fournie par l'HC^{ESRP} comme solution initiale du problème maître, en initialisant l'ensemble P des chemins avec les chemins représentant cette solution. Nous définissons le vecteur de constantes $\gamma^j_{(v_{o_i},v_{o_k})}$ de telle sorte que :

— $\gamma^j_{(v_{o_i},v_{o_k})} \in \{0,1\}$: égale à 1, si le chemin numéro $j \in P$ utilise l'arc $(v_{o_i}, v_{o_k}) \in U'$, 0 sinon.

Le modèle PLNE du problème maître se base sur un vecteur de variables de décisions x_j et il est formulé par PM^{ESRP}.

— $x_j \in \{0,1\}$: égale à 1 si le chemin numéro $j \in P$ est utilisé dans la solution, 0 sinon.

$$\min \quad Ct' = \delta \cdot (\sum_{j \in P} x_j) + \omega \cdot (\sum_{j \in P} \sum_{(v_{o_i},v_{o_k}) \in U'} \gamma^j_{(v_{o_i},v_{o_k})} \cdot r_{(v_{o_i},v_{o_k})} \cdot x_j) \quad (PM^{ESRP})$$

sous contraintes de (3.1) et (3.2).

$$\sum_{j \in P} \sum_{v_{o_k} \in V'} \gamma^j_{(v_{o_k},v_{o_i})} \cdot x_j = 1 \qquad \forall o_i \in V' \setminus \{V^{d'}, V^{f'}\} \tag{3.1}$$

$$\sum_{j \in P} x_j \leq A^{max} \tag{3.2}$$

Il y a deux contraintes dans le problème maître : l'équation (3.1) permet de satisfaire toutes les opérations d'équilibrage et l'équation (3.2) vérifie que le nombre de chemins utilisés ne dépasse pas le nombre d'agents.

Problème du *Pricing*

Le problème du *Pricing* est un problème de plus court chemin avec des contraintes de ressource entre $V^{d'}$ et $V^{f'}$ dans le graphe $G^{ESRP}(V', U', R')$. La ressource est définie comme le temps de travail maximal des agents R_{a_j} et les coûts des arcs dans le graphe sont obtenus grâce au calcul du coût réduit d'un chemin $C\bar{R}_j$.

$$C\bar{R}_j = \delta + \omega \cdot \Big(\sum_{(v_{o_i}, v_{o_k}) \in U'} \gamma^j_{(v_{o_i}, v_{o_k})} \cdot r_{(v_{o_i}, v_{o_k})} \Big) - \Big(\sum_{(v_{o_k}, v_{o_i}) \in U'} \gamma^j_{(v_{o_k}, v_{o_i})} \cdot \pi_{o_i} \Big) - \kappa$$

Avec :

— π_{o_i} : variables duales correspondantes aux contraintes (3.1) ;

— κ : variable duale correspondante à la contrainte (3.2).

Le coût réduit par arc est obtenu en simplifiant $C\bar{R}_j$ comme suit :

$$C\bar{R}_j = \delta - \kappa + \sum_{(v_{o_k}, v_{o_i}) \in U'} \gamma^j_{(v_{o_k}, v_{o_i})} \cdot \Big(\omega \cdot r_{(v_{o_k}, v_{o_i})} - \pi_{o_i} \Big)$$

Alors, le coût de l'arc $(v_{o_k}, v_{o_i}) \in U'$ est égal à $(\omega \cdot r_{(v_{o_k}, v_{o_i})} - \pi_{o_i})$. Le problème du *Pricing* est résolu en utilisant le *ESPPRC* (algorithme 3), si le coût $C\bar{R}_j$ du chemin calculé est inférieur à zéro, alors il est ajouté à l'ensemble P.

Règle de branchement

Dans chaque nœud de l'arbre de recherche, un branchement sur un arc donné $(v_{o_i}, v_{o_k}) \in U'$ est effectué avec la contrainte (3.3), dans le but de passer d'un nœud à un autre du *Branch and Price* et forcer à terme la solution de la GC a être entière.

$$\sum_{j \in P} \gamma^j_{(v_{o_i}, v_{o_k})} \cdot x_j = 1 \tag{3.3}$$

Après un tel branchement, les arcs (v_{o_e}, v_{o_k}) et (v_{o_i}, v_{o_l}) dans U' et qui ont une seule extrémité en commun avec l'arc du branchement (v_{o_i}, v_{o_k}) sont supprimés du graphe $G^{ESRP}(V', U', R')$. Cette manipulation est faite dans le but d'éviter le calcul des chemins qui ne respectent pas la contrainte de branchement et donc générer des itérations inutiles dans la GC.

Afin d'obtenir une solution dans un délai raisonnable, nous ne parcourons pas tout l'arbre de recherche, mais nous effectuons une recherche en profondeur d'abord et nous nous arrêtons à la première solution entière trouvée. Le choix de la branche de l'arbre à explorer est fait de manière heuristique, de telle sorte qu'à chaque nœud du BP^{ESRP}, le branchement est effectué sur l'arc (v_{o_i}, v_{o_k}) avec la plus petite consommation de ressources $r_{(v_{o_i}, v_{o_k})}$ dans le chemin ayant la plus grande valeur fractionnaire dans la solution du problème maître du nœud courant. Le *Branch and Price* est arrêté dès qu'une solution réalisable est obtenue. Son pseudo-code est donné dans l'algorithme 5.

Algorithm 5: Pseudo code du BP^{ESRP}

```
/* applique la génération de colonnes */
GC(ESRP) ;
while (∃ x_l ∈ P : 0 < x_l < 1 ) do
    /* Choisir x_j avec la valeur fractionnaire maximale */
     x_j ← max(x_l ∈ P : x_l < 1) ;
    (v*_{o_i}, v*_{o_k}) ← (v_{o_i}, v_{o_k}) ∈ path_j : r_{(v_{o_i},v_{o_k})} = min(r_{(v_{o_n},v_{o_m})} : (v_{o_n}, v_{o_m}) ∈
     path_j) ;
    PM^{ESRP} ← PM^{ESRP} + Σ_{l∈P} γ^l_{(v*_{o_i}, v*_{o_k})} · x_l = 1 ;
    forall (v_{o_i}, v_{o_k}) ∈ U' do
        if (v_{o_i} = v*_{o_i} & v_{o_k} ≠ v*_{o_k}) or (v_{o_i} ≠ v*_{o_i} & v_{o_k} = v*_{o_k}) then
            U' ← U' \ (v_{o_i}, v_{o_k}) ;
    GC(ESRP) ;
```

3.3.3 Évaluation des différents algorithmes

Afin d'évaluer les deux algorithmes proposés : le HC^{ESRP} et le BP^{ESRP} pour l'*ESRP*, ces deux algorithmes sont comparés avec le modèle $PLNE^{ESRP}$. La comparaison se base sur l'affectation de coûts discriminants dans la fonction objectif Ct'. En effet, le coût unitaire de déploiement d'un agent δ a été fixé à 100 fois ω, le coût unitaire du déplacement d'un agent sans voitures du service.

Dans la table 3.2, nous pouvons observer que le BP^{ESRP} donne de meilleurs résultats que le HC^{ESRP}, même si cela prend plus de temps. Toutefois, cela reste une durée raisonnable qui ne dépasse pas les 10 min sur la plus grande instance. Nous observons également que le BP^{ESRP} heuristique donne la solution optimale pour les petites instances. Pour les grandes instances, le modèle PLNE prend tellement de temps que nous devons arrêter l'exécution du modèle. Le

TABLE 3.2 – *Résultats de la simulation (nbS : nombre de stations, nbO : nombre d'opérations de relocalisation, nbA : nombre d'agents, gap1 : différence de coût relative (en %) entre HC^{ESRP} et BP^{ESRP}, gap2 : différence de coût relative (en %) entre BP^{ESRP} et le modèle $PLNE^{ESRP}$, gap3 : le GAP du modèle PLNE donné par CPLEX)*

Instances			Comparaison des Algorithmes pour l'*ESRP*								
			nbA			*Gap*			Temps en sec		
nbS	*nbO*	HC^{ESRP}	BP^{ESRP}	$PLNE^{ESRP}$	gap1	gap2	gap3	HC^{ESRP}	BP^{ESRP}	$PLNE^{ESRP}$
Bari (13 Stations)	16	3	3	3	0 %	0 %	0 %	0,03	0,09	0,21
Reggio d'Émilie (14 Stations)	20	3	3	3	3,72 %	0 %	0 %	0,06	0,18	0,4
Bergame (15 Stations)	23	5	5	5	0 %	0 %	0 %	0,06	0,19	2,63
Parme (15 Stations)	24	5	5	5	0,69 %	0 %	0 %	0,14	0,47	3,78
Trévise (18 Stations)	28	6	6	6	0,85 %	0 %	0 %	0,15	0,39	5,92
La (20 Stations)	33	5	5	5	0,33 %	0 %	0 %	0,17	1,14	1,36
Buenos Aires (21 Stations)	32	6	6	6	1 %	0 %	0 %	0,14	0,7	2,7
Ottawa (21 Stations)	33	6	6	6	0,72 %	0 %	0 %	0,13	0,62	8,37
San Antonio (23 Stations)	33	5	5	5	0 %	0 %	0 %	0,23	0,47	2,68
Brescia Stationscia (27 Stations)	45	6	6	6	0,41 %	0 %	0 %	0,32	3,25	7,97
Rome (28 Stations)	42	8	8	8	0,64 %	0 %	0 %	0,27	5,62	649,16
Madison Stationson(28 Stations)	43	10	9	9	9,74 %	0 %	0 %	0,16	2,17	18,42
Guadalajara (41 Stations)	71	11	11	11	0,7 %	0 %	0 %	0,5	20,7	24434,61
Dublin (45 Stations)	76	13	13	13	0,8 %	0 %	0 %	1	55,2	44462,59
Denver (51 Stations)	86	12	12	12	0 %	0,48 %	24,55 %	8	195	3684
Rio De Janeiro (55 Stations)	103	14	13	13	6,86 %	-1,79 %	16,2 %	2,19	99,56	3814,09
Boston (59 Stations)	99	12	12	12	0,89 %	0 %	0 %	1,4	55,21	24890,74
Nice (64 Stations)	131	17	17	17	0,6 %	0,96 %	18,2 %	2,6	216,6	15282.45
Turin (65 Stations)	135	19	19	18	0 %	1,11 %	21,4 %	3,77	316,6	15521,68
Toronto (75 Stations)	143	16	16	16	0,91 %	-1,91 %	15,39 %	3,21	312,14	15427,71
Miami (80 Stations)	152	16	15	16	5,89 %	-5,69 %	13,9 %	3,83	317,71	15721,18
Mexico (82 Stations)	152	18	18	18	0 %	1,69 %	6,57 %	10,06	370,4	15974,56
Minneapolis (90 Stations)	174	21	21	21	0,55 %	-2 %	15,65 %	5,66	578,4	18172,21

BP^{ESRP} donne des résultats presque identiques ou même meilleurs que le modèle PLNE, lorsque l'exécution à long terme du modèle PLNE est interrompue.

3.4 Simulations de l'auto-partage à Nice

Nous avons montré que les algorithmes proposés pour les deux problèmes *ROCSP* et *ESRP* donnent des résultats approchés de bonne qualité, par l'usage d'un algorithme génétique ou d'un branch and price, resp. AG^{ROCSP} et BP^{ESRP}, pour étudier le cas de l'auto-partage de la ville de Nice. La fonction objectif ou *fitness* de l'AG^{ROCSP} est utilisée avec d'autres coefficients alternatifs aux coefficients économiques définis précédemment, dans le but d'identifier les paramètres les plus influents sur l'efficacité du service. Pour mesurer l'influence de l'autonomie des voitures sur l'efficacité du service, plusieurs scénarios d'autonomie sont aussi définis.

3.4.1 Comparaison des résultats de scénarios avec différentes fonctions objectif

La fonction objectif utilisée dans l'algorithme AG^{ROCSP} peut potentiellement être ajustée pour optimiser un critère plutôt qu'un autre en fixant des valeurs discriminantes aux poids. Rappelons que dans la fonction objectif $Ct = \alpha \cdot C_V + \beta \cdot C_R - \gamma \cdot C_C$, C_V est le nombre de véhicules, C_R la distance parcourue pendant la relocalisation, C_C le gain dû à la satisfaction des requêtes et les coefficients α, β et γ sont respectivement leurs poids dans la fonction objectif.

Cinq scénarios différents avec des objectifs différents sont proposés : le scénario *Economical* a comme objectif une évaluation économique des coûts ; le scénario *Service-quality* favorise la qualité du service (nombre de clients satisfaits) ; le scénario *Free-reloc* néglige le coût de la relocalisation ; le scénario *Free-cars* néglige le coût de déploiement d'une voiture ; dans le scénario *No-reloc*, le coût de relocalisation est élevé de façon à ce que l'algorithme AG^{ROCSP} ne génère pas de relocalisation. Pour évaluer les avantages de la relocalisation, les quatre premiers scénarios avec relocalisation de voitures sont comparés avec le cinquième sans relocalisation. Les cinq scénarios et leurs fonctions objectif sont détaillés dans la table 3.3.

Tout d'abord, donnons un aperçu général des résultats présentés aux figures 3.6 à 3.9. Sur ces figures, les courbes résultent de la moyenne des scores de simulation de 20 instances du problème *ROCSP* générées pour la ville de Nice

TABLE 3.3 – *Différents scénarios selon la fonction objectif Ct du ROCSP*

Scénario	Fonction objectif correspondante	Explications
Economical	$Ct = 20 \cdot C_v + 4,57 \cdot C_R - 2 \cdot C_C$	Ce scénario est basé sur des estimations économiques des différents coûts et bénéfices de l'auto-partage à Nice réalisées par (Boyacı et al., 2015), comme expliqué dans 3.2.3
Service-quality	$Ct = 20 \cdot C_v + 4,57 \cdot C_R - 200 \cdot C_C$	Dans ce scénario, la priorité principale est de satisfaire le plus grand nombre de clients; puis vient la minimisation des ressources (nombre de voitures utilisées et temps de relocalisation). Selon cet ordre de préférence, la valeur du gain γ de la satisfaction d'un client est multiplié par 100
Free-reloc	$Ct = 20 \cdot C_v + 0.0457 \cdot C_R - 2 \cdot C_C$	Dans ce scénario, le coût de relocalisation est marginalisé, en divisant la valeur de β par 100
Free-cars	$Ct = 0.2 \cdot C_v + 4.57 \cdot C_R - 2 \cdot C_C$	Dans ce scénario, le coût de déploiement de voitures est marginalisé, en divisant la valeur α par 100
No-reloc	$Ct = 20 \cdot C_v + 457 \cdot C_R - 2 \cdot C_C$	Dans ce scénario, aucune relocalisation n'est effectuée; pour ce faire, le coût de relocalisation est surestimé en multipliant la valeur de β par 100

comme expliqué dans 2.6. Les calculs sont réalisés en fonction de plusieurs paramètres d'efficacité du service et d'une gamme de demandes de clients allant jusqu'à 1000 requêtes. Les figures 3.10 et 3.11, quant à elles, donnent les résultats relatifs aux agents de relocalisation calculés par le BP^{ESRP} pour les 4 premiers scénarios. Le dernier scénario *No-reloc* n'est pas concerné par ces deux figures, car il ne génère pas de relocalisation.

Un faible intervalle de confiance est attribué à chaque point des courbes. Dans la plupart des cas, les lignes des courbes sont distinctes et donc les résultats sont discriminants (seuls quelques intervalles de confiance se chevauchent), ce qui rend leur interprétation très fiable. Pour tout critère, il existe toujours au moins un scénario avec relocalisation qui est meilleur que celui sans relocalisation. De plus, la plupart du temps, le scénario *No-reloc* produit des scores plus bas ou même parfois les plus mauvais avec un très grand écart par rapport aux autres scores de scénarios avec relocalisation.

Généralement, on observe une relation linéaire entre le nombre de requêtes des clients et les gains réalisés, ainsi que la quantité de ressource utilisée. En revanche, cette relation est plutôt amortie quand il s'agit des taux de satisfaction des clients et de partage des voitures. Cela signifie que les systèmes avec relocalisations atteignent une bonne efficacité à partir d'un certain seuil de demandes, autour de 600 requêtes pour le cas de Nice. Lorsque la demande dépasse ce seuil, la part des clients satisfaits (cf. Figure 3.7) et le taux d'utilisation de véhicule (Figure 3.9) ne croissent pas en proportion. De tels systèmes semblent donc robustes et arrivent à une efficacité quasi-optimale à partir d'un taux de demande moyen (environ 10 requêtes par station dans la journée).

Analysons maintenant les résultats des différentes figures avec plus de détails :

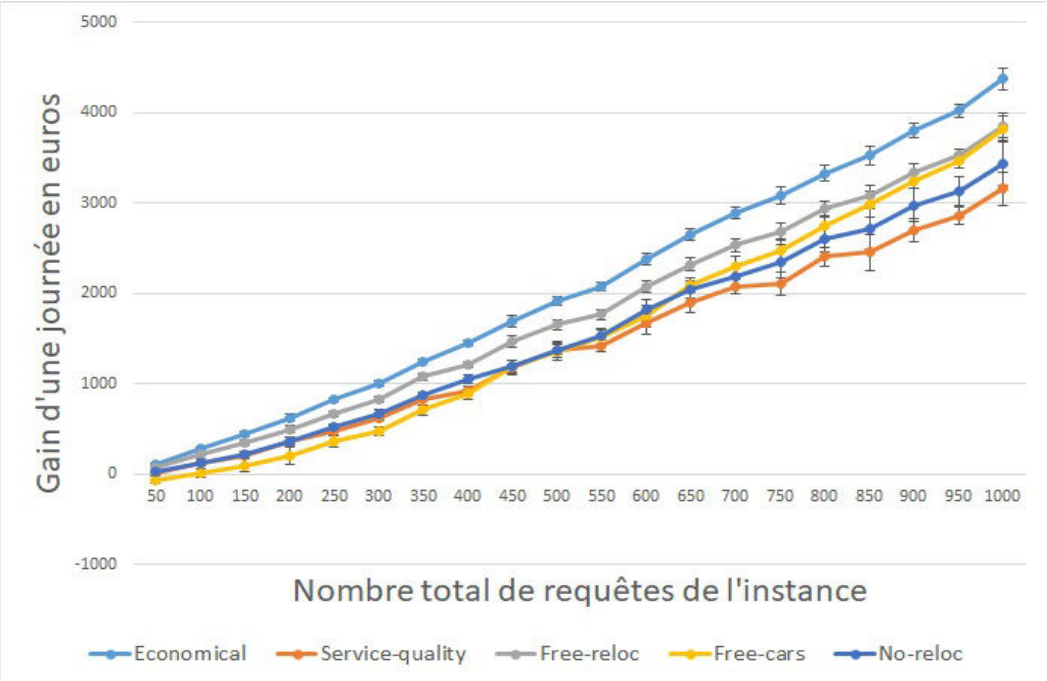


FIGURE 3.6 – *Résultats économiques avec différents scénarios de fonctions objectif multi-critères*

— la figure 3.6 fournit une évaluation du gain économique pour les solutions des cinq scénarios, même si les solutions de chaque scénario sont calculées en utilisant différentes fonctions *Fitness* dans l'AG^{ROCSP} ; mais elles sont réévaluées à la fin du calcul avec la fonction objectif économique. Nous pouvons observer que le scénario *Economical* présente le meilleur gain, ce qui est logique car il a comme objectif prioritaire de maximiser le gain économique. Le scénario *Free-reloc* est celui qui présente le second meilleur gain : c'est celui où l'on autorise beaucoup de relocalisations. Ce résultat nous permet de déduire que l'utilisation de la relocalisation pour satisfaire les clients est plutôt rentable. Les moins bons scénarios en termes économiques sont les scénarios *Service-quality* et *Free-cars* où beaucoup de voitures sont déployées (ce qui réduit le gain) et le scénario *No-reloc* où il n'y a pas de relocalisation. Toute fois, le scénario *Free-cars* se rapproche économiquement du scénario *Free-reloc* dans les journées de fortes demandes. Le rapprochement s'explique par le fait que le coût du grand nombre de voitures utilisées est rentabilisé quand la demande est forte car le temps de location de chaque voiture augmente.

