Système d'aide à la décision pour le dernier kilomètre : application au problème de tournées de véhicules riches

Gwénaël Rault

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE BRETAGNE SUD
ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Gwénaël Rault

Système d'aide à la décision pour le dernier kilomètre : application aux problèmes de tournées de véhicules riches

Thèse présentée et soutenue à Bordeaux, le 31 mai 2021
Unité de recherche : lab-STICC, UMR 6285
Thèse N° : 594

Composition du Jury :

Président :	Christian PRINS	Professeur, Université Technologique de Troyes
Rapporteurs :	Frédéric GARDI	Directeur associé, LocalSolver
	Michel GENDREAU	Professeur, École Polytechnique de Montréal
Examinateur	Christelle JUSSIEN-GUÉRET	Professeur, Université d'Angers
Dir. de thèse :	Marc SEVAUX	Professeur, Université Bretagne Sud
Co-dir. de thèse :	Philippe LACOMME	Maître de conférences, HDR, Université Clermont Auvergne

Remerciements

Je souhaite tout d'abord adresser mes remerciements à Marc Sevaux et Philippe Lacomme, mes directeurs de thèse pour m'avoir accompagné durant ces trois années. Vous avez su vous rendre disponible tout au long de cette thèse. Je vous remercie donc grandement pour vos conseils et tous les échanges que nous avons pu avoir.

Je remercie Frédéric Gardi et Michel Gendreau d'avoir accepté d'être rapporteurs pour ma thèse et pour l'intérêt qu'ils portent à ces travaux ainsi qu'au parcours de Mapotempo. De même, je souhaite remercier Christelle Jussien-Guéret et Christian Prins pour avoir accepté de faire partie de mon jury de thèse et pour les échanges scientifiques que nous avons pu avoir.

Je remercie également l'équipe pédagogique du Master ORO à Nantes, Xavier Gandibleux, Evgeny Gurevsky, Fabien Lehuédé et Anthony Przybylski pour m'avoir donné le goût de la recherche opérationnelle. J'ai toujours plaisir à échanger avec vous.

Je remercie Flavien Lucas pour sa collaboration durant ces trois années.

Je remercie Mehdi Jabrane, qui dès les débuts de Mapotempo, a décelé la pertinence de la Recherche Opérationnelle et m'a confié la responsabilité de mettre en place cette spécialité. Merci pour ta confiance, pour nos échanges et pour ton ambition. Tout cela a permis de mettre en place un cadre idéal pour cette thèse.

Je souhaite également adresser mes remerciements à Frédéric Rodrigo dont les compétences techniques m'impressionnent toujours. Tu as ouvert la voie et je suis heureux de pouvoir en suivre les traces.

Manon et Fabien, aujourd'hui et depuis mon premier jour à Mapotempo, j'ai plaisir à travailler et échanger avec vous. Je remercie Adeline, Halil et Mickaël pour m'avoir rejoint dans cette aventure. C'est un véritable plaisir de pouvoir partager tout cela avec vous.

Merci à l'équipe béarnaise, David, Flore, Jawad et Vincent. Je remercie tous ceux qui font partie ou qui sont passés par l'équipe bordelaise : Alain, Antoine, Baturay, Cynthia, Jean-Maxime, Julien, Nathan, Sébastien, Valentin, Yann, ainsi que tous les autres, sans oublier Alexis qui nous accompagne.

Merci également à mes amis qui me suivent et soutiennent depuis longtemps que ce soit les rambos, le CClub, les pouëzéens ou les clémentais. Je voudrais remercier en particulier :

Alexis, Etienne et Pierre-Arnaud qui sont là depuis que nous sommes tout petit.

Je remercie ma fraterie : mes grandes sœurs Laetitia et Priscilla, pour votre bienveillance, les conseils et les moments d'affection, et mon petit frère Quentin, pour la complicité qui nous lie.

Je souhaite, de tout mon cœur, remercier mon amie et épouse, Léa. Cela fait maintenant plus de dix ans que tu m'encourages et me soutiens. Il est clair que sans toi, je ne serais pas celui que je suis aujourd'hui. À tous nos bons moments et à ceux encore à venir.

Finalement, je remercie mes parents, Maryline et Pascal, sans qui rien de tout cela ne serait possible. Vous avez toujours été là pour moi, vous m'avez donné l'envie de découvrir et vous m'avez laissé la liberté d'explorer ce qui attisait ma curiosité.

Table des Matières

Résumé

Cette thèse s'intéresse à la résolution de problèmes de tournées riches dans un contexte industriel de création d'un environnement intégré de résolution destiné aux clients de la société Mapotempo. L'environnement intégré de résolution appartient à la famille des systèmes d'aide à la décision et il a été conçu dans un contexte client-serveur avec un lien direct vers le web et ses applications.

L'environnement intégré de résolution offre des services de manipulation des données à différents niveaux d'abstraction qui vont de la manipulation des données au niveau des nœuds du graphe, en passant par le calcul de plus courts chemins pour aller jusqu'aux calculs de solutions pour le VRP riche. Les services de manipulation des données sont proposés sous la forme d'une interface accessible dans un contexte client-serveur, ce qui a permis à Mapotempo de créer une web-application pour valoriser les méthodes mises en place dans l'environnement intégré de résolution qui ont été conçues pendant cette thèse.