— la figure 3.7 donne le pourcentage de satisfaction des clients pour chaque

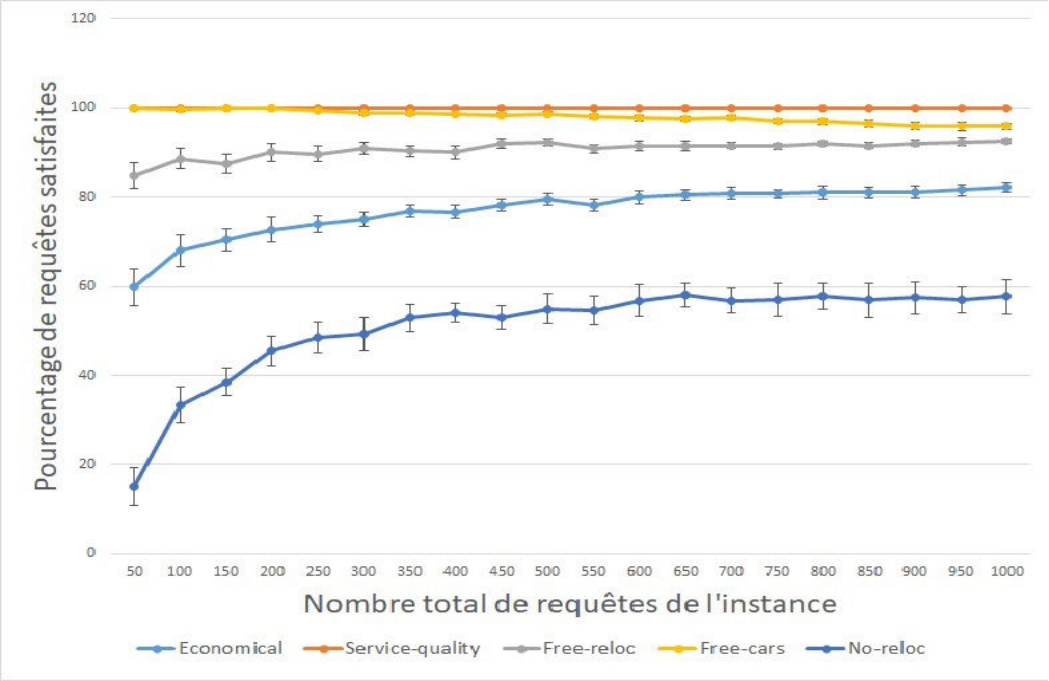


FIGURE 3.7 – *Pourcentage de satisfaction des requêtes pour différents scénarios de fonctions objectif multi-critères*

scénario. Nous pouvons observer que les scénarios *Service-quality* et *Free-cars* donnent des résultats très similaires avec un pourcentage de satisfaction des clients élevé ; celui-ci est de 100 % pour le scénario *Service-quality*, car il s'agit du scénario orienté qualité de service où tous les coûts sont négligés. Les résultats des deux scénarios permettent de conclure qu'en augmentant le nombre de voitures, on augmente significativement le taux de satisfaction des clients, ce qui est normal vu que le nombre de voitures se rapproche du nombre de clients, diminuant mécaniquement le taux de partage. En revanche, le scénario *Free-reloc* est le plus intéressant, car en négligeant seulement le coût de relocalisation, on arrive à avoir des taux de satisfaction élevés, sans diminuer le taux de partage. Ces taux commencent à 80 % pour les petites demandes et dépassent la barre des 90% sur les journées à forte demande, en se rapprochant des résultats du scénario *Free-cars* qui chute à 90%. Cette chute s'explique par la saturation des contraintes de capacité dans les stations : l'ajout de voitures sans relocalisation ne permet pas une satisfaction à 100%, car le service est saturé et la relocalisation est alors nécessaire pour optimiser les ressources.

— La figure 3.8, cohérente avec les résultats précédents, apporte un résultat intéressant. Globalement, les scénarios *Service-quality* et *Free-cars* semblent proches avec un grand nombre de voitures déployées, en raison de la négligence du coût des voitures dans la fonction objectif. Mais le résultat le

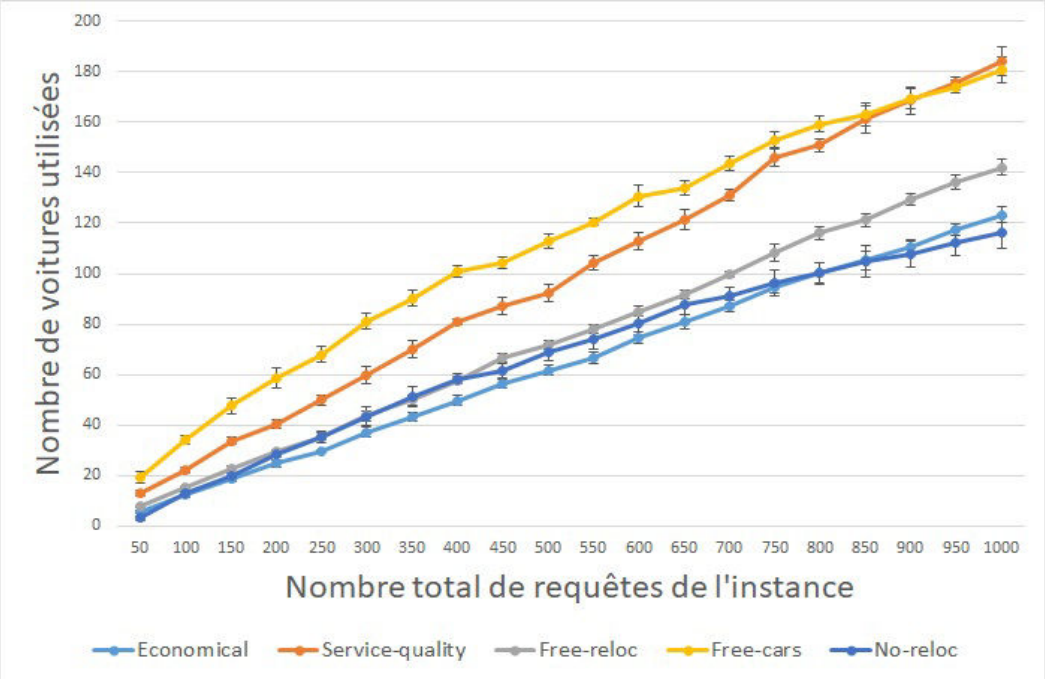


FIGURE 3.8 – *Nombres de voitures utilisées pour différents scénarios de fonctions objectif multi-critères*

plus intéressant dans cette courbe est que les scénarios *Economical*, *Free-reloc* et *No-reloc* sont proches également, avec presque le même nombre de voitures, alors que le scénario *Economical* et surtout le *Free-reloc* présentent de plus grands pourcentages de satisfaction que le *No-reloc* (cf. figure 3.7). Ce résultat est dû à l'apport de la relocalisation, notamment dans le scénario *Free-reloc*.

— La figure 3.9 montre le taux de partage des voitures qui est le nombre de requêtes moyen par voiture. Les taux les plus élevés sont observés dans les scénarios *Economical* et *Free-reloc*, leq plus mauvais dans le scénario *Free-cars*, où un nombre important de voitures est déployé (ce qui réduit le taux de partage), et dans le scénario *No-reloc* où l'absence de relocalisation diminue fortement le taux de partage.

— La figure 3.10 donne la moyenne d'heures de relocalisation par agent. Nous pouvons observer que la moyenne d'heures de relocalisation par agent se situe entre 2h (cas d'une faible demande) et 3 h (cas d'une demande élevée) pour tous les scénarios, alors que la journée de service d'un agent est estimée à 8h. Il passe le reste de son temps à se déplacer entre les stations, à attendre des relocalisations ou à réaliser une autre activité dans les interstices temporels libres.

— La figure 3.11 fournit le nombre moyen de clients (requêtes) satisfaits par

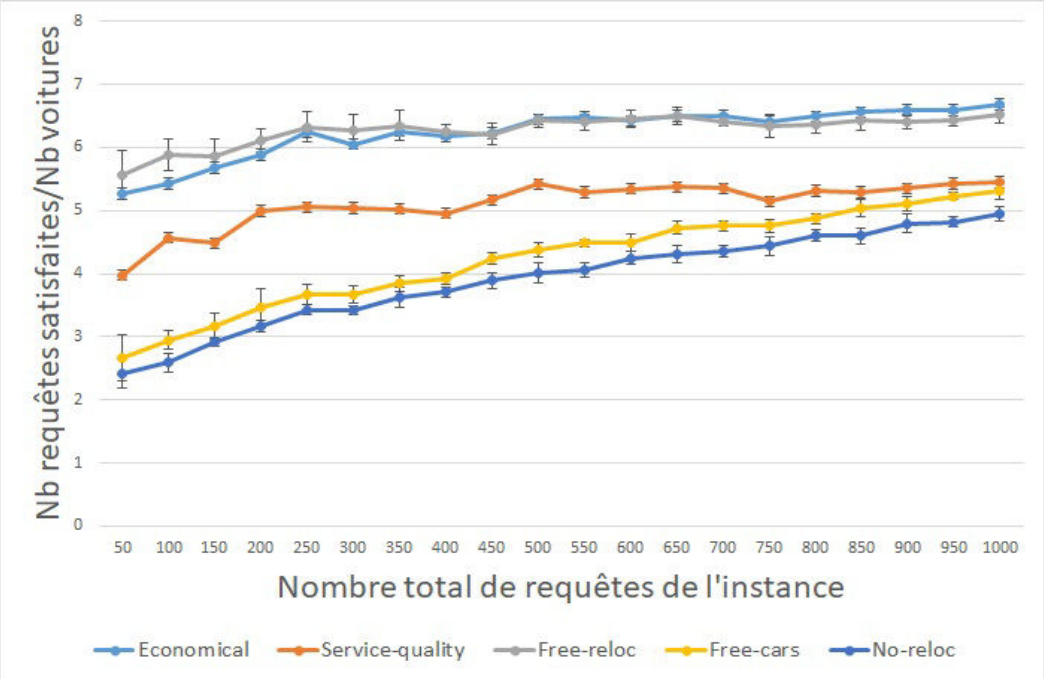


FIGURE 3.9 – *Résultats sur le taux de partage des voitures avec différents scénarios de fonctions objectif multi-critères*

agent déployé. Nous pouvons observer que ce nombre augmente assez linéairement pour tous les scénarios. Un agent est sollicité pour pouvoir satisfaire entre 10 à 15 requêtes (selon le scénario) dans le cas de faible demande. Dans le cas d'une forte demande, un agent peut suffire à satisfaire entre 20 à 40 requêtes (selon le scénario). Les deux scénarios *Service-quality* et *Free-reloc* sont les plus mauvais en terme de rapport entre le nombre d'agents déployés et le nombre de requêtes satisfaites. Dans ces deux scénarios, le coût de relocalisation est négligeable par rapport à celui de la satisfaction des clients et donc beaucoup d'opérations de relocalisation sont effectuées, pour une augmentation faible du pourcentage de satisfaction.

3.4.2 Impact de l'autonomie des voitures sur les performances du système

Avant d'analyser les figures 3.12 et 3.13, nous devons préciser que c'est la fonction objectif économique du scénario *Economical* qui a été utilisée pour obtenir les résultats dans ces deux figures. Celles-ci montrent des simulations pour trois scénarios avec différentes autonomies de voitures. Dans le scénario 1, nous avons fixé l'autonomie de toutes les voitures à 2 heures (8 périodes de temps), à 4 heures dans le scénario 2 et à 8 heures dans le scénario 3.

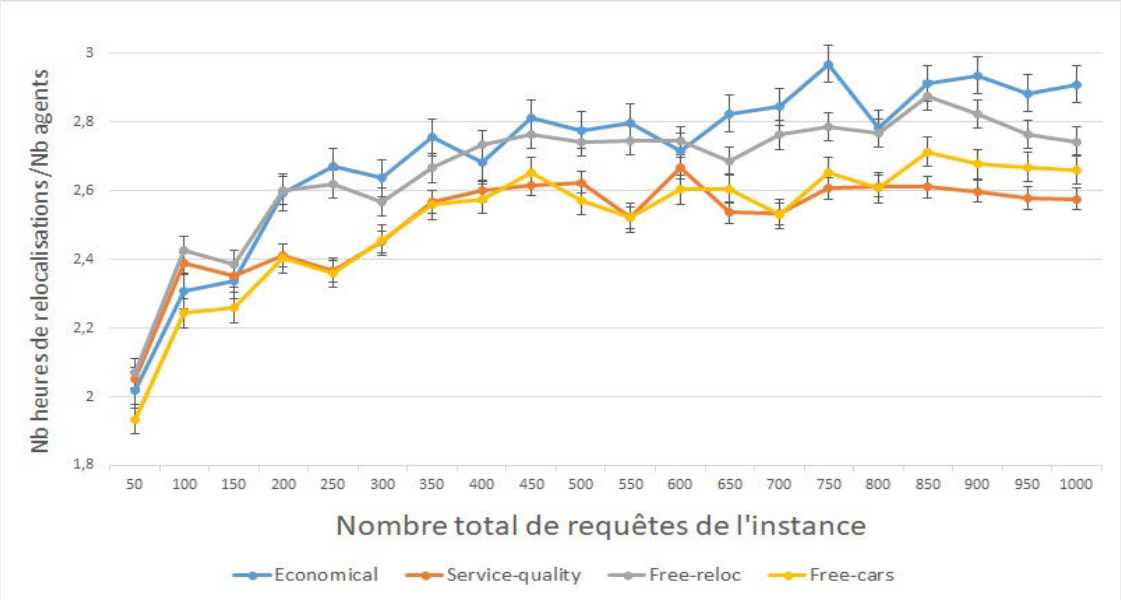


FIGURE 3.10 – *Résultats sur la moyenne d'heures de relocalisation par agent avec différents scénarios de fonctions objectif multi-critères*

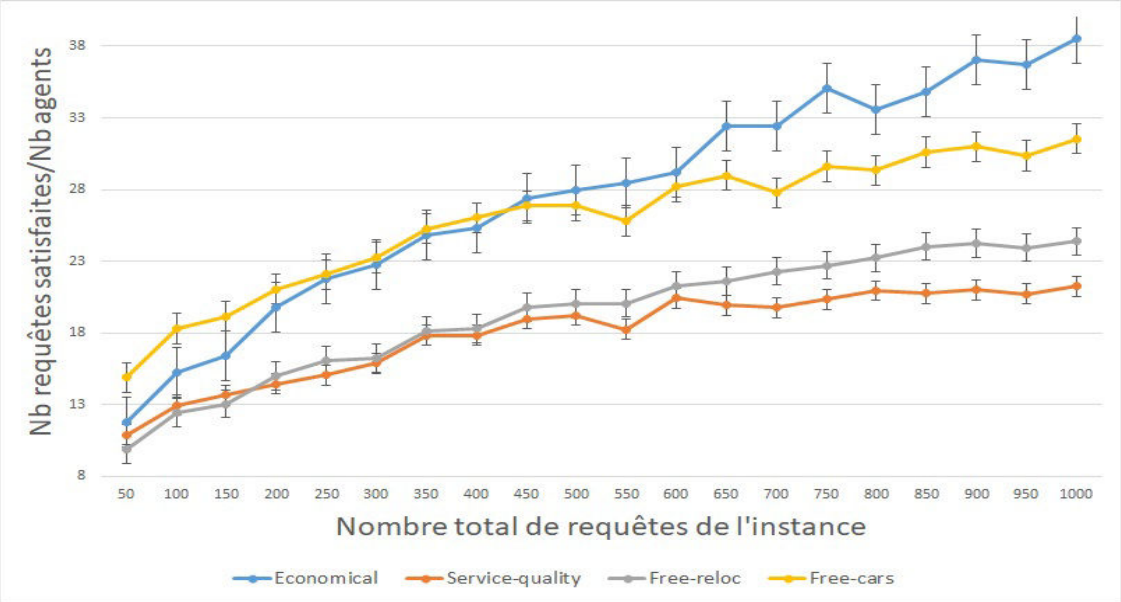


FIGURE 3.11 – *Rapport entre le nombre d'agents nécessaire pour effectuer les opérations de relocalisation et le nombre de clients satisfaits pour différents scénarios de fonctions objectif multi-critères*

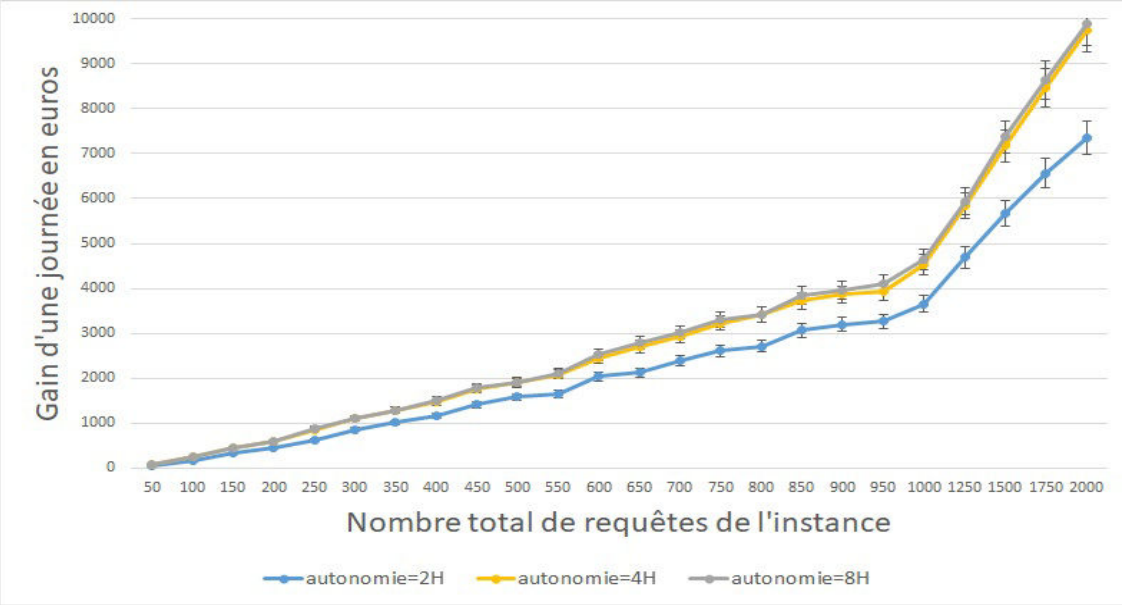


FIGURE 3.12 – *Gain d'une journée de service avec trois scénarios et différentes autonomies électriques des voitures (scénario 1 : 2 heures d'autonomie, scénario 2 : 4 heures d'autonomie, scénario 3 : 8 heures d'autonomie)*

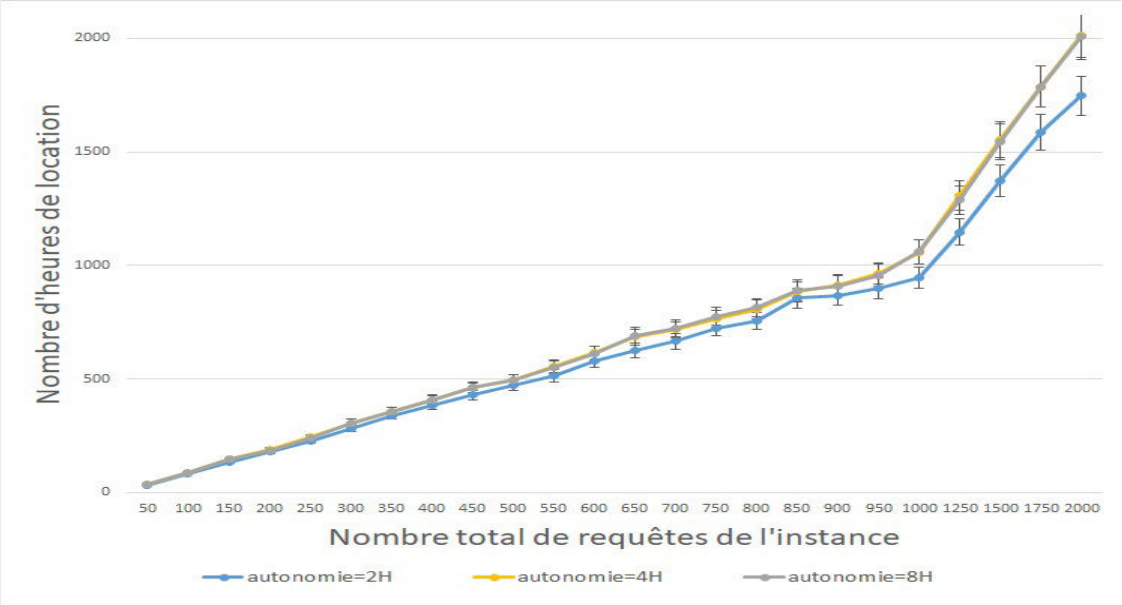


FIGURE 3.13 – *Temps total de location d'une journée de service avec trois scénarios et différentes autonomies électriques des voitures (scénario 1 : 2 heures d'autonomie, scénario 2 : 4 heures d'autonomie, scénario 3 : 8 heures d'autonomie)*

On peut observer dans les deux figures (3.12 et 3.13) que le gain et le temps de location ne sont pas vraiment affectés quand on passe de 8 heures à 4 heures

d'autonomie. En revanche, on observe une relative dégradation, quand on passe à 2 heures d'autonomie dans le scénario 1. Nous pouvons conclure que l'autonomie des voitures n'est pas un handicap majeur pour les performances d'un système d'auto-partage dans des conditions similaires à nos simulations, à condition d'utiliser des algorithmes d'optimisation capables de prendre en compte les contraintes de recharge électrique.

À partir des résultats de la simulation ci-dessus, nous pouvons tirer deux conclusions. Premièrement, la relocalisation des voitures est une stratégie importante pour améliorer la performance (augmentation de la satisfaction des clients et du taux d'utilisation de la voiture) du système d'auto-partage. Deuxièmement, la co-efficacité des différentes composantes de la fonction objectif doit être correctement choisie et ajustée par l'opérateur de transport, afin que différents critères puissent être satisfaits pour l'exploitation des systèmes réels.

3.5 Conclusion

Dans la première partie de chapitre, un algorithme génétique a été proposé pour la résolution du *ROCSP* avec contrainte de recharge *normale*. Ce dernier est basé sur une heuristique qui construit une solution en routant les voitures entre les stations, tout en prenant en compte les contraintes de recharge électrique et les contraintes de capacité des stations. L'algorithme génétique a donné des résultats satisfaisants avec un gap maximum de 6.5% sur les instances de moins de 15 stations où la comparaison était possible avec la solution exacte obtenue avec le modèle $R\text{-}PLNE^{ROCSP}$ proposé dans le chapitre précédant qui, même sur ces petites instances, nécessite des temps de calcul énormes, comparés à l'algorithme génétique.

Dans la deuxième partie de ce chapitre, nous avons abordé le problème de planning des agents qui sont chargés d'effectuer les opérations de relocalisation issues de la résolution du *ROCSP*. Nous avons proposé une heuristique constructive pour résoudre le problème de planning des agents, qui permet d'effectuer leur routage, avec la contrainte de temps de travail maximal pour chaque agent, ainsi qu'un algorithme de *Branch and Price* utilisé de manière heuristique qui a donné des solutions quasi-optimales et rapidement calculées, en comparaison avec le modèle $PLNE^{ESRP}$.

Les heuristiques développées pour le *ROCSP* et l'*ESRP* ont permis l'étude de cas de l'auto-partage de la ville de Nice, pour lequel, dans la dernière partie du chapitre, des simulations ont été menées et ont permis de valider la contribution de la relocalisation des voitures sur des services dans des situations similaires. Un objectif équilibré peut consister à trouver un bon compromis entre la maxi-

misation des utilisateurs satisfaits et la réduction des coûts logistiques (nombre d'agents et nombre de voitures). À cette fin, le poids de chacun de ces critères peut être modifié dans la fonction objectif et permet d'explorer plusieurs types de solutions. En outre, les simulations montrent que la façon dont nous pouvons combiner plusieurs critères dans la fonction objectif a un effet non négligeable sur la qualité des résultats. L'ajustement des poids de ces critères est alors très fructueux, car il peut aider les décideurs et les opérateurs à trouver des solutions fiables pour optimiser leur système d'auto-partage, via un système d'aide à la décision adapté à leurs besoins.

Chapitre 4

Résolution du problème de relocalisation dans l'auto-partage *à un seul sens* avec réservations dynamiques et concept d'Auto-CoPartage

Sommaire

4.1 Introduction

Dans la première partie de ce chapitre, nous commençons par définir le problème de réservation dynamique dans l'auto-partage *à un seul sens*, où l'ensemble des réservations n'est pas connu à l'avance. L'objectif est de proposer un système de réservation qui permet de répondre rapidement aux requêtes des clients qui arrivent dynamiquement. Pour ce faire, on va résoudre le *ROCSP* (cf. 2.3.1) à l'arrivée de chaque nouvelle requête. Dans la seconde partie de ce chapitre, un nouveau concept de transport nommé Auto-CoPartage est proposé, avec lequel nous investiguons une nouvelle forme d'hybridation d'auto-partage et de covoiturage. Il sera abordé de façon exploratoire, en adaptant le système de réservation en temps réel proposé pour l'auto-partage.

4.2 Problème avec réservations dynamiques

Nous commençons par décrire les spécificités du problème dynamique qui est par la suite décomposé en un ensemble de problèmes statiques à chaque instant du processus de réservation. Dans un premier temps, le modèle $F\text{-}PLNE^{ROCSP}$ est adapté pour modéliser le problème statique à chaque instant. Par la suite, un système de réservation basé sur une génération de colonnes est utilisé pour résoudre le problème dynamique, avec des temps rapides de réponse aux requêtes des clients.

4.2.1 Description du problème

Jusqu'à présent, dans la résolution du *ROCSP*, l'ensemble des requêtes de réservation pour la journée est supposé connu à l'avance et un sous-ensemble de requêtes à satisfaire peut être choisi. Le but était de résoudre un problème statique avec des décisions de niveau stratégique et d'évaluer le nombre de voitures et les relocalisations nécessaires, afin de maximiser le gain économique ou le nombre de clients satisfaits. Dans ce chapitre, on se place dans un contexte de décision de niveau opérationnel, en supposant un nombre de voitures fixe (que nous ne cherchons pas à minimiser), alors que l'ensemble des réservations n'est pas connu à l'avance. On souhaite concevoir un système de réservation dynamique qui maximise le nombre de requêtes satisfaites. Les demandes de réservation n'arrivent pas au même moment, comme l'illustre la figure 4.1, ce qui crée un problème dynamique où la décision d'accepter ou de refuser une réservation doit être faite après l'arrivée de chaque requête, sans connaître les

futures demandes de réservation. En réalité, il est difficile d'avoir des informations prédictives ou une distribution exacte de probabilité sur les demandes.

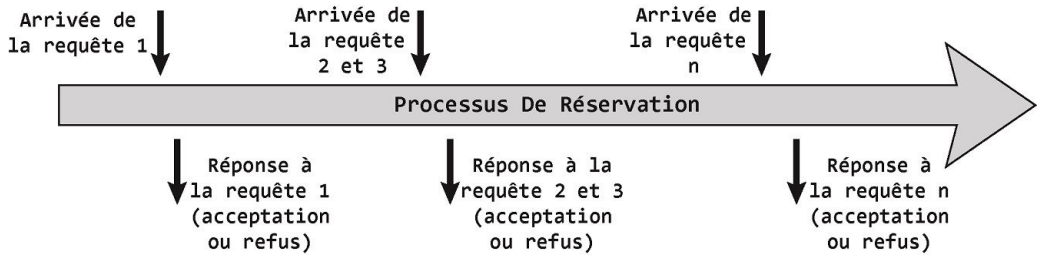


FIGURE 4.1 – *Arrivée dynamique des requêtes dans le processus de réservation*

Pour s'adapter à la réalité de la gestion opérationnelle, en plus des données et des contraintes définies précédemment pour le *ROCSP*, on considère des fenêtres de temps $[T^d_{q_i}, T^{d'}_{q_i}]$ et $[T^f_{q_i}, T^{f'}_{q_i}]$ au début et à la fin de chaque requête respectivement, à l'intérieur desquelles le client peut prendre et restituer sa voiture. Plus précisément, on bloque une voiture pendant un intervalle de temps au départ de chaque requête, ainsi qu'une place dans la station de restitution à la fin de la requête. Formellement, pour chaque requête acceptée q_i, une voiture est réservée dans la station de départ $S^d_{q_i}$ pour l'intervalle de temps $[T^d_{q_i}, T^{d'}_{q_i}]$ (fixé à 30 min dans nos simulations) et une place de restitution est réservée dans la station d'arrivée $S^f_{q_i}$ pour l'intervalle de temps $[T^f_{q_i}, T^{f'}_{q_i}]$ (fixé à 90 min). Les durées des intervalles de réservation sont fixées en se basant sur les durées du système de réservation d'Autolib en Ile de de France.

Transformation du problème de réservation dynamique global en un ensemble de problèmes statiques à chaque moment du processus de réservation

On définit l'ensemble M qui représente l'horizon temporel des réservations, qui débute la veille de la journée de service et se termine avant le début de la journée de service. Chaque $m \in M$ correspond au moment d'arrivée au système de réservation d'un nouvel ensemble de requêtes. Cet ensemble est non vide et peut contenir une ou plusieurs requête(s) qui arrivent à la fois. Nous transformons le problème dynamique global en un ensemble de problèmes d'optimisation statiques $ROCSP_m$ avec $m \in M$, à résoudre les uns après les autres. L'ensemble des requêtes Q_m arrivées avant le moment m contient deux sous-ensembles : Q'_m l'ensemble des requêtes acceptées à m et Q''_m l'ensemble des requêtes non encore traitées à m. Tous les ensembles sont vides au moment $m = 0$.

Pour plus de clarté, nous donnons un exemple illustratif dans la figure 4.2 où nous avons 20 requêtes pendant un jour complet de service qui constituent l'en-

semble $Q = \{q_1, q_2, ...q_{20}\}$. Soit le moment m qui correspond au même moment d'arrivée des trois requêtes q_7, q_8 et q_9. À ce moment m, le système a déjà :

— reçu 9 requêtes qui constituent l'ensemble $Q_m = \{q_1, q_2, ...q_9\}$;

— traité les requêtes de q_1 à q_6 ;

— accepté l'ensemble des requêtes $Q'_m = \{q_1, q_2, q_3, q_5\}$;

— refusé les requêtes q_4 et q_6.

Ici, le problème statique $ROCSP_m$, considéré à l'instant m, vise à satisfaire le nombre maximum possible de requêtes dans l'ensemble $Q''_m = \{q_7, q_8, q_9\}$, avec la contrainte de satisfaction de toutes les requêtes dans Q'_m.

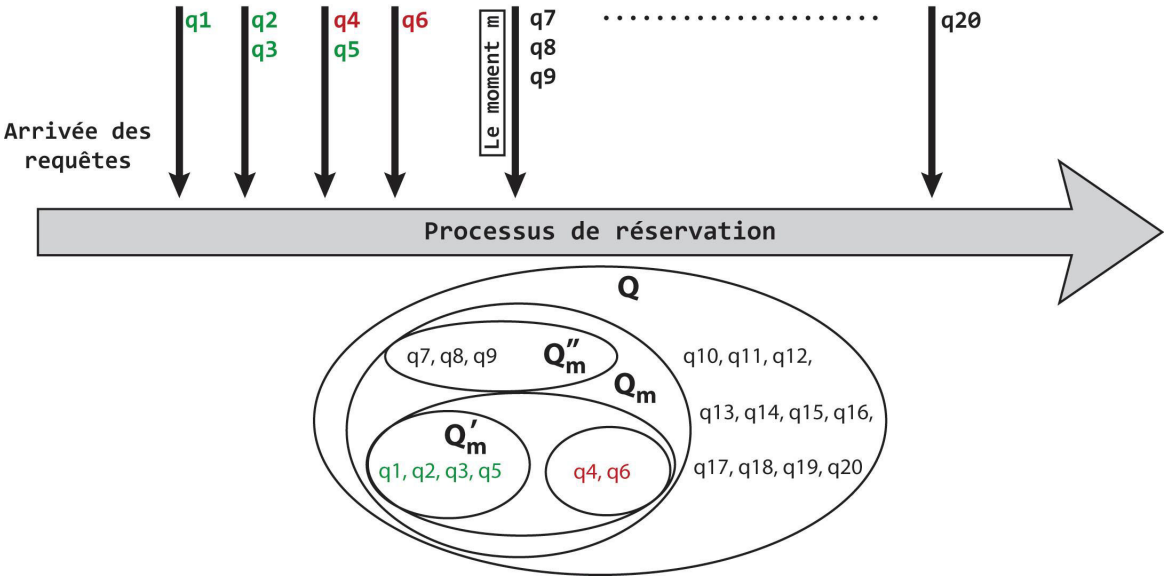


FIGURE 4.2 – *Exemple de classification des requêtes en sous-ensembles en fonction de leur moment d'arrivée dans le processus de réservation*

4.2.2 Système de réservation basé sur un modèle PLNE

Conformément au processus de réservation décrit dans 4.2.1, un système de réservation dynamique consiste à résoudre l'ensemble des problèmes statiques $ROCSP_m$ avec $m \in M$. Pour modéliser chaque $ROCSP_m$, nous avons le choix entre les deux modèles $F\text{-}PLNE^{ROCSP}$ et $R\text{-}PLNE^{ROCSP}$. Mais au vu des limites de résolution que présente le second dans les simulations du chapitre précédent où l'on a été contraint de s'arrêter à de petites instances, le choix s'est donc porté sur le premier modèle, même s'il impose une contrainte de recharge plus forte que dans le cas réel. En effet, le modèle $F\text{-}PLNE^{ROCSP}$ comporte un

nombre de variables beaucoup moins important et donc des temps de résolutions plus raisonnables, ce qui nous permet de l'utiliser comme référence pour évaluer l'heuristique de génération de colonnes proposée dans la section suivante. Afin de simplifier la résolution du problème $ROCSP_m$, nous modifions le graphe spatio-temporel $G^{ROCSP}(V, U, R)$ (cf. 2.4.1) pour prendre en compte les fenêtres de temps de réservation dans les stations de départs et d'arrivées des clients. En conséquence, nous incluons les contraintes de réservation dans la structure du graphe et non dans le modèle PLNE. Plus précisément, nous définissons des arcs dans notre graphe qui interdisent le déplacement des voitures réservées, en modifiant l'ensemble $U_4 \in U$ des arcs représentant les requêtes.

$U_4 = \{(v^{T^d_{q_i}}_{S^d_{q_i}}, v^{T^{f'}_{q_i}}_{S^f_{q_i}}) : q_i \in Q\}$; pour chaque requête de location q_i, nous associons l'arc $(v_{S^d_{q_i} T^d_{q_i}}, v_{S^f_{q_i} T^{f'}_{q_i}})$ dont ses deux extrémités sont définies par :

— $v^{T^d_{q_i}}_{S^d_{q_i}}$ où $S^d_{q_i}$ représente la station de départ et $T^d_{q_i}$ représente le début de l'intervalle de temps de réservation de départ $[T^d_{q_i}, T^{d'}_{q_i}]$;

— $v^{T^{f'}_{q_i}}_{S^f_{q_i}}$ où $S^f_{q_i}$ représente la station d'arrivée et $T^{f'}_{q_i}$ représente la fin de l'intervalle de temps de réservation d'arrivée $[T^f_{q_i}, T^{f'}_{q_i}]$.

Le fait d'utiliser les deux extrémités les plus éloignées des deux intervalles pour représenter l'arc nous permet de réserver la voiture pour tout le temps de la location de la requête q_i, y compris les intervalles de réservation.