Le travail de recherche a porté sur la création de méthodes de résolution dédiées au VRP riche et à leur intégration dans l'environnement intégré de résolution pour fournir aux utilisateurs des solutions qui correspondent à leurs besoins avec des temps de calcul acceptables.

Les problèmes de VRP riches, qui sont l'objet principal de cette thèse, concernent des problèmes de logistique du dernier kilomètre. Ce positionnement signifie qu'il s'agit de problèmes de VRP apparaissant essentiellement en milieu urbain. Ce milieu apporte des contraintes spécifiques et en particulier l'interdiction d'accès à certaines voies par certains véhicules et des temps de transport qui varient fortement d'une période à l'autre au cours de la journée et des horaires de livraisons stricts à respecter. Les problèmes sont étudiés en considérant :

— la qualité de service du point de vue du client final, qualité qui se mesure essentiellement par le respect des contraintes horaires de livraison,
— la qualité de service du point de vue des chauffeurs des véhicules qui ont besoin de tournées qui ne varient pas au cours de la journée et qui leur permettent de travailler si possible dans des quartiers de la ville qu'ils connaissent,
— la qualité de service du planificateur qui souhaite obtenir une solution de bonne

qualité dans des temps de calculs acceptables.

Ces considérations ont naturellement amené à concevoir des méthodes de résolution du VRP riche qui tirent profit des méthodes de clustering. Ces méthodes permettent de réduire la combinatoire et de garantir des tournées localisées réduisant ainsi l'itinérance des chauffeurs sur plusieurs quartiers. Ce type d'approche étend les approches connues sur le VRP de type *cluster-first route-second* en intégrant dans la notion de cluster des caractéristiques du graphe routier.

Cette analyse nous permet d'envisager la résolution d'une nouvelle variante du VRP, le Soft Layered Clustered VRP. Notre contribution porte d'une part sur l'association des caractéristiques du graphe routier et l'utilisation d'une méthode de clustering. D'autre part, nous proposons une approche de résolution du VRP riche constituée de deux étapes. La première étape est une méthode d'agrégation de données où les clusters sont des nœuds d'un graphe « agrégé » modélisant les besoins des clients en terme de quartiers (un nœud est alors un quartier déterminé par un algorithme de clustering). La résolution est réalisée à partir de ces données agrégées. La deuxième étape de la méthode consiste à créer, à partir de cette solution « agrégée », une solution du problème initial en résolvant le problème de tournée de manière à combiner les différents quartiers. Les méthodes de résolutions utilisées lors de la deuxième étape de la méthode utilisent soit une méthode de type métaheuristique au travers d'un modèle de programmation par contrainte (PPC), soit des métaheuristiques dédiées à certaines variantes du VRP.

Le travail réalisé pour la méthode de type PPC est un travail de modélisation (qui fait partie du chapitre 3 de cette thèse) qui a permis de tirer profit des spécificités des solveurs de type PPC pour le VRP riche. Le modèle et la métaheuristique utilisés ont été validés sur les jeux de données classiques de la littérature. Cette métaheuristique a montré sa capacité à résoudre les instances classiques de la littérature avec un écart moyen aux meilleures méthodes publiées pour les variantes du VRP inférieur à 7% alors même que elle a été conçue pour le VRP riche et n'a pas été optimisée pour la résolution du VRP.

Abstract

This thesis deals with the resolution of rich vehicle routing problems in an industrial context of creation of an integrated resolution environment for the customers of the company Mapotempo. The integrated resolution environment belongs to the family of decision support systems and has been designed in a client-server context with a direct link to the web and its applications.

The integrated resolution environment offers data manipulation services at different levels of abstraction from data manipulation at the graph node level, through shortest path computation, to solution computations for the rich VRP. The data manipulation services are offered in the form of an interface accessible in a client-server context, which has allowed Mapotempo to create a web-application to enhance the methods implemented in the integrated solution environment that were designed during this thesis.

The research work has been focused on the creation of resolution methods dedicated to rich VRP and their integration in the integrated resolution environment to provide users with solutions that meet their needs with acceptable computation times.

Rich VRP problems, which are the main focus of this thesis, concern last mile logistics problems. This positioning means that they are VRP problems appearing essentially in urban environments. This environment brings specific constraints and in particular the prohibition of access to certain roads by certain vehicles, transport times which vary strongly from one period to another during the day, and strict delivery schedules to respect. The problems are studied by considering :

— the quality of service from the point of view of the final customer, which is essentially measured by the respect of the delivery time constraints,
— the quality of service from the point of view of the drivers of the vehicles who need rounds that do not vary during the day and to work if possible in areas of the city that they know,
— the quality of service of the planner who wishes to obtain a good quality solution in acceptable calculation times.

These considerations have naturally led to the design of rich VRP solution methods that take advantage of clustering methods. These methods reduce combinatoriality and guaran-

tee localized routes, thus reducing the roaming of drivers over several districts. This type of approach extends the known cluster-first route-second VRP approaches by integrating road graph characteristics into the notion of cluster.