Pour adapter la fonction objectif Ct au $ROCSP_m$, nous fixons α le coefficient du coût des voitures à zéro, vu que le nombre de voitures est fixe et qu'on ne cherche pas à le minimiser dans le cas dynamique, ce qui donne $Ct = \beta \cdot C_R - \gamma \cdot C_C$. L'adaptation du $F\text{-}PLNE^{ROCSP}$ est donné par le modèle $F\text{-}PLNE^{ROCSP_m}$ avec, pour rappel, $f_{(v^t_{s_i}, v^{t'}_{s_k})}$ la variable de décision qui donne le nombre de voitures sur l'arc $(v^t_{s_i}, v^{t'}_{s_k})$ et z_{q_i} la variable binaire qui dit si la requête $q_i \in Q$ est satisfaite. Nous définissons également la constante $\theta^{q_l}_{(v^t_{s_i}, v^{t+1}_{s_i})} : (q_l \in Q, s_i \in S, t \in T \setminus \{T^{max}\})$, qui est égale à 1 si une place de stationnement dans la station s_i entre t et $t+1$ est réservée à la requête q_l, 0 sinon.

$$\theta^{q_l}_{(v^t_{s_i}, v^{t+1}_{s_i})} = \begin{cases} 1 & \text{si } (T^d_{q_l} \leq t < T^{d'}_{q_l} \wedge s_i = S^d_{q_l}) \vee (T^f_{q_l} \leq t < T^{f'}_{q_l} \wedge s_i = S^f_{q_l}) \\ 0 & \text{sinon} \end{cases}$$

$$\min \qquad Ct = \beta \cdot C_R - \gamma \cdot C_C \qquad (F\text{-}PLNE^{ROCSP_m})$$

avec :

— $C_R = \sum_{(v^t_{s_i}, v^{t'}_{s_k}) \in U_3} T_{s_i s_k} \cdot f_{(v^t_{s_i}, v^{t'}_{s_k})}$

— $C_C = \sum_{q_i \in Q_m} G_{q_i} \cdot z_{q_i}$

sous contraintes de (4.1) à (4.5).

$$\sum_{v^t_{s_i} \in V} f_{(v^t_{s_i}, v^{t'}_{s_k})} + \sum_{q_i \in Q_m : S^f_{q_i} = s_k, T^{f'}_{q_i} = t'} z_{q_i} = \sum_{v^{t''}_{s_l} \in V} f_{(v^{t'}_{s_k}, v^{t''}_{s_l})} + \sum_{q_i \in Q_m : S^d_{q_i} = s_k, T^d_{q_i} = t'} z_{q_i}$$

$$\forall v^{t'}_{s_k} \in V \setminus \{V^d, V^f\} \qquad (4.1)$$

$$\sum_{(V^d, v^t_{s_i}) \in U_1} f_{(V^d, v^t_{s_i})} = W^{max} \qquad (4.2)$$

$$f_{(v^t_{s_i}, v^{t+1}_{s_i})} + \sum_{q_l \in Q_m} \theta^{q_l}_{(v^t_{s_i}, v^{t+1}_{s_i})} \cdot z_{q_l} \leq K_{s_i} \qquad \forall (v^t_{s_i}, v^{t+1}_{s_i}) \in U_2 \qquad (4.3)$$

$$f_{(v^t_{s_i}, v^{t+1}_{s_i})} \geq \sum_{(v^{t'}_{s_k}, v^{t''}_{s_i}) \in U_3 : t'' \leq t < t'' + T^r_{s_k s_i}} f_{(v^{t'}_{s_k}, v^{t''}_{s_i})} + \sum_{q_l \in Q_m : S^f_{q_l} = s_i, T^{f'}_{q_l} \leq t < T^{f'}_{q_l} + T^r_{q_l}} z_{q_l}$$

$$\forall s_i \in S, t \in T \setminus \{T^{max}\} \qquad (4.4)$$

$$z_{q_i} = 1 \qquad \forall q_i \in Q'_m \qquad (4.5)$$

L'équation (4.1) assure la contrainte de continuité du flux des voitures. L'équation (4.2) garantit que le flux total des voitures ne dépasse pas le nombre de voitures. L'équation (4.3) garantit la contrainte de capacité à chaque station et à chaque période de temps du jour de service. L'équation (4.4) permet de garantir que les voitures se rechargent après chaque trajet effectué pendant une durée proportionnelle au trajet. Enfin, la dernière équation ajoutée au modèle (4.5) oblige à satisfaire les clients dont les requêtes ont été acceptées avant le moment m.

4.2.3 Système de réservation basé sur une Génération de Colonnes (GC)

Dans le système de réservation proposé, une heuristique de génération de colonnes a été choisie pour se substituer au modèle $F\text{-}PLNE^{ROCSP_m}$ dans le processus de réservation. Ce choix a été motivé par le fait que les modèles PLNE

risquent de ne pas pouvoir répondre rapidement aux requêtes dynamiques des clients. Une alternative aurait été d'utiliser l'heuristique constructive développée pour le *ROCSP* statique (cf. 3.2), mais cette dernière ne permet pas d'ajouter la contrainte d'obligation de satisfaction des requêtes acceptées dans le passé d'un moment m donné. En revanche, une génération de colonnes nous permet d'ajouter cette contrainte dans le problème maître, tout en donnant des temps de réponse plus raisonnables que ceux du PLNE. En effet, la génération de colonnes est adaptée au traitement d'un plus grand nombre de variables que ne peut le faire le modèle PLNE. Le choix a été aussi motivé par la possibilité d'exploiter les colonnes générées depuis le début du processus de réservation. La génération de colonnes que nous allons définir est une sorte de recherche locale dans le sens où l'on peut passer d'une solution au moment m à une autre voisine au moment $m+1$ par l'ajout d'une (ou plusieurs) colonne(s).

Le système de réservation se base sur une heuristique de génération de colonnes où chaque colonne est un chemin dans le graphe spatio-temporel et le $ROCSP_m$ est décomposé en deux sous problèmes à chaque moment m du processus de réservation : le problème maître donné prend en charge les contraintes de capacité des stations et choisit les meilleures colonnes à utiliser dans la solution ; le problème du *Pricing* présenté plus bas, sert à calculer des colonnes de coût réduit négatif qui peuvent améliorer la solution. Dans notre système de réservation, on passe d'un moment m à un moment $m+1$, en ajoutant les requêtes arrivées à $m+1$ et en générant des nouvelles colonnes issues de la résolution du problème du *Pricing* au moment $m+1$ tout en gardant les anciennes colonnes générées depuis le début du processus de réservation. Pour alléger le problème maître, on supprime les anciennes colonnes non utilisées dans les solutions des problèmes maîtres correspondant au 20 derniers moments $m \in M$. Le nombre de 20 a été choisi après un certain nombre de simulations, car on a pu observer qu'en moyenne, si une colonne n'est pas utilisée depuis 20 moments, alors elle a peu de chance d'être réutilisée à nouveau dans une solution du problème maître. Cela s'explique par le fait que les nouvelles requêtes arrivées lors des 20 derniers moments ont changé la nature de l'instance du problème et que les anciennes tournées correspondant à des colonnes non utilisées ne sont plus exploitables. Une requête est acceptée dès qu'une colonne qui la prend en charge est générée et refusée au bout de 10 itérations de la GC sans pouvoir la satisfaire. Ce chiffre a été aussi fixé après l'observation de la faible probabilité de satisfaire la requête en question après 10 itérations. Le pseudo-code du système de réservation est donné par l'algorithme 6 et la génération de colonne est détaillée ci-dessous.

Algorithm 6: Pseudo code du système de réservation basé sur la GC

initialisation;
$m \leftarrow 0$;
$Q_m, Q'_m, Q''_m \leftarrow \emptyset$;
$j \leftarrow 0$;
initialiser le problème maître en ajoutant les colonnes (chemins) représentant les voitures restant à la même station toute la journée ;
solution = résoudre le problème maître relaxé ;
while $m \in M$ **do**
  if *arrivée de nouvelles requêtes dans le système* **then**
    $m \leftarrow m + 1$;
    $Q''_m \leftarrow Q''_{m-1} \cup \{$ nouvelles requêtes $\}$;
    $Q_m \leftarrow Q_{m-1} \cup Q''_m$;
    for $q_i \in Q''_m \setminus Q''_{m-1}$ **do**
      $arrival_{q_i} \leftarrow j$
  `/* résoudre le probleme du pricing */`
  $path_j \leftarrow ESPPRC()$;
  if $CR_j < 0$ **then**
    $P \leftarrow P \cup path_j$;
  résoudre le problème maître relaxé ;
  if *solution non entière* **then**
    résoudre le problème maître entier ;
  for $q_i \in Q''_m$ **do**
    `/* Accepter la requête */`
    if $(\exists l \in P : (x_l == 1 \wedge \beta^l_{q_i} == 1))$ **then**
      $Q'_m \leftarrow Q'_m \cup \{q_i\}$;
      $Q''_m \leftarrow Q''_m \setminus \{q_i\}$;
    `/* Refuser la requête */`
    if $(!\exists l \in P : (x_l == 1 \wedge \beta^l_{q_i} == 1) \wedge j - arrival_{q_i} > 10)$ **then**
      $Q_m \leftarrow Q_m \setminus \{q_i\}$;
      $Q''_m \leftarrow Q''_m \setminus \{q_i\}$;
  Supprimer les colonnes $path_l$ non utilisées depuis 20 itérations pour alléger la résolution du problème maître ;
  $j = j + 1$;

Problème maître

Contrairement au modèle $F\text{-}PLNE^{ROCSP_m}$, le problème maître est basé sur les chemins et non sur les flots, permettant de considérer une contrainte de re-

charge *normale*. Cela n'empêche pas son exploitation dans un cadre de système de réservation dynamique, où le temps de résolution est un paramètre très important. En effet, la décomposition du problème en deux sous problèmes maître et *Pricing* permet de gérer, dans le problème maître, un petit nombre de variables, car l'ensemble des variables P est généré progressivement. À l'état initial, avant l'arrivée de toutes les requêtes, l'ensemble P comporte les variables (chemins des voitures dans le graphe) qui représentent des voitures garées dans la même station toute la journée, de telle sorte que dans la solution initiale, le nombre global de voitures est divisé équitablement entre les stations.

Donc, soit P l'ensemble des chemins générés entre V^d et V^f dans le graphe $G^{ROCSP}(V, U, R)$ et soit :

— $\alpha^j_{(v^t_{s_i}, v^{t'}_{s_k})}$: égale à 1 si le chemin numéro $j \in P$ utilise l'arc $(v^t_{s_i}, v^{t'}_{s_k}) \in U \setminus U_4$, 0 sinon ;

— $\beta^j_{q_i}$: égale à 1 si le chemin numéro $j \in P$ utilise l'arc $(v^{T^d_{q_i}}_{S^d_{q_i}}, v^{T^{f'}_{q_i}}_{S^f_{q_i}}) \in U_4$, 0 sinon.

Le problème maître est donné par le modèle PM^{ROCSP_m} et se base sur le vecteur de variables de décision $x_j : j \in P$, qui nous permet de sélectionner l'ensemble des chemins utilisés dans la solution.

— $x_j \in \{0, 1\}$ est égale à 1 si le chemin numéro j est utilisé, 0 sinon.

On a :

$$\min Ct = \beta \cdot C_R - \gamma \cdot C_C \qquad (PM^{ROCSP_m})$$

avec :

— $C_R = \sum_{j \in P} \sum_{(v^t_{s_i}, v^{t'}_{s_k}) \in U_3} T_{s_i s_k} \cdot \alpha^j_{(v^t_{s_i}, v^{t'}_{s_k})} \cdot x_j$

— $C_C = \sum_{j \in P} \sum_{q_i \in Q_m} \beta^j_{q_i} \cdot G_{q_i} \cdot x_j$

sous contraintes de (4.6) à (4.9).

$$\sum_{j\in P} \alpha^{j}_{(v^t_{s_i}, v^{t+1}_{s_i})} \cdot x_j + \sum_{j\in P} \sum_{q_l \in Q_m} \theta^{q_l}_{(v^t_{s_i}, v^{t+1}_{s_i})} \cdot \beta^{j}_{q_l} \cdot x_j \leq K_{s_i} \qquad \forall (v^t_{s_i}, v^{t+1}_{s_i}) \in U_2 \quad (4.6)$$

$$\sum_{j\in P} \beta^{j}_{q_i} \cdot x_j = 1 \qquad \forall q_i \in Q'_m \quad (4.7)$$

$$\sum_{j\in P} \beta^{j}_{q_i} \cdot x_j \leq 1 \qquad \forall q_i \in Q''_m \quad (4.8)$$

$$\sum_{j\in P} x_j = W^{max} \quad (4.9)$$

L'équation (4.6) assure la contrainte de capacité, l'équation (4.7) vérifie que toutes les requêtes déjà acceptées seront satisfaites et l'équation (4.8) permet que chaque nouvelle requête soit satisfaite par au maximum une seule voiture. Enfin, l'équation (4.9) permet de garantir que le nombre de colonnes utilisées est égal au nombre de voitures.

Problème du *Pricing*

Nous utilisons le problème du *Pricing* pour trouver des colonnes avec un coût réduit négatif et les ajouter à l'ensemble P. Le problème du *Pricing* est le problème de plus court chemin avec contraintes de ressource de V^d à V^f dans le graphe $G^{ROCSP}(V, U, R)$, résolu avec l'algorithme 3 (*ESPPRC*) présenté dans le chapitre précédent. Le coût réduit d'une colonne $\bar{CR}_j$ est calculé en utilisant les variables duales $\pi_{(v^t_{s_i}, v^{t+1}_{s_i})}$, μ'_{q_i}, μ''_{q_i} et κ.

$$\begin{aligned}\bar{CR}_j =& \beta \cdot \Big(\sum_{(v^t_{s_i}, v^{t'}_{s_k}) \in U_3} T_{s_i s_k} \cdot \alpha^{j}_{(v^t_{s_i}, v^{t'}_{s_k})} \Big) - \gamma \cdot \Big(\sum_{q_i \in Q_m} \beta^{j}_{q_i} \cdot G_{q_i} \Big) \\ & - \sum_{(v^t_{s_i}, v^{t+1}_{s_i}) \in U_2} \left(\alpha^{j}_{(v^t_{s_i}, v^{t+1}_{s_i})} + \sum_{q_l \in Q_m} \theta^{q_l}_{(v^t_{s_i}, v^{t+1}_{s_i})} \cdot \beta^{j}_{q_l} \right) \cdot \pi_{(v^t_{s_i}, v^{t+1}_{s_i})} \\ & - \sum_{q_i \in Q'_m} \beta^{j}_{q_i} \cdot \mu'_{q_i} - \sum_{q_i \in Q''_m} \beta^{j}_{q_i} \cdot \mu''_{q_i} - \kappa\end{aligned}$$

où :

— $\pi_{(v^t_{s_i}, v^{t+1}_{s_i})}$ sont les variables duales correspondantes aux contraintes (4.6) ;

— μ'_{q_i} sont les variables duales correspondantes aux contraintes (4.7) ;

— μ''_{q_i} sont les variables duales correspondantes aux contraintes (4.8) ;

— κ variable duale correspondante à la contrainte (4.9).

Nous transformons la première formulation du coût réduit pour avoir un coût sur chaque arc pour calculer le plus court chemin élémentaire avec contraintes de ressource dans le graphe $G^{ROCSP}(V, U, R)$.

$$\begin{aligned} C\bar{R}_{jm} = &- \kappa + \sum_{(v^t_{s_i}, v^{t'}_{s_k}) \in U_3} \alpha^j_{(v^t_{s_i}, v^{t'}_{s_k})} \cdot (\beta \cdot T_{s_i s_k}) \\ &+ \sum_{(v^t_{s_i}, v^{t+1}_{s_i}) \in U_2} \alpha^j_{(v^t_{s_i}, v^{t+1}_{s_i})} \cdot (-\pi_{(v^t_{s_i}, v^{t+1}_{s_i})}) \\ &+ \sum_{q_l \in Q'_m} \beta^j_{q_l} \cdot \left(-\mu'_{q_l} - \sum_{(v^t_{s_i}, v^{t+1}_{s_i}) \in U_2} (\pi_{(v^t_{s_i}, v^{t+1}_{s_i})} \cdot \theta^{q_l}_{(v^t_{s_i}, v^{t+1}_{s_i})}) - \gamma \cdot G_{q_l} \right) \\ &+ \sum_{q_l \in Q''_m} \beta^j_{q_l} \cdot \left(-\mu''_{q_l} - \sum_{(v^t_{s_i}, v^{t+1}_{s_i}) \in U_2} (\pi_{(v^t_{s_i}, v^{t+1}_{s_i})} \cdot \theta^{q_l}_{(v^t_{s_i}, v^{t+1}_{s_i})}) - \gamma \cdot G_{q_l} \right) \end{aligned}$$

4.2.4 Simulations

Cette partie, concernant les simulations, permet d'évaluer le système de réservation basé sur la génération de colonnes. En premier lieu est évalué le temps de réponse du système de réservation, paramètre très important dans le cas dynamique. Ensuite, nous passons à l'évaluation de la qualité des solutions. Dans les simulations, le coefficient γ du gain unitaire de location de voitures aux clients est fixé à 10 fois β, le coefficient du coût unitaire des distances parcourues pendant les relocalisations. De ce fait, la priorité est donnée à l'acceptation des requêtes. Vient ensuite la minimisation des coûts de relocalisation. Ce choix s'explique par l'inutilité de l'utilisation d'une fonction objectif économique dans la résolution dynamique, car cela reviendrait à refuser par exemple un client engendrant beaucoup de relocalisations dans la solution partielle. Or, l'ensemble des relocalisations de la solution partielle, au moment de l'arrivée d'une réservation donnée, n'est pas forcément le même que l'ensemble obtenu après la fin des réservations. Il peut même diminuer, car de nouvelles réservations peuvent intervenir dans le sens des relocalisations prévues. Les temps d'arrivées des requêtes dans le système de réservation sont générés aléatoirement à l'intérieur d'un horizon temporel de 3 heures qui permet de simuler le système de réservation.

Évaluation du temps de réponse aux requêtes du système de réservation basé sur la génération de colonnes

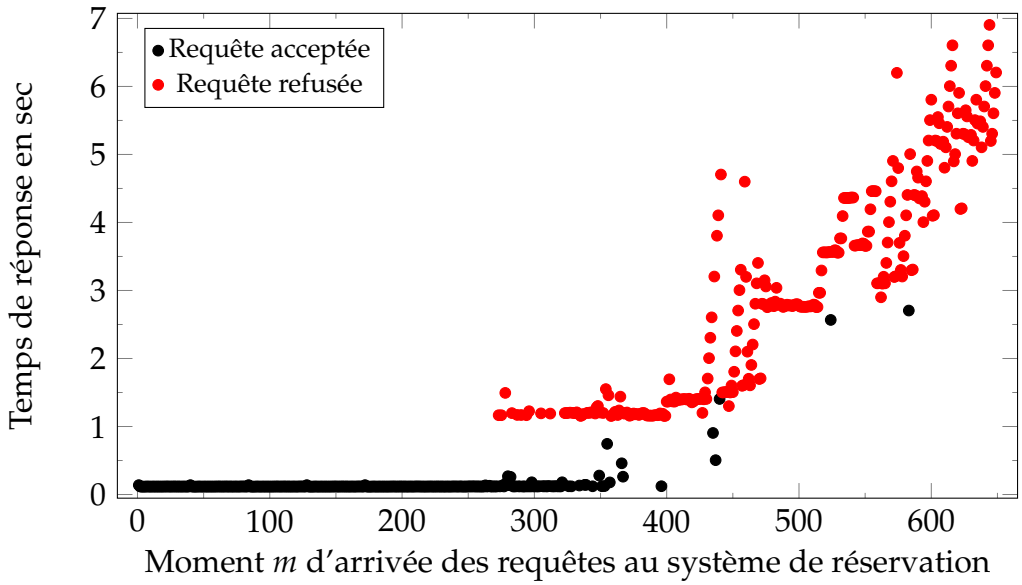


FIGURE 4.3 – *Traitement et temps de réponse aux requêtes dynamiques simulées d'auto-partage de la ville de Nice*

Prenons la ville de Nice (France) comme premier exemple. La figure 4.3 montre le temps de réponse pour plus de 600 requêtes envoyées consécutivement au système. nous pouvons voir que jusqu'aux 400 premières demandes, le temps de réponse des demandes acceptées est inférieur à une seconde, alors que le temps de réponse pour les demandes refusées est approximativement supérieur à une seconde. Cela est dû au fait qu'une demande refusée consomme plus d'itérations : l'algorithme essaie de trouver une solution réalisable, jusqu'à ce qu'il atteigne un critère d'arrêt. Au-delà des 400 premières demandes, nous pouvons remarquer que seulement quelques requêtes sont acceptées et que le temps de réponse est plus long. Tout cela peut s'expliquer par la saturation du système. Nous pouvons aussi observer que le nombre de requêtes acceptées a considérablement diminué par rapport aux résultats du chapitre précédant. Cela s'explique par l'ajout des fenêtres de temps de réservation, qui fait passer la durée moyenne d'une requête d'une heure à 3 heures. Ajoutons à cela les places de stationnement réservées, qui saturent le système.