This analysis allows us to consider the resolution of a new variant of the VRP, the Soft Layered Clustered VRP. On one hand, our contribution is about the association of road graph features and the use of a clustering method. On the other hand, we propose an approach to solve the rich VRP based on two steps. The first step is a data aggregation method where the clusters are nodes of an "aggregated" graph modeling the customers' needs in terms of districts (a node is then a districts determined by a clustering algorithm). The resolution is performed on the basis of this aggregated data. The second step of the method consists in creating, from this "aggregated" solution, a solution of the initial problem by solving the vehicle routing problem in a way that combines the different districts. The solution methods used in the second step of the method use either a metaheuristic method through a constraint programming (CP) model, or metaheuristics dedicated to certain variants of the VRP.

The work done for the CP method is a modeling work (which is part of chapter 3 of this thesis) that allowed to take advantage of the specificities of CP solvers for the rich VRP. The model and the metaheuristic used have been validated on classical datasets of the literature. This metaheuristic has shown its ability to solve the classical instances of the literature with an average deviation from the best published methods for variants of the VRP of less than 7% even though it was designed for the rich VRP and has not been optimized for solving the VRP.

Chapitre 1

Contexte de l'étude

Glossaire

Acheminement	Transport permettant de répondre aux demandes des clients finaux
Chauffeur	Individu chargé de réaliser les tournées
Client final	Personne physique ou morale ayant effectué un achat
Consommateur	Personne physique ou morale pouvant effectuer un acte d'achat
Demande	Produit ou service demandé par un client final
Consolider	Désigne le groupement de plusieurs demandes vers un même client final afin d'y répondre en une fois

1.1 Dernier kilomètre

L'expression dernier kilomètre désigne le dernier maillon de la chaîne logistique, le segment permettant la distribution finale des biens ou des services au client final (Cardenas *et al.*, 2017). S'il s'agit principalement de répondre aux besoins de particuliers ou de petites structures, les quantités à acheminer à chaque nœud [1], appelées demandes, sont faibles. Les coûts associés à leur acheminement sont donc par essence les plus élevés de la *supply chain*. En effet, il est rarement possible de consolider la demande. En particulier, Macharis et Melo (2011) étudient les freins au déploiement des centres de consolidation dans le cadre de la logistique urbaine. Ces centres centralisent la réception des produits sur une unique plateforme avant de les transmettre pour leur acheminement. Or, si un client demande de multiples produits, leur acheminement fait intervenir plusieurs acteurs par des canaux différents. La mise en place de tels centres demande donc aux acteurs de la *supply chain* de déléguer une partie de la chaîne à un autre acteur, le plus souvent, public. Par ailleurs, même s'il est possible de grouper les demandes, l'arrivée des produits

1. Le réseau routier peut être représenté comme un graphe orienté. Dans ce graphe, des sommets représentent des points géographiques avec des coordonnées. Nous appelons nœud une activité à réaliser sur l'un de ces points. Un point peut avoir plusieurs nœuds.

se fait de manière différée. Il existe donc également une incertitude sur les dates d'arrivées des autres produits. De plus, le coût de stockage et le manque de coopération entre les différents acteurs rendent la consolidation à la fois :

— peu probable, puisqu'il faut que l'ensemble des flux soient synchronisés,
— peu rentable, puisqu'il faut prendre en compte le coût de stockage,
— et demande une coordination massive des différents acteurs.

Le dernier kilomètre prend principalement place dans un milieu urbain. Dans les centres urbains, la concentration de la demande et une importante variation dans la congestion du réseau routier rendent la mise en place de solutions robustes difficiles. Une tournée peut être rendue irréalisable par : un arrêt qui prend plus de temps que prévu, la congestion du réseau routier ou tout autre aléa qui rend impossible l'arrivée dans la fenêtre de temps prévue.

Les véhicules qui réalisent ces tournées peuvent être de natures différentes : piéton, vélo, véhicule léger, poids lourds, etc. Chaque mode de transport est soumis à des restrictions de circulation et des vitesses qui lui sont propres.

Par ailleurs, l'essor des plateformes de marchés en ligne (*online marketplaces*), tels que AliExpress, Amazon ou Ebay, a changé les réflexes d'achat des consommateurs. Les achats sur ces plateformes de marchés en ligne ont apporté un changement profond sur le dernier kilomètre. Là où, les achats étaient auparavant effectué auprès de commerces de proximité qui concentraient la demande. L'éclatement de la demande a mis les sociétés de transport, qui sont en charge d'acheminer les demandes, au contact direct du client final. Ce changement, en plus de rendre plus difficile la livraison, s'accompagne d'une transformation complète de l'expérience client. Quand le consommateur prenait auparavant sur son temps libre pour aller en boutique, il peut maintenant recevoir ses demandes à toute heure du jour et en tout lieu. Cela pose la question des échecs à la livraison. Le client final peut ne pas être disponible sur tout ou partie des horaires de livraison. Cela demande d'informer précisément le client final des horaires de livraison, récupérer ses préférences et ajuster les tournées. Il est également possible de changer le point de livraison en fonction des horaires pour correspondre au mieux à la préférence et à la position du client.

L'augmentation de l'activité logistique en centre urbain voit de nouvelles contraintes apparaître pour réduire les nuisances et la pollution liées à ce secteur. De nouvelles normes vont progressivement pousser les acteurs à basculer leur flotte de véhicule vers des modes de livraison plus vertueux. En effet, les véhicules thermiques vont progressivement être interdits dans les grands centres urbains (Pouponneau et Cape, 2017). Les niveaux de

nuisances sonores autorisés vont également être réduits. Ces évolutions peuvent passer par un changement de flotte vers des véhicules au gaz vert ou électriques. Elles peuvent également passer par des changements complets de mode de transport : acheminement à pied, triporteurs, vélos, etc.

1.2 Mapotempo

En 2012, Mehdi Jabrane part du constat que les solutions existantes sont trop complexes et s'adressent à des experts en logistique. Or, l'explosion du e-commerce apporte un flot d'utilisateurs pour lesquels ces solutions sont inexploitables ou demandent un investissement trop important. Mapotempo voit le jour, au sein de l'incubateur de la Technopôle Helioparc à Pau (64), pour démocratiser les solutions d'optimisation sans exiger de l'utilisateur qu'il soit issu d'une formation en logistique. L'objectif est de proposer une solution utilisable dans le cadre de toutes les activités du dernier kilomètre quel que soit le secteur d'activité : portage de repas, visites commerciales, distribution de presse, de médicaments, ramasse de linge, collecte de déchets, etc.

Frédéric Rodrigo, expert en cartographie et impliqué dans le projet OpenStreetMap (*maintainer* du projet de qualité osmose), rejoint le projet Mapotempo en 2013 et apporte son expertise technique. Il installe le pôle Recherche et Développement à Bordeaux (33). Dès lors, le développement de Mapotempo Web, solution SaaS de cartographie et d'optimisation de tournées, est initié. Sa commercialisation débutera en 2014. Le pôle R&D emménage à la Pépinière Éco-créative des Chartrons à Bordeaux en 2015. De nouveaux produits sont créés à partir des principes explorés au travers du projet Mapotempo Web et exposés sous formes d'API : routage, géocodage, vue de Mapotempo Web et optimisation de tournées. Ce choix de développer des interfaces pour les voir intégrer dans les solutions d'éditeurs tiers est appuyé par la signature d'un partenariat technologique avec Apologic intégré au groupe UP qui est un éditeur leader sur le marché de l'action sociale. À titre personnel, j'ai rejoint le projet en novembre 2015.

Une levée de fonds est menée auprès d'institutionnels de Nouvelle-Aquitaine en 2016. Celle-ci permet de doubler l'effectif, portant le nombre de collaborateurs à 13. De nouveaux partenariats sont conclus auprès de structures telles que TomTom Telematics, Idea Informatique et VIF. Par ailleurs, Mapotempo rejoint le consortium européen SELIS (Shared European Logistics Intelligent Information Space) qui réunit 38 acteurs européens de la logistique pour co-construire une plateforme d'échange innovante pour une logistique

écologique, agile et collaborative.

En 2017, Mapotempo intègre le groupe international Octime-Spec. Le groupe est présent dans 40 pays et est leader dans les domaines de la planification RH, de la gestion des temps et du contrôle d'accès. Rejoindre Octime-Spec permet à Mapotempo de montrer sa crédibilité et sa viabilité auprès d'entreprises de taille importante qui ont besoin de s'assurer de la pérennité du choix technologique d'optimisation de tournées. Il s'agit également d'un rapprochement humain, puisque Guillaume Berbinau, président du groupe Octime est également le parrain d'entreprise de Mapotempo. Le siège social est alors déplacé de Pau à Biron (64).

Un nouveau produit, Mapotempo Live est lancé en 2018. Il s'agit d'une application Android qui permet l'exécution et le suivi en temps réel des tournées planifiées et optimisées sur Mapotempo Web. Par ailleurs, les demandes relatives à l'optimisation de tournées augmentent à la fois en nombre de variantes à traiter et en taille d'instance. La taille des entreprises et des structures intéressées laisse présager des besoins importants. Il s'agit alors du point de départ de la thèse présentée ici ainsi que de l'équipe R&D Recherche Opérationnelle au sein de Mapotempo.

1.3 Contexte

Mapotempo a commercialisé son premier projet en 2014, Mapotempo Web qui intégrait les prémisses de l'optimisation de tournées en proposant de résoudre le problème du voyageur de commerce (Traveling Salesman Problem TSP) avec la prise en charge de la variante Semi-Soft Time Windows (TSPSSTW) (Qureshi *et al.*, 2010). Dans cette variante, il n'est pas possible de servir un nœud avant le début de sa fenêtre de temps, mais arriver après la fin de cette fenêtre est possible et génère une pénalité.

C'est dans ce contexte, que j'ai rejoint Mapotempo en novembre 2015 afin d'internaliser les compétences en Recherche Opérationnelle, explorer les possibilités avec les outils actuels, prospecter sur les outils qui pourraient être utilisés, déterminer les concepts qui pourraient être généralisés pour enrichir l'offre commerciale de Mapotempo. Nous avons dès lors pu commencer à intégrer les besoins remontés par nos clients dans les modèles utilisés en adaptant les concepts proposés dans la littérature.