La figure 4.4 montre le temps nécessaire pour répondre aux demandes avec notre système de réservation dynamique. Les tests sont effectués sur plusieurs instances de différentes tailles présentées dans 2.4, car la taille de l'instance affecte le temps de réponse. Nous pouvons voir que pour de petites instances de 13 stations (Bari) à 45 stations (Dublin), le temps d'acceptation et de refus est

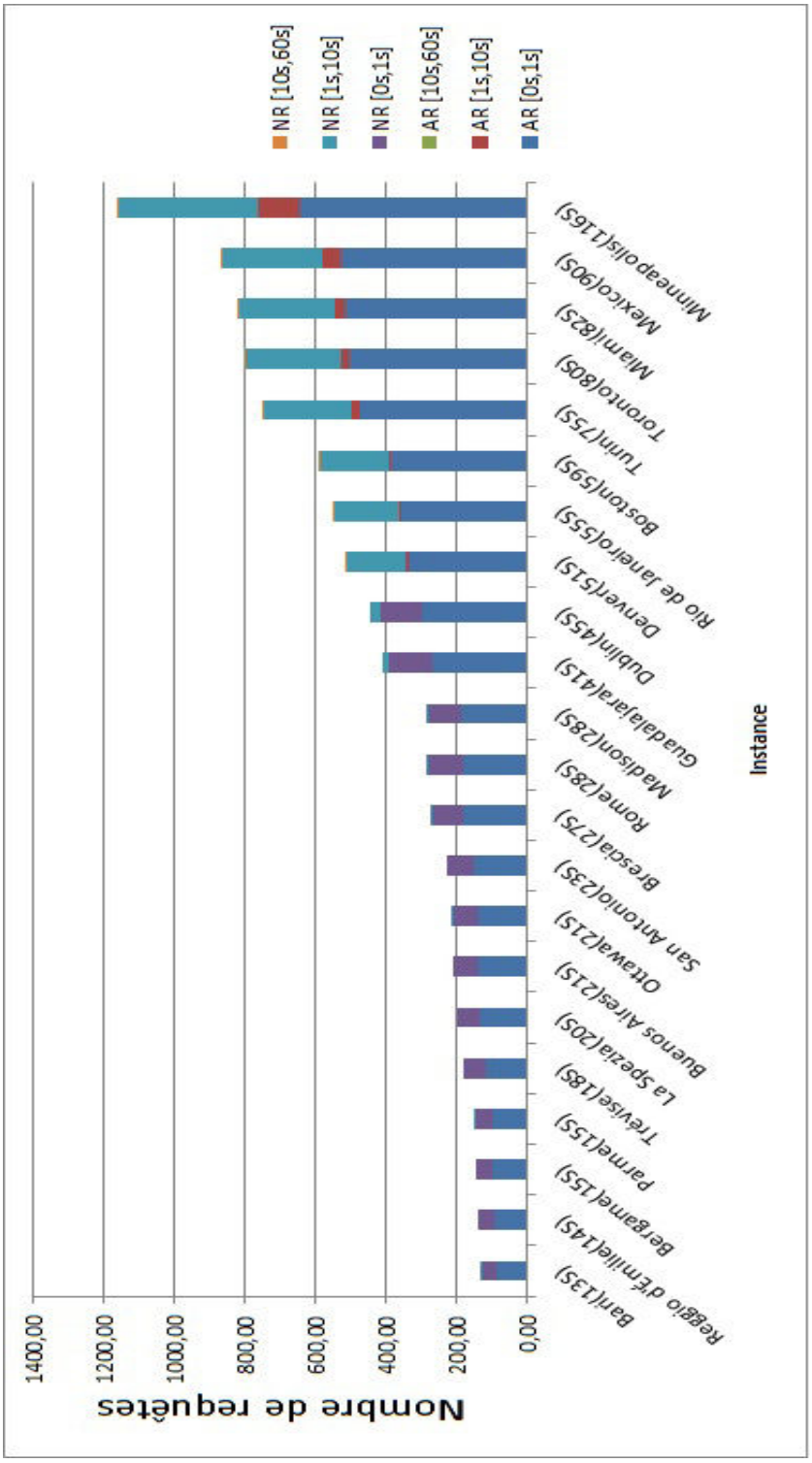


FIGURE 4.4 – *Temps de réponse du système de réservation aux requêtes dynamiques simulées (AR : nombre de requêtes acceptées ; NR : nombre de requêtes non acceptées)*

inférieur à une seconde. Pour les instances de plus grande taille de 51 stations (Denver) à 116 stations (Minneapolis), environ 90% des requêtes acceptées sont traitées en moins d'une seconde, 10% des requêtes restantes nécessitent entre 1 et 10 secondes pour être acceptées. Le temps des refus est généralement plus important car le refus est émis après 10 itérations (10 colonnes générées). Plus l'instance est grande, plus l'itération est coûteuse en temps.

Évaluation de la qualité des solutions

Pour évaluer la qualité des solutions du système de réservation basé sur l'heuristique de la GC, nous allons procéder en deux étapes, en utilisant comme référence le modèle $F\text{-}PLNE^{ROCSP_m}$ de la façon suivante :

— Étape 1 (*GC dynamique vs PLNE statique*). Nous utilisons comme référence le modèle $F\text{-}PLNE^{ROCSP_m}$, une seule fois au moment $m = 0$ pour résoudre le problème de réservation statique. Pour revenir à un problème statique, nous supposons qu'on dispose de toutes les requêtes à l'avance, c'est à dire que toutes les requêtes arrivent au moment $m = 0$ et on a donc $Q_0 = Q$ uniquement pour le modèle PLNE ;

— Étape 2 (*GC dynamique vs PLNE dynamique*). Nous utilisons comme référence le système de réservation basé sur le modèle $F\text{-}PLNE^{ROCSP_m}$ à chaque moment $m \in M$, ce qui veut dire que la comparaison se fait entre résultats de deux systèmes de réservation dynamique, sauf qu'à chaque moment du processus de réservation, dans l'un on utilise une résolution exacte (PLNE) et que dans l'autre une résolution heuristique (GC).

Pour rappel, le modèle $F\text{-}PLNE^{ROCSP_m}$ utilisé est un modèle de flot simplifié, avec des temps de résolution raisonnables, qui permettent d'effectuer des simulations. La simplification consiste à intégrer une contrainte de recharge électrique *forte*. Donc, l'utilisation de ce dernier modèle signifie que l'évaluation est faite dans le cas particulier de contrainte de recharge *forte*. Or, l'heuristique de CG considère une contrainte de recharge *normale*. De ce fait, pour cette partie des simulations uniquement, la GC a été adaptée pour une contrainte de recharge *forte*. L'adaptation a été réalisée en utilisant le même principe d'extension d'arcs utilisé pour réserver les voitures afin de forcer les voitures à se recharger à la fin de chaque voyage. Chaque arc de déplacement entre stations est étendu avec un nombre de périodes de temps égal à sa consommation d'autonomie électrique. Les contraintes de capacité du problème maître sont adaptées en fonction.

Dans les figures 4.5 , 4.6, 4.8 et 4.9, chaque valeur des graphiques fournit un gap qui est l'écart résultant, en pourcentage, entre les deux méthodes comparées. Chaque ligne verticale représente les résultats pour une ville où la valeur

du milieu est la moyenne sur 10 instances du *ROCSP*, tandis que les valeurs supérieures et inférieures de chaque ligne verticale représentent respectivement les valeurs maximales et minimales obtenues sur ces 10 instances.

Étape 1 : comparaison avec le modèle PLNE appliqué au problème global statique. Dans cette étape, nous comparons le système de réservation dynamique basé sur la GC avec un modèle statique. Bien évidemment, le modèle statique nous donne de meilleurs résultats, parce que toutes les demandes sont supposées connues à l'avance et que cette information n'est pas disponible dans le cas dynamique. Mais il est intéressant de les comparer, car résoudre de manière optimale le problème statique nous permet de calculer une borne inférieure pour le problème dynamique. Sans oublier de mentionner qu'aucun algorithme ne peut garantir la réalisation de cette limite inférieure résolvant le problème dynamique, en raison du manque d'informations sur les futures demandes.

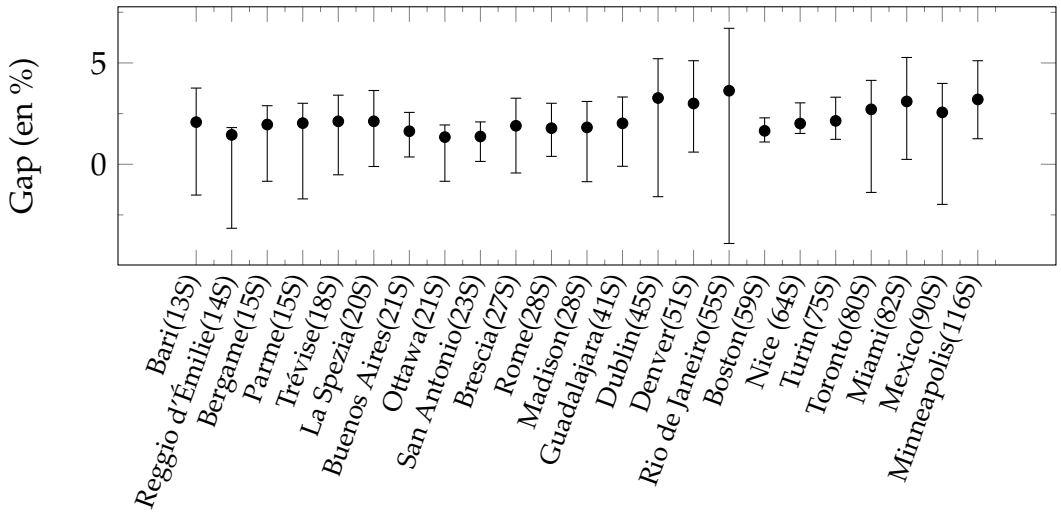


FIGURE 4.5 – *Gap d'écart relatif (en%) dans le nombre de requêtes acceptées entre le système de réservation basé sur GC et le modèle PLNE statique (GC dynamique vs PLNE statique)*

Nous pouvons observer dans la figure 4.5 que pour la plupart des villes, le gap en nombre de requêtes acceptées moyen est entre 2 % et 4 % et le gap maximal sur toutes les villes est de 11 %. Plus précisément, le gap en termes de coût (fonction objectif) n'est pas donné car il est très proche de celui du nombre de requêtes acceptées, ce qui est normale vu que la priorité est donnée à la satisfaction des clients dans la fonction objectif. Donc pour résumer, même si nous comparons notre modèle dynamique à un modèle statique, il semble que les écarts dans le nombre de requêtes acceptées soient relativement acceptables. Mais d'un autre côté, on remarque qu'en moyenne, le système dynamique gé-

nère environ trois fois plus d'opérations de relocalisation (Figure 4.6). Cela peut s'expliquer par le fait que le modèle statique utilise les informations supplémentaires sur les futures demandes pour sélectionner un ensemble de demandes à accepter qui ne nécessitent qu'un petit nombre d'opérations de relocalisation.

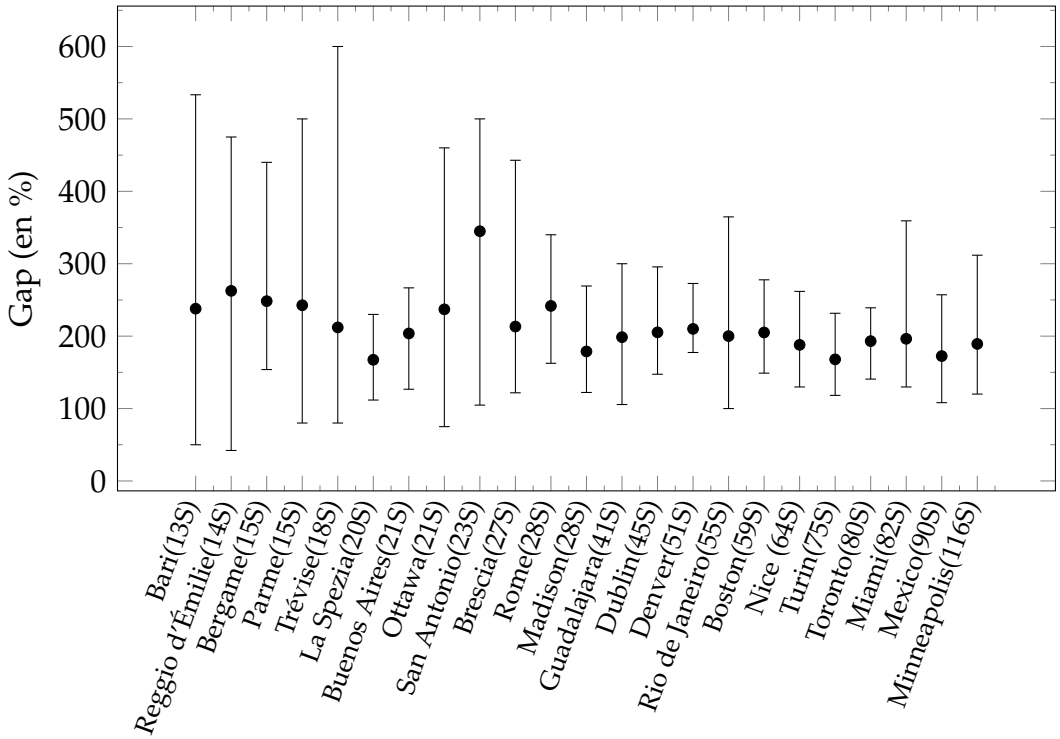


FIGURE 4.6 – *Gap d'écart relatif (en%) dans le nombre d'opérations de relocalisation générées entre le système de réservation basé sur GC et le modèle PLNE statique (GC dynamique vs PLNE statique)*

Étape 2 : comparaison avec le modèle PLNE appliqué à chaque moment du processus de réservation. Dans cette dernière étape, nous comparons les résultats du système de réservation basé sur l'heuristique de GC avec le système de réservation basé sur le modèle PLNE exact appliqué à chaque moment $m \in M$, dans le but de savoir de «combien» est dégradée la qualité de la solution par le fait d'utiliser l'heuristique de GC dans le système de réservation, au lieu d'une méthode exacte. Toutefois, il est impossible dans la pratique d'utiliser le modèle PLNE exact dans le système de réservation dynamique, en particulier pour les grandes instances, où le délai de réponse est beaucoup trop long, comme l'indique la figure 4.7. On peut voir que le temps de réponse moyen est de l'ordre de 5 min pour l'instance de Dublin avec 45 stations et atteint 20 min pour l'instance de Minneapolis avec 116 stations. Sans parler du fait que l'heuristique de GC est capable de prendre en compte une contrainte de recharge

normale, contrairement au modèle $F\text{-}PLNE^{ROCSP}$. *In fine*, une comparaison avec le modèle $R\text{-}PLNE^{ROCSP}$ aurait été plus appropriée, mais hélas impossible dans le cadre dynamique, au vu des temps d'exécution de ce dernier.

Précisons que le gap dans les figures 4.8 et 4.9 représente un gap global calculé après la fin de tout le processus de réservation (l'arrivée de toutes les requêtes). Nous aurions pu calculer les gaps à chaque moment du processus de réservation, mais il n'est pas nécessaire de le faire, puisque nous sommes intéressés par la qualité de la solution globale sur toute la journée et non celle des solutions partielles.

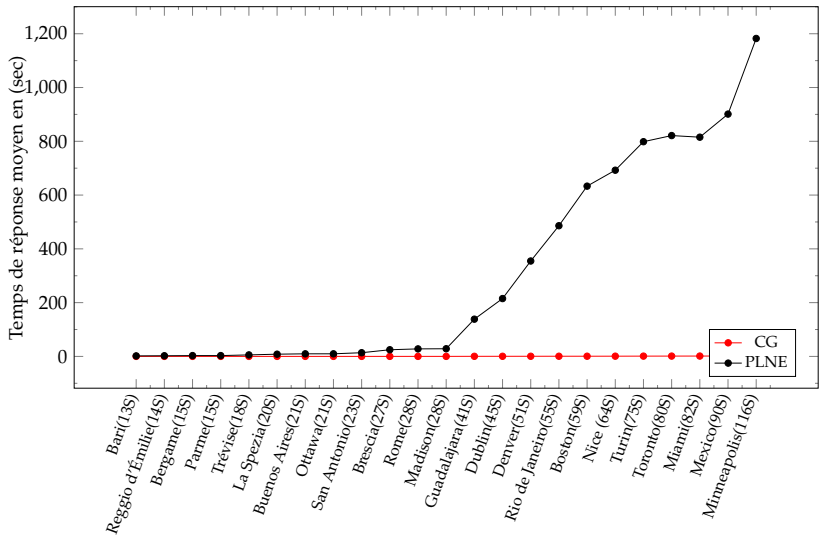


FIGURE 4.7 – *Comparaison entre le système de réservation basé sur le PLNE et celui basé sur la GC en termes de temps de réponse aux requêtes (GC dynamique vs PLNE dynamique)*

La Figure 4.8 montre les écarts dans le nombre de requêtes acceptées entre la solution du système de réservation basé sur l'heuristique de GC et celui basé sur le modèle PLNE exact. Ces écarts sont en moyenne nuls et ne dépassent pas 2,5 %. Dans certains cas, on peut même étonnamment obtenir un écart négatif, c'est-à-dire que le système de réservation basé sur une heuristique a donné de meilleures solutions globales que celui basé sur un modèle exact. Cela est dû au fait que résoudre exactement le problème $ROCSRP_m$ à chaque $m \in M$ pour répondre à chaque requête ne garantit pas la solution optimale globale. Par exemple, la solution optimale au moment m peut accepter une requête difficile à satisfaire qui engendre beaucoup de contraintes, alors que le système basé sur une heuristique va refuser cette requête, lui évitant de saturer et lui permettant de satisfaire un plus grand nombre de futures requêtes.

Dans la figure 4.9, on peut observer que le gap dans le nombre d'opérations

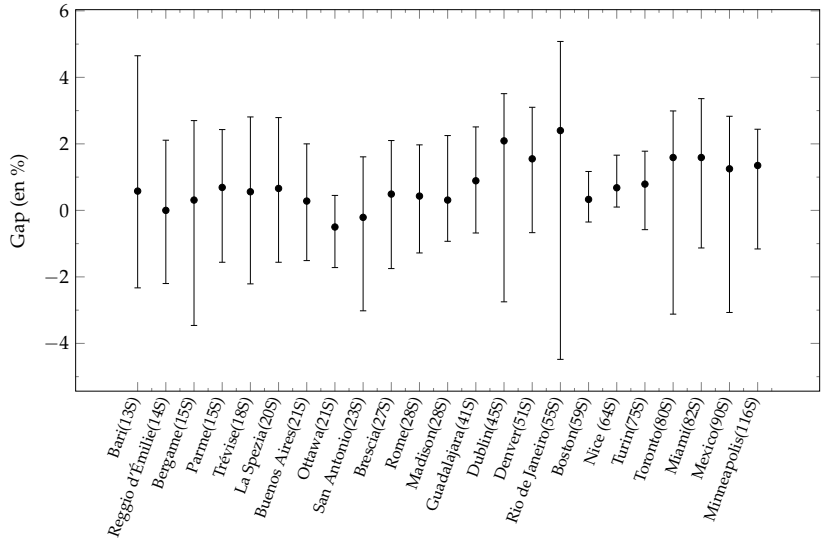


FIGURE 4.8 – *Gap d'écart relatif (en%) du nombre de requêtes acceptées entre le système de réservation basé sur la GC et celui basé sur le PLNE (GC dynamique vs PLNE dynamique)*

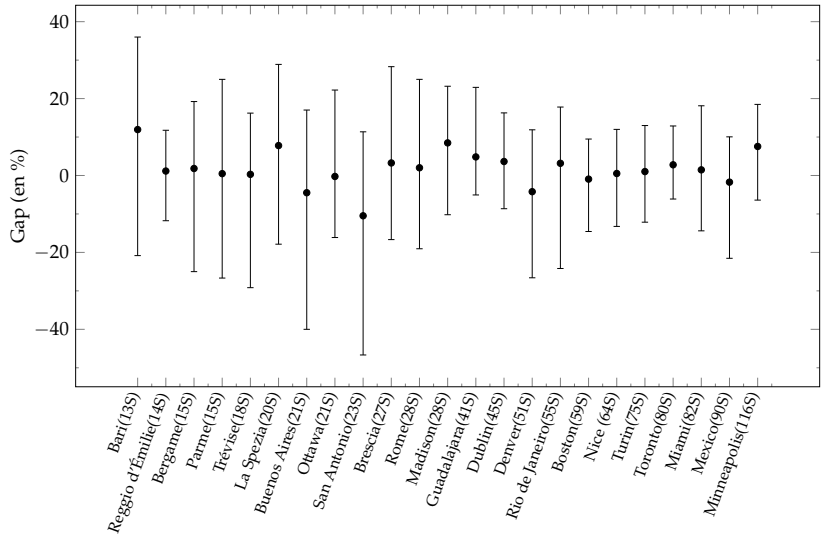


FIGURE 4.9 – *Gap d'écart relatif (en%) du nombre d'opérations de relocalisation entre le système de réservation basé sur la GC et celui basé sur le PLNE (GC dynamique vs PLNE dynamique)*

de relocalisation entre le système de réservation basé sur la GC et le modèle PLNE est égal en moyenne à 5%. Ce dernier gap est beaucoup moins important que le gap observé précédemment dans la figure 4.6, égal en moyenne à 200%, entre le système de réservation basé sur la GC et le modèle PLNE statique appliqué au problème global. Cela prouve que ce gap très important n'était pas dû

au fait d'utiliser l'heuristique de CG, mais plutôt à la perte d'informations due au passage au problème dynamique. En effet, même quand on utilise une résolution exacte dans le système de réservation dynamique, on obtient un nombre important d'opérations de relocalisation, car on accepte les requêtes automatiquement et on ne sélectionne pas celles qui génèrent le moins de relocalisation.

Globalement, on peut en déduire que le système de réservation dynamique avec une approche exacte à chaque moment $m \in M$ ne donne pas forcément de meilleurs résultats que l'utilisation de l'heuristique de GC dans le système. En moyenne, la différence de qualité des solutions n'est pas importante, alors que le modèle basé sur l'heuristique de GC prend beaucoup moins de temps pour répondre à chaque requête et permet de prendre en compte la contrainte de recharge *normale* des véhicules.

4.3 l'Auto-CoPartage

Dans cette section, nous commençons par présenter un nouveau concept, l'Auto-CoPartage, ainsi que les objectifs et avantages d'un tel système, simulé via une adaptation du système de réservation présenté dans la section précédente. Le concept d'Auto-CoPartage a été proposé par l'équipe dans un projet ANR porté par D. Josselin et F. Zhou, projet qui n'a malheureusement pas été obtenu en 2017.

4.3.1 Description de l'Auto-CoPartage

À l'heure actuelle, les deux modes de transport urbain qui permettent de partager des voitures avec les utilisateurs comme conducteurs de véhicule, sans faire appel à un chauffeur salarié d'un opérateur de transport, sont le covoiturage et l'auto-partage.

L'Auto-CoPartage constitue un nouveau mode souple privé qui se positionne à l'interface entre les deux derniers modes de transport dans le but de palier leurs défauts et regrouper leurs avantages. En effet, l'auto-partage permet de maximiser le temps d'utilisation de la voiture, car elle passe d'un utilisateur à un autre et se trouve rarement à l'arrêt ; mais l'auto-partage ne permet pas le partage de trajets. De ce fait, le remplissage des voitures n'est pas favorisé : la voiture est louée à un seul utilisateur, même s'il existe d'autres utilisateurs avec le même trajet ou une partie du trajet en commun. Quant au covoiturage, il permet le partage de trajets et donc le remplissage des voitures, mais ne maximise pas le temps d'utilisation des véhicules. Également, le covoiturage dépend fortement du propriétaire du véhicule qui détermine comme tête de pont la forme

et l'orientation des trajets. En effet, le partage d'un trajet n'est possible que si les trajets des passagers correspondent exactement au trajet du propriétaire ou à une partie incluse.

Foncièrement, l'Auto-CoPartage correspond à une inflexion de l'auto-partage, qui généralement n'affecte qu'une réservation à un véhicule sans regroupement. L'inflexion consiste en l'augmentation de capacité de regroupement par l'ajout de passagers sur une partie du trajet, comme le ferait un transport à la demande optimisé.

Règles et contraintes

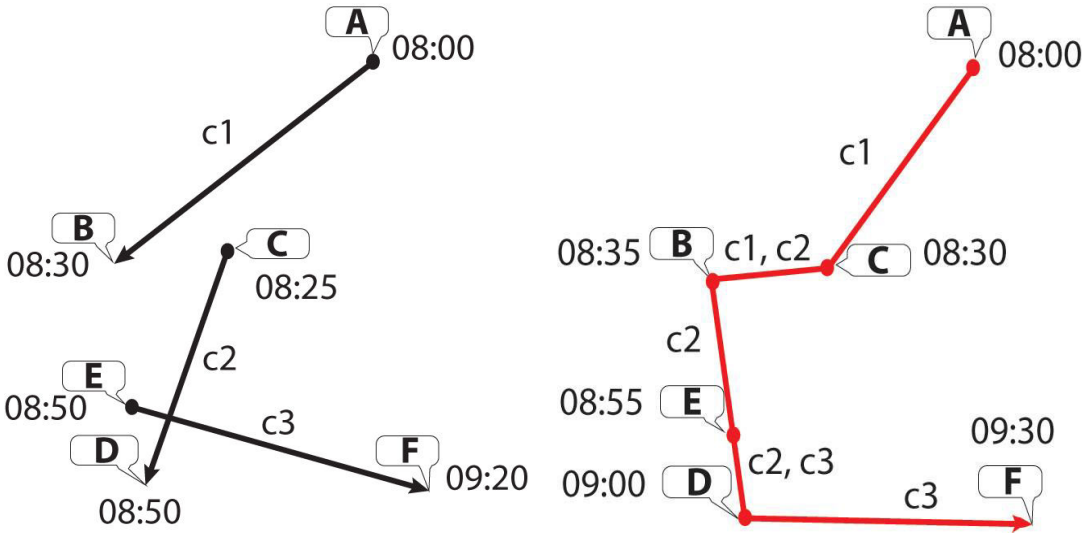
En plus des contraintes de l'auto-partage, l'Auto-CoPartage présente la possibilité de partager un trajet entre plusieurs utilisateurs avec les contraintes suivantes dans notre modèle :

- Chaque utilisateur accepte d'être conducteur ou passager du trajet partagé ou bien les deux à la fois.
- Chaque utilisateur accepte un détour limité par un intervalle de temps.
- Le nombre d'utilisateurs qui partagent un trajet ne doit pas dépasser la capacité du véhicule utilisé.
- La consommation de recharge électrique d'un trajet partagé est proportionnelle à la durée du trajet.
- Dans ce système, l'usage des fenêtres de temps de réservation n'est pas considéré, donc nous revenons à des temps de départ et d'arrivée fixes pour chaque requête ($T^d_{q_i}$ et $T^f_{q_i}$).

Exemple

Pour mieux expliquer le concept, prenons l'exemple d'un service d'Auto-CoPartage avec 3 clients : $C1$ part de A pour aller à B, $C2$ va de C à D et $C3$ de E à F. Ici, le covoiturage n'est pas possible : aucun des clients ne peut prendre une voiture et transporter les autres, car le détour engendré est trop grand ($C1$ devrait faire le trajet $A - C - E - D - F - B$, environs 4 fois plus long que son trajet initial $A - B$). Avec l'auto-partage, une voiture est affectée à chaque client. L'alternative que représente le transport à la demande, est qu'il peut regrouper les 3 clients, mais cela nécessite un agent (chauffeur) supplémentaire partant et revenant à un dépôt (davantage de distances parcourues à cause du «haut-le-pied»). Avec l'Auto-CoPartage (cf. figure 4.10), les 3 clients partagent une seule voiture sans agent et supportent des détours plus au moins acceptables. $C1$ conduit la voiture et fait le trajet $A - C - B$ (5 min de détour par rapport à

son trajet initial) ; $C2$ prend le volant à B et fait le trajet $C - B - E - D$ (10 min de détour : 5 min détour + 5 min attente du client $C1$) ; $C3$ est devenu le chauffeur à D et fait le trajet $E - D - F$ (10 min de détour).



(a) requêtes (b) fusion des requêtes

FIGURE 4.10 – *Exemple d'Auto-CoPartage avec fusion de 3 requêtes clients et les horaires correspondants*

Avantages et objectifs

Le concept d'Auto-CoPartage présente par ailleurs plusieurs avantages selon différents points de vue :

— Opérateur : maximisation des recettes issues des abonnements, de la satisfaction des clients et du temps journalier d'utilisation des véhicules partagés ; minimisation des distances parcourues relatives à un ensemble d'origines-destinations des clients et de la fréquence des recharges électriques et des temps de rechargement ;

— Client : minimisation de la dépense, soit en tant que conducteur (remboursement partiel dû au transport à la demande d'autres clients), en s'autorisant quelques détours dans une certaine fenêtre de temps, soit en tant que passager, car une partie du coût est supportée par le conducteur (partage des dépenses grâce au regroupement).

4.3.2 Adaptation du système de réservation dynamique à l'Auto-CoPartage

Pour simuler l'Auto-CoPartage, nous adaptons le système de réservation dynamique basé sur l'heuristique de GC présenté en première partie de ce chapitre pour l'auto-partage. Pour ce faire, nous définissons un modèle PLNE modélisant le problème de fusion de requêtes et nous modifions la génération de colonnes sur laquelle se base le système de réservation, afin de prendre en compte les requêtes fusionnées.

Problème de fusion de requêtes

La fusion de requêtes est le fait de regrouper en une seule requête un ensemble de requêtes de clients qui peuvent adapter leur trajet, pour que la fusion soit possible. Plus formellement, soit q_F la requête résultante de la fusion de l'ensemble des requêtes $F \subset Q$; par abus de langage les requêtes q_F sont appelées requêtes *fusion vs* requêtes *simple* pour les $q_i \in Q$. À chaque q_F est associé un trajet avec un ensemble d'étapes $E = \{e_k : (\exists q_i \in F : (S^d_{q_i} = S_{e_k} \vee S^f_{q_i} = S_{e_k}))\}$, avec S_{e_k} la station de l'étape e_k. Nous supposons que dans le trajet d'une requête *fusion*, il est interdit de revisiter la même station. Donc si e_l et e_k sont deux étapes différentes, alors $S_{e_l} \neq S_{e_k}$. Cependant, il est possible qu'une seule étape corresponde à plusieurs requêtes en même temps, ce qui veut dire prendre ou déposer plus d'un client à la même station. Le problème de fusion de requêtes est un problème de type *DARP* avec la contrainte supplémentaire de chevauchement de clients, car on ne dispose pas de chauffeur qui puisse relier deux points d'un trajet entre lesquels il n'y a aucun client. Le problème de fusion défini, tout comme le *DARP*, peut être vu comme un problème d'ordonnancement (cf. Gondran et al. (2018)), où il s'agit d'ordonner l'ensemble des étapes. Il est donné par le modèle $PLNE^{fusion}$ et se base sur 3 vecteurs de variables de décision binaires $x_{e_i e_k}$, ainsi que les variables continues t_{e_i} et t'_{e_i}. Nous définissons aussi B comme une constante représentant un grand nombre entier.

— $x_{e_i e_k} \in \{0, 1\}, e_i \neq e_k$: égal à 1 si l'étape e_k est après l'étape e_i dans le trajet de la fusion, 0 sinon.

— $t_{e_i} \in [0, T^{max}]$: le temps de départ de la station S_{e_i} vers la station de la prochaine étape dans le trajet de la fusion.

— $t'_{q_i} \in [0, T^{max}]$: le détour enregistré chez le client de la requête q_i, après la fusion par rapport à son temps d'arrivée initial.

$$\min \quad \sum_{q_i \in F} t'_{q_i} \qquad (PLNE^{fusion})$$

sous contraintes de (4.10) à (4.15).

$$t_{e_i} + T_{S_{e_i} S_{e_k}} \leq t_{e_k} + (1 - x_{e_i e_k}) \cdot B \qquad \forall e_i, e_k \in E \tag{4.10}$$

$$t_{e_k} \geq T^d_{q_i} \qquad \forall q_i \in F, \forall e_k \in E : S^d_{q_i} = S_{e_k} \tag{4.11}$$

$$t'_{q_j} \geq (t_{e_i} + T_{S_{e_i} S_{e_k}}) - T^f_{q_j} + (x_{e_i e_k} - 1) \cdot B \qquad \forall e_i, e_k \in E, \forall q_j \in F : S^f_{q_j} = e_k \tag{4.12}$$

$$t'_{q_j} \leq detour^{max} \qquad \forall q_j \in F \tag{4.13}$$

$$x_{e_k e_l} = 1 \qquad \forall q_i \in F, \forall (e_k, e_l) \in E : S_{e_k} = S^d_{q_i}, S_{e_l} = S^f_{q_i} \tag{4.14}$$

$$\sum_{(e_k, e_n, e_l) \in E : S_{e_k} = S^d_{q_i}, S_{e_l} = S^f_{q_i}} (x_{e_k e_n} + x_{e_n e_l} - 1)$$
$$+ \sum_{q_j \in F} \sum_{(e_n, e_k, e_l, e_m) \in E : S_{e_n} = S^d_{q_j}, S_{e_k} = S^d_{q_i}, S_{e_l} = S^f_{q_i}, S_{e_m} = S^f_{q_j}} (x_{e_n e_k} + x_{e_l e_m} - 1)$$
$$+ \sum_{q_j \in F : S^d_{q_j} = S^d_{q_i}} 1 + \sum_{q_j \in F : S^f_{q_j} = S^f_{q_i}} 1 \geq 1 \qquad \forall q_i \in F \tag{4.15}$$

La contrainte (4.10) permet de garantir que le temps de départ de la station de chaque étape est supérieur ou égal au temps de départ de toute station qui représente une étape précédente, en plus du temps de trajet qui sépare les deux stations. La contrainte (4.11) permet de garantir que le temps de départ de chaque étape est supérieur ou égal au temps de départ de toutes les requêtes correspondantes à cette étape. La contrainte (4.12) permet de calculer le détour pour chaque client par rapport à son temps d'arrivée initial. La contrainte (4.14) permet de garantir que l'étape correspondante au départ de chaque requête vient après l'étape correspondante à son arrivée. La contrainte (4.15) permet d'assurer que la fusion soit faisable dans le sens où il n'est pas possible de diviser la requête *fusion* en deux. Pour que ce soit le cas, il faut que les trajets des clients se chevauchent de façon à ce que chaque requête vérifie :

— soit il existe au moins une étape (e_n) qui sépare les deux étapes correspondantes aux départs et arrivées de la requête : $\overbrace{e_k \rightarrow e_n \rightarrow e_l}^{q_i}$ ce qui donne $(x_{e_k e_n} + x_{e_n e_l} - 1)$ égal à 1, 0 sinon, car au moins une des deux variables utilisées est égal à 1.

— soit le trajet de la requête est strictement inclus dans le trajet d'une autre requête : $\underbrace{e_n \rightarrow \overbrace{e_k \rightarrow e_l}^{q_i} \rightarrow e_m}_{q_j}$ ce qui donne $(x_{e_n e_k} + x_{e_l e_m} - 1)$ égal à 1. Cette dernière expression peut toutefois être égale à -1, dans le cas où le trajet de q_j est strictement inclus dans celui de q_i. Mais cela ne pose pas de problème car dans ce cas : la première somme de la contrainte sur q_i sera égale à 2, vu que deux étapes (celles de q_j) se trouvent au milieu de son trajet et cela permet d'annuler le -1 de la deuxième somme.
— soit le trajet de la requête est inclus dans le trajet d'une autre requête mais les deux requêtes partagent la station de départ ou d'arrivée, ou les deux en même temps.