Par ailleurs, des discussions ont débuté au cours de l'année 2015 avec le groupe de presse Ouest-France SIPA. Au cours des échanges, deux problématiques principales pour l'acheminement de ses journaux ont été définies, selon la terminologie de l'entreprise :

— le portage - livraison directe aux abonnés,
— le routage - livraison de points de ventes ou d'approvisionnements.

Cela a été la première étape dans les travaux de recherche en RO au sein de Mapotempo. Nous avons tout d'abord exploré les pistes qui nous étaient accessibles afin de répondre aux besoins des tournées de portage à l'échelle d'un département, soit environ 20 000 adresses à livrer. Or, à cette période Mapotempo Web ne permettait de traiter que le TSP et sa variante TSPSSTW. Notre première approche fut de déterminer l'impact de méthodes de clustering pour découper le problème global pour obtenir des groupes de taille modeste à résoudre individuellement. Les résultats étaient corrects avec toutefois un déséquilibre entre les sous problèmes ainsi générés. Nous avons alors mis en place la résolution du VRP en ajoutant à notre modèle des contraintes aux véhicules dont une amplitude horaire limitée et une contrainte de capacité qui nous permettent d'assurer la faisabilité des tournées. La preuve de concept a été concluante et nous avons pu entamer l'autre partie du projet.

La problématique de routage a requis de traiter un ensemble de variantes du VRP supplémentaires. En effet, Ouest-France, dans leur activité, édite, depuis leurs imprimeries, plusieurs tirages de manières successives pour l'édition standard et les éditions locales. Ceci nous a obligé à considérer de multiples capacités. Le tirage des différentes éditions se fait de manière séquentielle. Elles sont donc disponibles à des horaires différents pour l'expédition. Les véhicules qui ont débuté une tournée lors de la mise à disposition des premières éditions peuvent, et pour certains doivent, revenir au dépôt pour se réapprovisionner afin de servir les clients dits dépositaires. Nous avons exploré la faisabilité de cette contrainte en utilisant Jsprit (Schröder, 2014) de Graphhopper, qui implémente une méta-heuristique paramétrable permettant de traiter de nombreuses variantes du VRP. Cet outil nous a permis de démontrer la faisabilité du projet grâce au VRP avec Pickup and Delivery et Time Windows (PDVRPTW) (Ropke et Pisinger, 2006).

En parallèle de ce projet, nous avons commencé à mettre en place le projet Optimizer-API afin d'y interfacer les différents solveurs que nous souhaitions utiliser, à savoir Jsprit, OR-Tools (Perron et Furnon, 2013) et VROOM (Coupey, 2015). Ce dernier nous étant utile pour résoudre les TSP. Le projet Optimizer-API, ainsi que OR-Tools et VROOM seront présentés plus en détail dans le chapitre 3. OR-Tools et Jsprit permettent de couvrir un large panel de variantes du VRP. Jsprit intègre une métaheuristique générique prête à l'emploi. OR-Tools est un solveur de programmation par contrainte (PPC) pour lequel nous avons mis en place un modèle qui correspond à nos besoins. Nous avons mis du

temps à recouvrir l'ensemble des fonctionnalités de Jsprit par notre modèle PPC, mais puisqu'elles n'étaient pas alors disponibles au travers de Mapotempo Web, nous n'avons jamais eu besoin de déployer Jsprit sur nos serveurs de production. En effet, nous voulions favoriser OR-Tools puisqu'il permet de couvrir les variantes de Semi-Soft Time Windows. De plus, Jsprit est gourmand en mémoire et a des performances moins bonnes que OR-Tools. L'un des avantages de la PPC est qu'un problème très contraint verra le domaine de ses variables réduit et la recherche sera par conséquent accélérée. C'est en particulier le cas sur la génération de premières solutions sur les instances de PDVRPTW proposées par Ouest-France (environ 600 dépositaires à livrer).

OR-Tools nous permet d'obtenir des solutions de bonne qualité dans un temps raisonnable pour des instances allant jusqu'à 500 clients finaux. Cependant au-delà, nous avions 2 problématiques. Tout d'abord, le temps nécessaire au calcul des matrices de temps et de distances en amont de la résolution devient conséquent. Ensuite, la recherche de solutions de bonne qualité se heurte à la complexité du problème. La résolution devient longue, or le temps alloué est limité. Interrompre la résolution prématurément amène alors à retourner un résultat qui est médiocre.

Cependant, puisque Mapotempo se positionne sur le créneau du dernier kilomètre, les nœuds à visiter se trouvent souvent concentrés autour de quelques zones où la densité est importante. Nous avons alors commencé à réfléchir aux moyens qui étaient à notre portée pour grouper ces grands ensembles de points, en basant notre réflexion sur les essais déjà mené pour Ouest-France et les retours de nos autres clients.

Différentes questions de recherche se sont alors dessinées :

— Comment augmenter la taille des instances traitables en gardant un temps de résolution raisonnable et une qualité de résultat conforme aux attentes des clients ?
— Quelles méthodes de clustering permettent de traiter un volume important de données et d'intégrer un nombre important de contraintes liées aux problèmes de tournées de véhicules ?
— Comment peut-on faire interagir les méthodes de clustering et la résolution du VRP ?
— Comment conserver un maximum de flexibilité pour prendre en charge le plus de variantes du VRP possibles ?