Le modèle ($PLNE^{fusion}$) est utilisé à chaque moment $m \in M$ pour savoir si les requêtes d'un ensemble donné $F \subset Q_m$ peuvent être fusionnées. Dans le cas où la fusion est possible, le modèle retourne la fusion qui minimise la somme des détours des clients et qui est associée à une requête q_F telle que $T^d_{q_F}$ et $S^d_{q_F}$ correspondent à la station et au temps de départ de la première étape du trajet de la fusion et $T^f_{q_F}$ et $S^f_{q_F}$ correspondent à la station et au temps de départ de la dernière étape du trajet. Dans le système de réservation, seulement les ensembles $F \in Q_m$ tels que $|F| \leq 4$ sont considérés, ce qui correspond à la capacité des voitures et rend le problème relativement facile à résoudre.

En effet, nous considérons la possibilité d'insérer chaque nouvelle requête dans le trajet de chaque requête *fusion* ou *simple* déjà existantes. Ce processus d'insertion n'impacte pas vraiment les temps de réponse du système de réservation, car l'acceptation d'une nouvelle requête ne dépend pas vraiment du calcul des fusions possibles. De plus, celle-ci peut être acceptée en tant que requête *simple* et en même temps on peut continuer à rechercher une fusion faisable avec elle. Plus formellement, soit Q^f_m l'ensemble des requêtes faisables qui inclut aussi les requêtes simples. En effet, pour chaque requête simple $q_i \in Q$, on peut considérer la requête fusion correspondante $q_F : F = \{q_i\}$, ce qui fait que Q^f_m est initialisé à chaque moment $m \geq 1$ à l'ensemble Q_m union Q^f_{m-1} avec $Q^f_0 = \emptyset$. Puis, pour chaque requête q_i qui arrive dans le système au moment m, nous générons tous les ensembles $F \subset Q_m$ tels que : $F = \{q_i\} \cup \{q_{F'} : F' \in Q^f_m, |F'| \leq 3\}$.

Problème maître

Pour que le modèle puisse prendre en compte les requêtes fusionnées, nous redéfinissons l'ensemble U_4 dans $G^{ROCSP}(V, U, R)$, ce qui donne $U_4 = \{(v^{T^d_{q_F}}_{S^f_{q_F}}, v^{T^f_{q_F}}_{S^f_{q_F}}) :$

$q_F \in Q_m^f\}$. À chaque requête fusion q_F est associée la constante $\beta_{q_F}^j$ tel que :

— $\beta_{q_F}^j$: égale à 1 si le chemin numéro $j \in P$ utilise l'arc $(v_{S_{q_i}^d}^{T_{q_i}^d}, v_{S_{q_i}^f}^{T_{q_i}^f}) \in U_4$, 0 sinon.

Le problème maître est donné par le modèle $PM^{CoPartage}$, qui est une adaptation du modèle PM^{ROCSP_m}, en ajoutant la constante $\beta_{q_F}^j$.

$$\min Ct = \beta \cdot C_R - \gamma \cdot C_C \qquad (PM^{CoPartage})$$

avec :

— $C_R = \sum_{j \in P} \sum_{(v_{s_i}^t, v_{s_k}^{t'}) \in U_3} T_{s_i s_k} \cdot \alpha^j_{(v_{s_i}^t, v_{s_k}^{t'})} \cdot x_j$

— $C_C = \sum_{j \in P} \sum_{q_F \in Q_m^f} \beta_{q_F}^j \cdot G_{q_F} \cdot x_j$

sous les contraintes de (4.16) à (4.19).

$$\sum_{j \in P} \alpha^j_{(v_{s_i}^t, v_{s_i}^{t+1})} \cdot x_j \leq K_{s_i} \qquad \forall (v_{s_i}^t, v_{s_i}^{t+1}) \in U_2 \qquad (4.16)$$

$$\sum_{j \in P} \sum_{q_F : q_i \in F} \beta_{q_F}^j \cdot x_j = 1 \qquad \forall q_i \in Q'_m \qquad (4.17)$$

$$\sum_{j \in P} \sum_{q_F : q_i \in F} \beta_{q_F}^j \cdot x_j \leq 1 \qquad \forall q_i \in Q''_m \qquad (4.18)$$

$$\sum_{j \in P} x_j = W^{max} \qquad (4.19)$$

L'équation (4.16) assure la contrainte de capacité. Dans l'adaptation du problème maître, les équations (4.17) et (4.18) permettent d'assurer, qu'au plus une requête *fusion* est satisfaite parmi celles qui partagent au moins une requête *simple*. Enfin, l'équation (4.19) permet de garantir que le nombre de colonnes utilisées est égal au nombre de voitures.

Problème du *Pricing*

Rappelons que le problème du *Pricing* est le problème de plus court chemin avec contraintes de ressource entre V^d et V^f où la consommation électrique

$T^r_{q_F}$ de chaque requête *fusion* est fixée proportionnellement à la durée totale de son trajet, ainsi que le gain G_{q_F}. Le coût réduit d'une colonne $C\bar{R}_j$ est recalculé en utilisant les nouveaux vecteurs de variables doubles $\pi_{(v^t_{s_i}, v^{t+1}_{s_i})}$, μ'_{q_i} , μ''_{q_i} et κ correspondant respectivement aux contraintes (4.16),(4.17), (4.18) et (4.19).

$$\begin{aligned} C\bar{R}_j =& \beta \cdot \Big(\sum_{(v^t_{s_i}, v^{t'}_{s_k}) \in U_3} T_{s_i s_k} \cdot \alpha^j_{(v^t_{s_i}, v^{t'}_{s_k})} \Big) - \gamma \cdot \Big(\sum_{q_F \in Q^f_m} \beta^j_{q_F} \cdot G_{q_F} \Big) \\ & - \sum_{(v^t_{s_i}, v^{t+1}_{s_i}) \in U_2} \alpha^j_{(v^t_{s_i}, v^{t+1}_{s_i})} \cdot \pi_{(v^t_{s_i}, v^{t+1}_{s_i})} \\ & - \sum_{q_i \in Q'_m} \sum_{q_F : q_i \in F} \beta^j_{q_F} \cdot \mu'_{q_i} - \sum_{q_i \in Q''_m} \sum_{q_F : q_i \in F} \beta^j_{q_F} \cdot \mu''_{q_i} - \kappa \end{aligned}$$

Pour obtenir un coût par arc, $C\bar{R}_j$ est reformulé de la sorte :

$$\begin{aligned} C\bar{R}_j =& -\kappa + \sum_{(v^t_{s_i}, v^{t'}_{s_k}) \in U_3} \alpha^j_{(v^t_{s_i}, v^{t'}_{s_k})} \cdot (\beta \cdot T_{s_i s_k}) \\ & + \sum_{(v^t_{s_i}, v^{t+1}_{s_i}) \in U_2} \alpha^j_{(v^t_{s_i}, v^{t+1}_{s_i})} \cdot (-\pi_{(v^t_{s_i}, v^{t+1}_{s_i})}) \\ & + \sum_{q_F \in Q^f_m} \beta^j_{q_F} \cdot \left(- \sum_{q_i \in Q'_m : q_i \in F} \mu'_{q_i} - \sum_{q_i \in Q''_m : q_i \in F} \mu''_{q_i} - \gamma \cdot G_{q_F} \right) \end{aligned}$$

4.3.3 Simulations

Pour évaluer les apports de l'Auto-CoPartage, nous réalisons des simulations sur les instances de Nice et calculons à chaque fois la moyenne des résultats sur 10 instances ainsi qu'un intervalle de confiance. Une comparaison entre 3 scénarios est réalisée :

- scénario 1 : correspond à l'Auto-CoPartage où aucun détour n'est autorisé ($detour^{max} = 0$ min).
- scénario 2 : correspond à l'Auto-CoPartage où on autorise un détour de 15 min ($detour^{max} = 15$ min).
- scénario 3 : correspond à l'auto-partage classique et sert de référence.

Dans la figure 4.11, on peut observer que l'Auto-CoPartage (scénarios 1 et 2) augmente le taux de satisfaction à partir des instances de plus de 600 requêtes

par rapport à l'auto-partage classique (scénario 3), atteignant une augmentation d'environ 4% pour le scénario 1 sur l'instance avec 1000 requêtes. Quant au scénario 2, il permet une meilleure amélioration du taux de satisfaction, qui atteint environ 10 % sur l'instance avec 1000 requêtes. L'augmentation du taux de satisfaction s'explique par le fait que plus le nombre de requêtes est grand, plus on a de chance de trouver des requêtes dont la fusion est faisable. Aussi le détour autorisé augmente le nombre de fusions possibles.

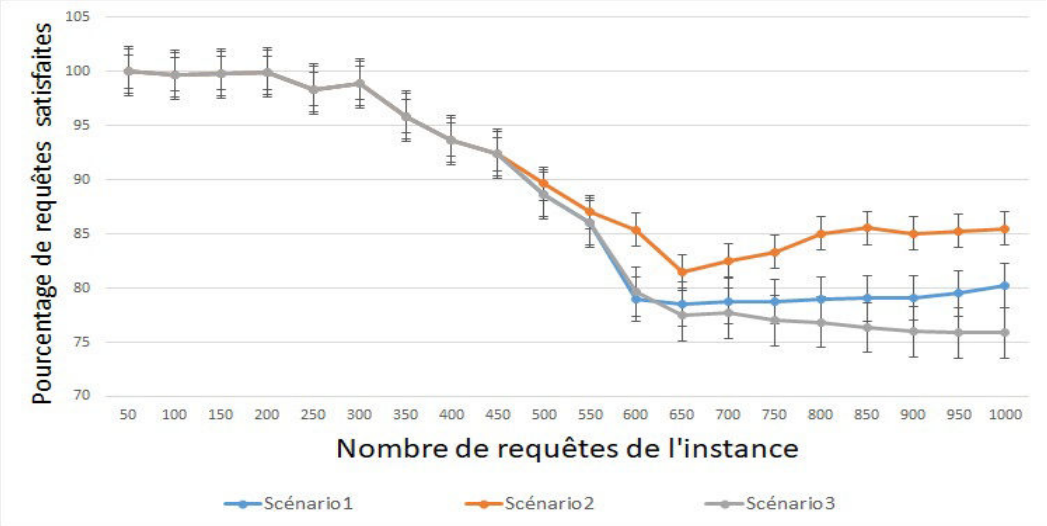


FIGURE 4.11 – *Pourcentage de satisfaction des requêtes avec différents scénarios pour les 3 scénarios :* 1 *(detour*$^{max} = 0$ *min),* 2 *(detour*$^{max} = 15$ *min) et* 3 *(auto-partage classique sans fusion)*

Toutefois l'intérêt de l'Auto-CoPartage n'est pas uniquement l'augmentation du taux de satisfaction, mais aussi l'augmentation du remplissage des voitures, critère que l'on peut observer dans les figures 4.12 et 4.13 qui représentent les scénarios 1 et 2 respectivement. Pour le scénario 2 ($detour^{max} = 15min$), on peut observer par exemple que le pourcentage de requêtes fusionnées peut atteindre 20% de l'ensemble des requêtes satisfaites sur l'instance de 1000 requêtes, alors que le taux de satisfaction n'a augmenté que de 10 % (cf. figure 4.11). Cela s'explique par le fait que la fusion des requêtes est réalisée, même si elle n'est pas nécessaire à la satisfaction des requêtes concernées et qu'il reste assez de voitures disponibles pour satisfaire chaque requête par une voiture dédiée, le cas échéant.

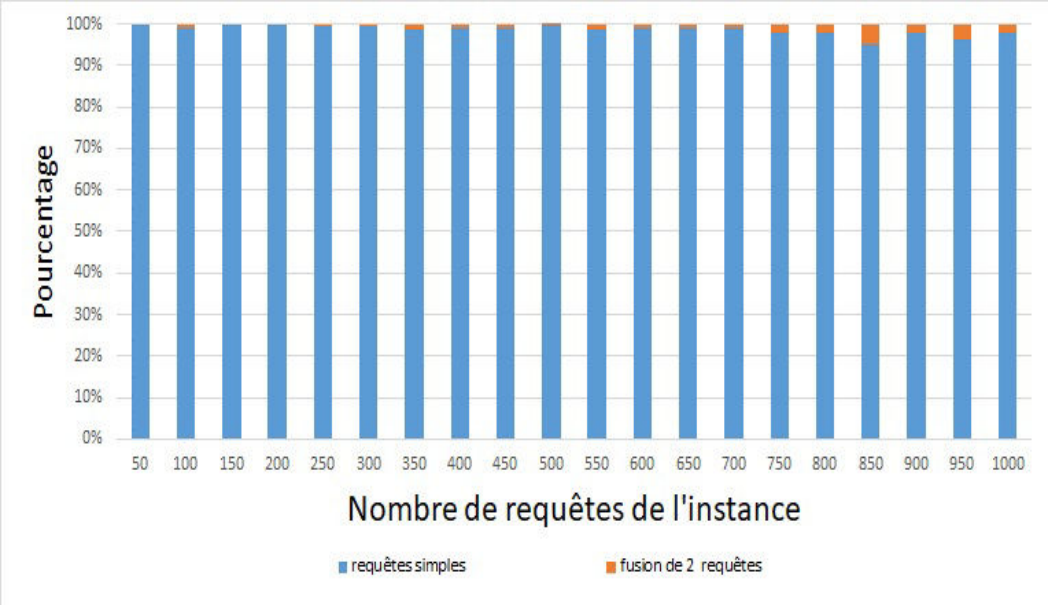


FIGURE 4.12 – *Pourcentage de requêtes fusionnées pour le scénario* 1 *(detour^{max} = 0 min)*

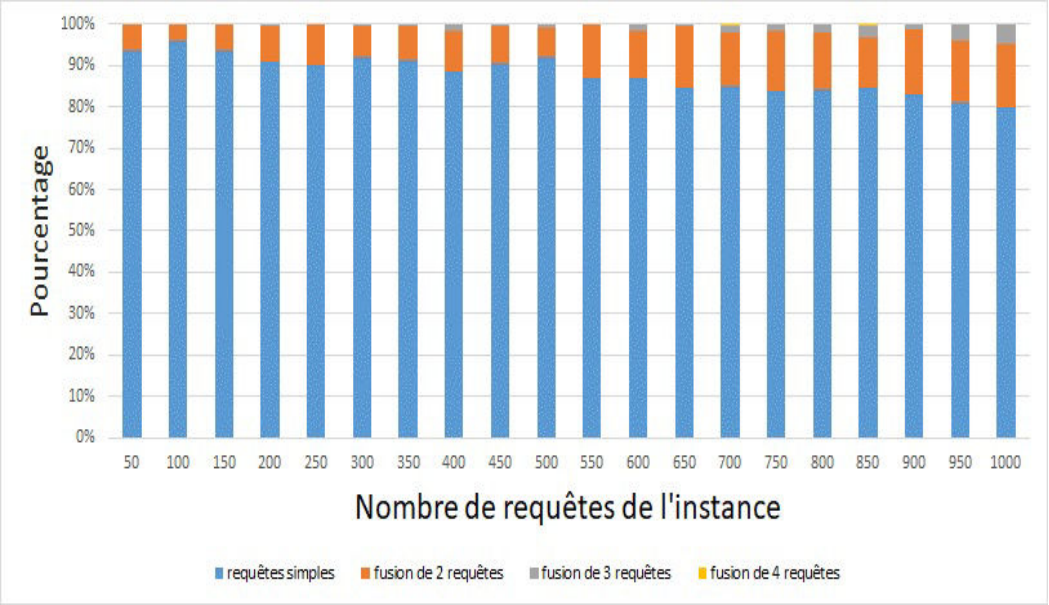


FIGURE 4.13 – *Pourcentage de requêtes fusionnées pour le scénario* 2 *(detour^{max} = 15 min)*

4.4 Conclusion

Dans ce chapitre, nous avons proposé un système de réservation en temps réel pour le service de partage de voitures électriques. Ce système est capable de répondre rapidement à la demande d'un client, tout en satisfaisant un nombre maximum de requêtes et prend en compte l'optimisation des ressources. Nous avons adapté le modèle PLNE pour la résolution exacte du *ROCSP* en temps réel. Mais comme notre système doit fournir des réponses rapides, chose qui ne peut pas être réalisée en utilisant le modèle PLNE en raison de son grand temps d'exécution, nous avons opté pour une génération de colonnes, via une heuristique qui donne presque les mêmes résultats que le modèle PLNE appliqué au problème de réservation dynamique, mais avec des temps de réponses beaucoup plus courts et adaptés à un système de réservation en temps réel.

Dans la dernière partie du chapitre, nous avons proposé un nouveau mode de partage de voitures à savoir l'Auto-CoPartage, qui permet de regrouper un ensemble de clients dans une voiture. Des simulations de ce nouveau mode ont été menées sur les instances de Nice à l'aide d'un modèle PLNE qui permet de fusionner les requêtes et en adaptant les systèmes de réservation que nous avons proposés pour l'auto-partage classique. Les résultats ont montré que quand la demande est grande, l'Auto-CoPartage permet d'augmenter le taux de satisfaction jusqu'à 4 % dans une ville comme Nice, lorsqu'on ne tolère pas de détour et 10 % lorsqu'on autorise un détour de 15 min pour chaque client. En plus du taux de satisfaction, on peut observer que le remplissage des voitures augmente sensiblement, notamment quand le nombre de requêtes augmente.

Conclusion générale

Dans cette thèse, nous nous somme intéressés à l'auto-partage qui représente aujourd'hui une alternative sérieuse à la voiture privée. Notre intérêt s'est focalisé sur l'optimisation de la gestion de l'auto-partage *à un seul sens* qui engendre des problèmes d'optimisation liés à la relocalisation des voitures. Ces problèmes sont de nature combinatoire et de ce fait, représentent des sujets privilégiés pour la communauté de recherche opérationnelle. En effet, nous avons proposé des programmes et des méthodes approchées qui peuvent être potentiellement mis en œuvre dans les collectivités, car ils fournissent de bons résultats dans des temps raisonnables. Dans les années à venir, nous pensons que d'autres travaux seront effectués pour tenter de résoudre, autant que faire se peut, ces nouvelles problématiques de transport à forte dimension combinatoire.

Le problème d'équilibrage des voitures dans l'auto-partage a été décomposé en deux sous-problèmes, à savoir le *ROCSP* et l'*ESRP*, les deux étant des problèmes de routage. Dans le premier sont gérés les réservations des clients et le calcul des opérations de relocalisation nécessaires. Quant au second, son objectif est d'optimiser l'affectation des opérations de relocalisation résultant du *ROCSP* aux agents. Comme montré dans le chapitre 3 via une série de tests expérimentaux, la relocalisation des voitures joue beaucoup dans la performance du système. L'importance de la relocalisation fait que l'utilisation d'algorithmes efficaces d'optimisation est primordiale, avec des temps de calcul raisonnables et des solutions qui se rapprochent de l'optimalité. En ce sens, nos algorithmes semblent prometteurs. Bien entendu, ce travail ne constitue qu'une première ébauche, qui a permis de modéliser et de résoudre différentes variantes du problème de relocalisation dans l'auto-partage *à un seul sens* avec voitures électriques. De ce fait, les pistes de recherche futures concernant ce problème sont nombreuses. Nous pouvons dresser le bilan des travaux effectués et dégager ensuite les perspectives envisageables.