1.4 Qualité de Service

La logistique du dernier kilomètre consiste à satisfaire les demandes des clients finaux. Il s'agit alors d'une production de service, ce qui inclus la logistique du dernier kilomètre dans le secteur tertiaire. Ainsi contrairement aux secteurs primaires et secondaires la valeur produite est intangible (Alzaydi *et al.*, 2018). En effet, même s'il s'agit de livraison de colis, le service, en lui-même, ne se matérialise pas par un produit que le client final peut visualiser ou manipuler. Cette nature du service le rend alors difficilement quantifiable.

La prise en compte de la qualité de service dans la résolution des problèmes de tournées de véhicules est peu courante. Expósito *et al.* (2019) appliquent la qualité de service au VRPTW en mesurant le positionnement de la visite d'un nœud par rapport à sa fenêtre de temps. Les travaux de synthèse de Paquette *et al.* (2009) montrent en effet que la notion de qualité en Recherche Opérationnelle (RO) se concentre principalement sur la relation au temps. Ces critères peuvent impliquer le respect des contraintes, tel que le fait de servir un nœuds dans ses fenêtres de temps. Ils peuvent également mesurer la différence entre les tournées prévues et les tournées réalisées en termes d'horaire de passage, de temps d'attentes ou de temps de routes. Ces travaux de synthèse ont pu être appliqués au problème de transport à la demande (Dial-a-ride problem) (Paquette *et al.*, 2012).

Mapotempo ne se positionne pas uniquement sur la résolution de problèmes à l'aide des méthodes de RO. En effet, puisque nous intervenons à différents niveaux de la prise de décision et du suivi des tournées la notion de service s'applique à différents profils :

— le **client final** reçoit la visite d'un chauffeur pour satisfaire sa demande,
— le **chauffeur** reçoit de Mapotempo ses tournées,
— le **sous-traitant** envoi ses chauffeurs effectuer ses tournées,
— le **planificateur** utilise les outils à sa disposition pour construire les tournées,
— le **donneur d'ordre** souhaite que le client final voie sa demande satisfaite avec la meilleure qualité de service possible.

Il faut alors intégrer des mesures de qualité de service qui se reportent à ces différents acteurs. Les donneurs d'ordres et les sous-traitants ont leurs mesures de satisfaction directement liées à celles du client final, du chauffeur et du sous-traitant. Ces mesures sont complétés par l'équilibre entre le coût de l'acheminement et le revenu perçu pour celui-ci.

1.4.1 Client final

Mapotempo n'est pas chargé de visiter le client final pour satisfaire ses demandes, nous ne pouvons donc intervenir qu'en tant que conseiller. Les mesures de satisfaction liées à la prestation effectivement réalisée par le chauffeur et perçues par le client final ne sont pas mesurables ici. Cependant, nous pouvons améliorer un certain nombre de points qui sont liés à cette visite. En particulier, nous pouvons déclencher à différents instants de la planification ou de la réalisation l'envoi de notifications au client final. Ces notifications lui permettent d'obtenir l'horaire d'arrivée du chauffeur. Cet horaire est affiné à mesure que l'horaire de la visite approche. Cet horaire de livraison doit en effet être assez large avant le départ de la tournée afin de ne pas être impacté par les aléas qui pourraient alors rendre l'expérience du client final négative. La partie optimisation doit prévoir et garantir le respect des contraintes énoncées par le client final. Pour assurer la faisabilité des tournées fournies le système, qui planifie et accompagne la réalisation des tournées, doit utiliser des données fiables. De plus, l'affectation régulière d'un même chauffeur à un client final permet de créer une relation client (Tseng et Wu, 2014). Le chauffeur devient alors le représentant de l'ensemble de la chaîne logistique. Le client final est une source d'information importante et peut également en cas de satisfaction recommander le service.

1.4.2 Chauffeur

Un chauffeur exécute une partie critique du service car il est en interaction directe avec le client final. De ce fait, il est important de placer les chauffeurs dans les meilleures dispositions pour réaliser ce service. Une première mesure importante est alors le respect des contraintes horaires ainsi que des pauses qui interviennent en cours de tournées. Maguire *et al.* (2006) ont montré au travers de la comparaison des chauffeurs de taxi et de bus londoniens que les connaissances liées au déplacement dans l'espace étaient impliquées dans l'augmentation du volume de la matière grise de l'hippocampe. Celui-ci faciliterait ainsi le déplacement dans l'espace, mais cette augmentation de volume viendrait au prix d'une réduction la mémorisation de nouveaux espaces. L'affectation des chauffeurs à leurs secteurs de prédilections et de taille restreinte aurait, en plus de leur préférence, un impact sur leurs capacités à long terme.

1.4.3 Planificateur

Les planificateurs reçoivent les demandes à satisfaire par le donneur d'ordre. L'organisation des tournées est de leur responsabilité. Les outils à leur disposition doivent correspondre à leur besoin et permettre de mettre en place l'organisation voulue. De plus, le dernier kilomètre étant un environnement dynamique, les tournées doivent pouvoir évoluer rapidement. Puisque le planificateur a des objectifs financiers sur les tournées, chaque changement a un impact puisqu'il modifie le coût associés à l'acheminement. Ces changements doivent être quantifiés afin de donner au planificateur les indicateurs nécessaires au choix le plus pertinent afin de répondre aux aléas en trouvant le meilleur compromis en objectifs financiers et qualité de service pour les clients finaux et les chauffeurs.

Du point de vue du planificateur, la qualité de service passe essentiellement par la simplicité des décisions et la fluidité des transmissions entre le donneur d'ordre et les chauffeurs. Également, il doit avoir un retour sur la réalisation des tournées. Ainsi, il peut réagir et choisir les meilleures options lorsqu'un aléa survient sur l'une des tournées.

Il est à noter que les planificateurs rencontrent des freins importants à présenter aux chauffeurs des tournées dont les tracés se croisent ou se superposent. En effet, en utilisant des distances euclidiennes, il parait peu probable que des tournées qui se croisent soient optimales. Cependant, il existe des cas où la structure du réseau routier peut obliger la solution à contenir de telles configurations. La modélisation et la résolution proposent souvent un ordre de passage qui est contre intuitif pour les chauffeurs. Si les planificateurs n'ont pas les clés pour expliquer les choix effectués, alors ils jouent leur crédibilité auprès de leurs chauffeurs. Cela peut engendrer des conflits.

De plus, comme représenté dans la figure 1.1, lorsque 4 nœuds sont répartis autour d'un carrefour où le véhicule ne peut pas faire demi-tour facilement, la projection des points sur l'axe routier le plus proche et la modélisation du VRP feront que la solution proposée fera un détour pour servir les deux axes (il s'agit d'ailleurs d'une situation, où il est autorisé de servir les deux côtés d'une voie lors de son parcours). En réalité, un chauffeur aura tendance à stationner son véhicule auprès du premier nœud à servir, puis à parcourir les autres nœuds autour du carrefour dans une micro tournée à pied, tel que représenté dans la figure 1.2.

La résolution du VRP permet certes, au global des gains opérationnels conséquents, mais interrogent les équipes terrains. Ce type de configuration amène donc les décideurs à préférer les tournées bien sectorisées puisqu'elles sont plus facilement acceptées par les équipes terrains qui trouvent ainsi la possibilité d'ajuster certaines parties des tournées.

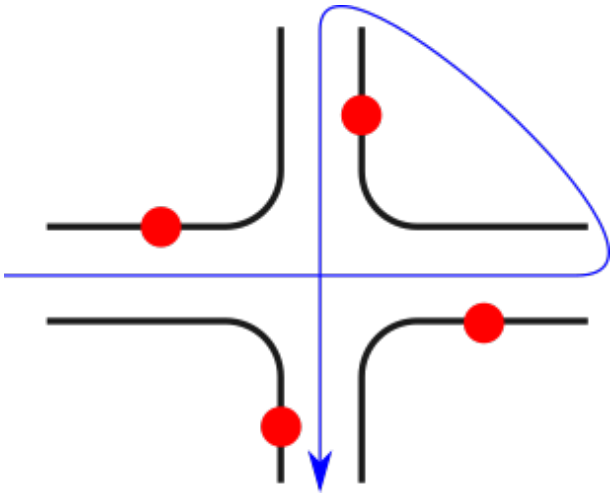

FIGURE 1.1 – Projection et parcours autour d'un carrefour avec la modélisation

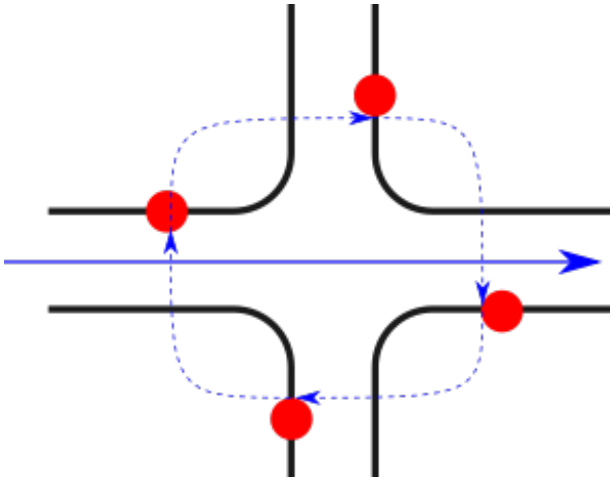

FIGURE 1.2 – Stationnement et parcours à pied autour d'un carrefour

L'organisation des tournées sectorisées devient alors un outil de gestion macroscopique qui permet lors d'indisponibilités de ressources de transférer tout ou partie de tournées en faisant simplement basculer ces secteurs vers une autre tournée plutôt que chaque point individuellement. De plus les tournées optimisées et sectorisées fournissent un guide pour les entreprises qui accueillent de nouveaux collaborateurs. Cette organisation donne également aux chauffeurs expérimentés une certaine latitude dans la réalisation de leurs tournées.