Bilan

Nous résumons brièvement les contributions principales de cette thèse :

1. Des modélisations PLNE ont été proposées au problème de relocalisation à travers ses deux sous-problèmes : le *ROCSP* et l'*ESRP*. La formulation PLNE proposée pour le *ROCSP* prend en compte différentes fonctions objectif, ainsi que les contraintes liées à la recharge électrique des véhicules de manière plus réaliste que ce qu'on peut trouver dans la littérature. Le problème de relocalisation des voitures est passé d'un problème de flot à un problème de routage, plus complexe. La preuve a été faite que cette variante du problème est de complexité *NP-difficile*, tout comme l'*ESRP* modélisé.
2. Un modèle de génération d'instances pour le *ROCSP* à partir de données géographiques de sites d'auto-partage issus du monde réel a été proposé. Les instances sont construites à partir d'un modèle gravitaire de flux théoriques.
3. Un algorithme génétique basé sur une heuristique constructive pour le *ROCSP* nous a permis d'obtenir des solutions qui se rapprochent de l'optimalité avec un gap moyen d'environ 5% sur des petites instances et un temps de calcul ne dépassant pas les 10 min, pendant que le modèle PLNE pouvait prendre jusqu'à 30 heures pour des instances de 10 stations.
4. Une heuristique constructive ainsi qu'un algorithme de *Branch and Price* avec un parcours d'arbre et branchement heuristique ont permis de résoudre l'*ESRP* avec des solutions optimales pour les instances de moins de 51 stations et un gap maximum de 2 % pour les instances plus grandes, ne dépassant pas les 10 min en temps de traitement.
5. Grâce aux heuristiques proposées aux *ROCSP* et l'*ESRP*, nous avons effectué des simulations pour l'auto-partage à Nice, ville comptant 64 stations géolocalisées d'auto-partage. Ces simulations nous permettent de constater que la relocalisation augmente significativement le pourcentage de satisfaction des requêtes de clients et le taux d'utilisation des voitures.
6. Après avoir proposé des approches de résolution pour la version statique du problème *ROCSP*, nous nous sommes intéressés à la variante dynamique, où les requêtes de réservation des clients ne sont pas connues à l'avance. Elles arrivent progressivement l'une après l'autre, avec une obligation d'accepter ou de refuser les requêtes dans les secondes qui suivent leur arrivée. Une génération de colonnes a été proposée comme outil dynamique pour l'acceptation des requêtes. Le problème maître de cette génération de colonnes est modifié à l'arrivée de chaque requête, mais les colonnes générées précédemment restent exploitables. On peut ainsi répondre rapidement aux requêtes, car la nouvelle solution de la génération

de colonnes est obtenue en ajoutant seulement un petit nombre de colonnes. Les temps de réponses étaient de l'ordre de 10 secondes maximum et avec un gap qui ne dépasse pas 2,5 % par rapport au PLNE.

7. Pour finir, nous avons proposé d'investiguer l'idée d'un nouveau mode d'auto-partage, l'Auto-CoPartage, concept imaginé comme une hybridation entre l'auto-partage et le covoiturage. Ce système permet de remplir les voitures d'auto-partage, en plus du partage de leurs temps d'utilisation. Pour simuler l'Auto-CoPartage, le système de réservation proposé pour l'auto-partage a été adapté en ajoutant un modèle PLNE qui permet de calculer les fusions possibles des trajets de clients dans le but de les regrouper. Les simulations ont montré une augmentation du taux de satisfaction ainsi que du remplissage des voitures, notamment dans le cas d'une montée en charge du service.

Perspectives

Même si les résultats que nous avons obtenus via nos algorithmes sont satisfaisants, ils restent toutefois perfectibles. Dans un premier temps, il est indispensable de confirmer nos résultats sur une plus grande palette de sites d'auto-partage et avec des simulations à partir de données réelles. Concernant les perspectives méthodologiques, elles sont bien sûr nombreuses et nous ne prétendons pas être exhaustifs dans la liste qui suit :

1. Granularité optimale pour la discrétisation du temps : dans nos simulations, nous avons discrédité le temps en périodes de 15 minutes. Il serait intéressant d'étudier l'impact de la discrétisation avec des périodes plus petites et plus grandes, pour fixer la durée qui réalise le meilleur compromis entre temps de calcul et qualité des solutions. Une modélisation du problème avec temps continu peut être envisagée, même si une telle modélisation nécessite de faire des suppositions sur le nombre maximum de visites d'une station par chaque voiture, donc perdre en généralité.
2. Problèmes spatiaux : nous pensons qu'il serait pertinent de découper les grandes villes (clustering) pour améliorer la qualité de la résolution avec l'aide d'heuristiques. L'idée est de diviser la ville en petites régions intra-urbaines, où l'on pourra résoudre le problème de relocalisation de façon locale.
3. Voitures autonomes : l'avènement des voitures autonomes, qui sont aptes à rouler sans intervention d'un conducteur, peut être l'élément qui va permettre à l'auto-partage *à un seul sens* de se substituer de manière définitive à la voiture privée, vu que la relocalisation pourra se faire sans agent et que les voitures pourront se rendre aux bornes de recharge de manière

autonome. Les modèles que nous avons proposés restent utilisables dans ce cadre là, vu que nous avons séparé le problème de routage des voitures de celui des agents.

4. Adapter nos algorithmes à la variante de l'auto-partage *à un seul sens* en *Free-Float* : un travail a été entamé dans ce sens, où les stations ont été remplacées par les communes d'une ville donnée. Il s'agirait de calculer les relocalisations entre les communes en relaxant les contraintes de capacités liées aux stations. Par contre, la contrainte de recharge électrique est plus difficile à gérer, car les agents devront se charger de conduire les voitures déchargées vers les bornes de recharge, sans l'aide des clients qui ne sont pas contraints à restituer les véhicules dans certains lieux. De plus, le *Free-Float* est utilisé majoritairement en libre service, vu la difficulté de gérer des réservations en dehors des stations. En l'absence de requêtes de réservation, l'ajout de probabilités prédictives et méthodes d'apprentissage automatique aux modèles développés devient nécessaire pour pouvoir calculer les relocalisations.

Liste des illustrations

Liste des tableaux

Bibliographie

(Agatz et al., 2012) N. Agatz, A. Erera, M. Savelsbergh, et X. Wang, 2012. Optimization for dynamic ride-sharing : A review. *European Journal of Operational Research* *223*(2), 295 – 303.

(Ait-Ouahmed et al., 2018) A. Ait-Ouahmed, D. Josselin, et F. Zhou, 2018. Relocation optimization of electric cars in one-way car-sharing systems : modeling, exact solving and heuristics algorithms. *International Journal of Geographical Information Science* *32*(2), 367–398.

(Aïvodji et al., 2018) U. M. Aïvodji, K. Huguenin, M.-J. Huguet, et M.-O. Killijian, 2018. SRide : A Privacy-Preserving Ridesharing System. Dans les actes de *11th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, Proceedings of the 11th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec), Stockholm, Sweden, 40–50.

(Auto-bleue, 2017) Auto-bleue, consulté le 23 septembre 2017. Auto bleue, location de véhicules électriques en libre-service. https://www.auto-bleue.org/fr.

(Autolib, 2014) Autolib, 2014. Rapport d'activité 2014. https://www.autolibmetropole.fr/le-service-autolib/les-chiffres-en-1-clic/.

(Bard et al., 2002) J. F. Bard, G. Kontoravdis, et G. Yu, 2002. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science* *36*(2), 250–269.

(Barth et al., 2004) M. Barth, M. Todd, et L. Xue, 2004. User-Based Vehicle Relocation Techniques for Multiple-Station Shared-Use Vehicle Systems. *Transportation Research Board 83th Annual Meeting* (04).

(bcg.perspectives, 2017) bcg.perspectives, Consulté le 20 décembre 2017. What's ahead for car sharing ? https://www.bcgperspectives.com/content/articles.

(Berbeglia et al., 2007) G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, et G. Laporte, 2007. Static pickup and delivery problems : aclassification scheme and survey. *TOP 15*(1), 1–31.

(Boyacı et al., 2015) B. Boyacı, N. Geroliminis, et K. Zografos, 2015. An optimization framework for the development of efficient one-way car-sharing systems. *European Journal of Operational Research 240*, 718–733.

(Camagny, 1996) R. Camagny, 1996. *Principes et modèles de l'économie urbaine*. Economica, Paris.

(Canaguier et Ag, 2012) B. Canaguier et P. E. I. Ag, 2012. *Élaboration Selon Les Principes Des Acv Des Bilans Énergétiques , Des Émissions De Gaz a Effet De Serre Et Des Autres Impacts Environnementaux Ind Uits Par L ' Ensemble Des Filieres De Vehicules Electriques Et De Vehicules Thermiques , Vp De Segment B (C*.

(Ceschia et al., 2011) S. Ceschia, L. Di Gaspero, et A. Schaerf, 2011. Tabu search techniques for the heterogeneous vehicle routing problem with time windows and carrier-dependent costs. *Journal of Scheduling 14*(6), 601–615.

(Chang et Chen, 2007) Y. Chang et L. Chen, 2007. Solve the vehicle routing problem with time windows via a genetic algorithm. *Discrete and continuous dynamical systems supplement*, 240–249.

(Chen et Sun, 2015) Q. Chen et T. Sun, 2015. A model for the layout of bike stations in public bike-sharing systems. *Journal of Advanced Transportation 49*, 884– 900.

(Clarke et Wright, 1964) G. Clarke et J. W. Wright, 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research 12*(4), 568–581.

(Correia et Antunes, 2012) G. H. D. A. Correia et A. P. Antunes, 2012. Optimization approach to depot location and trip selection in one-way carsharing systems. *Transportation Research Part E : Logistics and Transportation Review 48*(1), 233–247.

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

(Deng et Lin, 2011) G.-F. Deng et W.-T. Lin, 2011. Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Systems with Applications 38*(5), 5787–5793.

(Desrosiers et al., 1986) J. Desrosiers, Y. Dumas, et F. Soumis, 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*(3-4), 301–325.

(Elodie Castex, 2007) D. J. Elodie Castex, 2007. Temporalités éclatées : La réponse des transports à la demande aux nouvelles formes de mobilité. *Espace populations sociétés* (2007/2-3), 433–447.

(Fukasawa et al., 2006) R. Fukasawa, H. Longo, J. Lysgaard, M. P. de Aragão, M. Reis, E. Uchoa, et R. F. Werneck, 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming* *106*(3), 491–511.

(Garaix et al., 2010) T. Garaix, C. Artigues, D. Feillet, et D. Josselin, 2010. Vehicle routing problems with alternative paths : An application to on-demand transportation. *European Journal of Operational Research* *204*(1), 62 – 75.

(Gaskell, 1967) T. J. Gaskell, 1967. Bases for vehicle fleet scheduling. *Journal of the Operational Research Society* *18*(3), 281–295.

(George et Xia, 2011) D. K. George et C. H. Xia, 2011. Fleet-sizing and service availability for a vehicle rental system via closed queueing networks. *European Journal of Operational Research* *211*(1), 198–207.

(Gillett et Miller, 1974) B. E. Gillett et L. R. Miller, 1974. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research* *22*(2), 340–349.

(Glover, 1989) F. Glover, 1989. Tabu search—part i. *ORSA Journal on computing* *1*(3), 190–206.

(Glover, 1990) F. Glover, 1990. Tabu search—part ii. *ORSA Journal on computing* *2*(1), 4–32.

(Gondran et al., 2018) M. Gondran, M.-J. Huguet, P. Lacomme, A. Quilliot, et N. Tchernev, 2018. A dial-a-ride evaluation for solving the job-shop with routing considerations. *Engineering Applications of Artificial Intelligence 74*, 70 – 89.

(Group, 2017) U. O. R. Group, Consulté 30 juillet 2017. Bike-sharing instances. http://www.or.unimore.it/site/home/online-resources/bike-sharing-rebalancing-problem/instances.html.

(Hoffman et Kruskal, 2010) A. J. Hoffman et J. B. Kruskal, 2010. Integral boundary points of convex polyhedra. Dans les actes de *50 Years of Integer Programming 1958-2008*, 49–76. Springer.

(Jewell et University of California, 1966) W. Jewell et B. O. R. C. University of California, 1966. *Multi-commodity Network Solutions*. Operations Research Center, University of California.

(Jorge et Correia, 2013) D. Jorge et G. Correia, 2013. Carsharing systems demand estimation and defined operations : A literature review. *European Journal of Transport and Infrastructure Research 13*(3), 201–220.

(Jorge et al., 2012) D. Jorge, G. Correia, et C. Barnhart, 2012. Testing the Validity of the MIP Approach for Locating Carsharing Stations in One-way Systems. *Procedia - Social and Behavioral Sciences 54*, 138–148.

(Jorge et al., 2014) D. Jorge, G. H. a. Correia, et C. Barnhart, 2014. With Simulated Relocation Policies in One-Way Carsharing Systems. *Transactions on Intelligent Transportation Systeùs 15*(4), 1667–1675.

(Josselin et Nicot, 2003) D. Josselin et B. Nicot, 2003. Un modèle gravitaire géoéconomique des échanges commerciaux entre les pays de l'U.E., les PECO et les PTM. *Cybergeo : European Journal of Geography*.

(Kadri et al., 2016) A. Kadri, I. Kacem, et K. Labadi, 2016. A branch-and-bound algorithm for solving the static rebalancing problem in bicycle-sharing systems. *Computers & Industrial Engineering*.

(Kaspi et al., 2014) M. Kaspi, T. Raviv, et M. Tzur, 2014. Parking reservation policies in one-way vehicle sharing systems. *Transportation Research Part B : Methodological 62*, 35–50.

(Kek et al., 2009) A. G. H. Kek, R. L. Cheu, Q. Meng, et C. H. Fung, 2009. A decision support system for vehicle relocation operations in carsharing systems. *Transportation Research Part E : Logistics and Transportation Review 45*(1), 149–158.

(Kellerer et al., 2004) H. Kellerer, U. Pferschy, et D. Pisinger, 2004. Introduction to np-completeness of knapsack problems. Dans les actes de *Knapsack problems*, 483–493. Springer.

(Kolen et al., 1987) A. W. J. Kolen, A. H. G. R. Kan, et H. W. J. M. Trienekens, 1987. Vehicle routing with time windows. *Operations Research 35*(2), 266–273.

(Labadi et al., 2015) K. Labadi, T. Benarbia, J.-p. Barbot, et S. Hamaci, 2015. Stochastic Petri Net Modeling , Simulation and Analysis of Public Bicycle Sharing Systems. *Ieee Transactions on Automation Science and Engineering 12*(4), 1380–1395.

(Lenstra et Kan, 1981) J. K. Lenstra et A. H. G. R. Kan, 1981. Complexity of vehicle routing and scheduling problems. *Networks 11*(2), 221–227.

(Letchford et al., 2007) A. N. Letchford, J. Lysgaard, et R. W. Eglese, 2007. A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Society* *58*(12), 1642–1651.

(Liu et al., 2017) M. Liu, Y. Li, F. Zheng, et F. Chu, 2017. Integration of Timetabling, Multi-type Vehicle Scheduling and User Routing in Public Transit Network considering Fuel Consumption. Dans les actes de *7th International Conference on Industrial Engineering and Systems Management (IESM 2017)*, Proc. of the 7th International Conference on Industrial Engineering and Systems Management (IESM 2017), Saarbrücken, Germany, (elec. proc.).

(Louvet et Godillon, 2013) N. Louvet et S. Godillon, 2013. *Enquête nationale sur l'autopartage*. 6T - bureau de recherche.

(Luminita et al., 2009) I. Luminita, C. T., B. J.M., T. F., et B. D., 2009. Site selection for electric cars of a car-sharing service. *World Electric Vehicle Journal* *3*(1), 1–10.

(Macqueron, 2017) G. Macqueron, Consulté le 12 décembre 2017. Futura-sciences : Parts des émissions de co2 par mode de transport en france (données : Insee 2007). http://www.futura-sciences.com/planete/questions-reponses/automobile-transport-co2-part-emissions-1017/.

(Papadimitriou, 1981) C. H. Papadimitriou, 1981. On the complexity of integer programming. *Journal of the ACM (JACM)* *28*(4), 765–768.

(Papadimitriou, 2003) C. H. Papadimitriou, 2003. *Computational complexity*. John Wiley and Sons Ltd.

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

(Qureshi et al., 2009) A. Qureshi, E. Taniguchi, et T. Yamada, 2009. An exact solution approach for vehicle routing and scheduling problems with soft time windows. *Transportation Research Part E : Logistics and Transportation Review* *45*(6), 960–977.

(Rechenberg, 1973) I. Rechenberg, 1973. Evolutionsstrategie–optimierung technisher systeme nach prinzipien der biologischen evolution.

(Renaud et Boctor, 2002) J. Renaud et F. F. Boctor, 2002. A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research* *140*(3), 618 – 628.

(Renaud et al., 1996) J. Renaud, F. F. Boctor, et G. Laporte, 1996. An improved petal heuristic for the vehicle routeing problem. *Journal of the Operational Research Society 47*(2), 329–336.

(Ryan et al., 1993) D. M. Ryan, C. Hjorring, et F. Glover, 1993. Extensions of the petal method for vehicle routeing. *The Journal of the Operational Research Society 44*(3), 289–296.

(Schuijbroek et Hampshire, 2013) J. Schuijbroek et R. Hampshire, 2013. Inventory rebalancing and vehicle routing in bike sharing systems.

(Tavakkoli-Moghaddam et al., 2006) R. Tavakkoli-Moghaddam, N. Safaei, et Y. Gholipour, 2006. A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length. *Applied Mathematics and Computation 176*(2), 445–454.

(Uesugi et al., 2007) K. Uesugi, N. Mukai, et T. Watanabe, 2007. Optimization of vehicle assignment for car sharing system. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4693 LNAI*(part 2), 1105–1111.

(Vanderbeck, 2000) F. Vanderbeck, 2000. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research 48*(1), 111–128.

(Wang et Chen, 2013) H.-F. Wang et Y.-Y. Chen, 2013. A coevolutionary algorithm for the flexible delivery and pickup problem with time windows. *International Journal of Production Economics 141*(1), 4 – 13. Meta-heuristics for manufacturing scheduling and logistics problems.

(Wren et Holliday, 1972) A. Wren et A. Holliday, 1972. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly (1970-1977) 23*(3), 333–344.

(Xu et al., 2003) H. Xu, Z.-L. Chen, S. Rajagopal, et S. Arunapuram, 2003. Solving a practical pickup and delivery problem. *Transportation science 37*(3), 347–364.

(Yantong et al., 2018) L. Yantong, F. Chu, C. Feng, C. Chu, et M. Zhou, 2018. Integrated production inventory routing planning for intelligent food logistics systems. *IEEE Transactions on Intelligent Transportation Systems* (99), 1–12.

(Yellow, 1970) P. C. Yellow, 1970. A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly (1970-1977) 21*(2), 281–283.