Cette orientation vers l'intégration de méthodes de clustering a plusieurs objectifs :

— réduire la complexité du VRP et de ses variantes,
— faciliter l'exploitation des résultats par les décideurs,
— donner aux chauffeurs des tournées cohérentes globalement et qui peuvent bénéficier de leur expérience localement.

1.5 Vehicle Routing Problem

Le problème de tournées de véhicules (VRP) permet de donner un modèle aux situations où une flotte de véhicules doit se déplacer pour servir un ensemble de clients finaux, représentés par des nœuds dans un graphe [2]. Sa définition a été introduite par Dantzig et Ramser (1959) comme une généralisation du problème de voyageur de commerce (TSP) dans lequel il n'existe pas un mais plusieurs véhicules qui ont une capacité limitée.

Ces véhicules doivent visiter l'ensemble des nœuds formant le problème pour servir leurs demandes. Ici, les véhicules ont une capacité homogène et partent tous d'un unique dépôt. La distance entre les nœuds à visiter est modélisée dans une matrice de distances. L'objectif est de minimiser la distance totale parcourue par la flotte de véhicules pour satisfaire les demandes des nœuds.

Depuis lors, de nombreuses variantes ont définies les contraintes rencontrées par la *supply chain* : les travaux de synthèse de Vidal *et al.* (2013b) et Caceres-Cruz *et al.* (2014) permettent d'avoir un aperçu exhaustif des recherches menées par la communauté. Les variantes présentées ci-dessous sont un aperçu des thématiques qui sont rencontrées régulièrement par les clients de Mapotempo.

2. Ici, un client final est représenté par un unique nœud. Le Generalized TSP (GTSP) (Noon, 1988) représente un client final par un groupe de nœuds mutuellement exclusifs. L'objectif est alors de visiter chaque groupe exactement une fois.

1.5.1 Variantes

Contraintes horaires

Les contraintes horaires peuvent se matérialiser sur plusieurs aspects du VRP.

Tout d'abord, les nœuds à visiter peuvent avoir un ou plusieurs créneaux horaires d'ouverture, il s'agit du VRP with Time Windows (VRPTW) (Cordeau *et al.*, 2001; Sivaram Kumar *et al.*, 2014; Gendreau et Tarantilis, 2010).

Les clients finaux ont des impératifs horaires, qui se matérialisent pas des fenêtres de temps sur les nœuds à visiter. C'est également le cas des chauffeurs qui réalisent les tournées car ils ont une amplitude horaire limitée, une durée de travail maximale et des pauses à prendre de manière fixe ou régulière (Goel, 2009; Benkebir *et al.*, 2019).

De manière générale, si un nœud n'a pas de fenêtre de temps, les horaires de visites sont limités par l'amplitude horaire appliquée aux véhicules.

Dépôts multiples et tournées ouvertes

Les véhicules partent d'un dépôt pour leurs tournées. Le dépôt est un point géographique d'où les véhicules débutent ou terminent leur tournée. Chaque véhicule a alors un nœud de début et de fin localisé au dépôt. Dans le cas de transport de marchandises, un dépôt peut être un point d'approvisionnement auprès duquel les véhicules se fournissent pour pouvoir satisfaire les demandes des clients finaux.

Cependant, un dépôt n'est pas nécessairement unique, par exemple, différents points d'approvisionnement peuvent être répartis autour des centres urbains. Les dépôts peuvent également correspondre aux domiciles des chauffeurs, en particulier dans le cas de tournées de commerciaux. La variante Multi-Depot VRP (MDVRP) permet de modéliser ces cas de figure (Cordeau *et al.*, 1997; Dondo et Cerdá, 2007).

Les véhicules sont néanmoins rattachés à un dépôt duquel ils partent et reviennent. Dans certains cas de figures, en particulier où une partie de la flotte est externalisée, le retour au dépôt n'est pas un pré-requis et la tournée se termine au dernier nœud visité ; cette variante est connue sous l'appelation Open VRP (OVRP) (Soto *et al.*, 2017).

Dans le cas d'interventions de techniciens pour répondre aux demandes des clients finaux, où les tournées sont sous-traitées, que les demandes ne correspondent pas à des biens de consommation à récupérer auprès des points d'approvisionnement et que les tournées sont facturées de la première à la dernière visite, les tournées peuvent être organisées telles qu'elles débutent au premier nœud visité et se terminent au dernier nœud de la

tournée.

Flotte hétérogène

Le Heterogeneous Fleet VRP (HFVRP) permet de modéliser les cas d'applications où l'utilisation de véhicules différents n'implique pas les mêmes coûts d'exploitation ni les mêmes contraintes (Liu et Shen, 1999; Duhamel *et al.*, 2012; Belmecheri *et al.*, 2013). Chaque type de véhicules possède des caractéristiques qui lui sont propres. En milieu urbain, les véhicules de petites tailles sont privilégiés à la fois parce qu'il est plus simple de les stationner et parce les quantités à livrer sont généralement plus modestes et fragmentées. Cela peut passer par les voies du réseau routier que le véhicule peut emprunter ainsi que la vitesse à laquelle il peut parcourir ces voies. Un vélo pourra emprunter des voies interdites à d'autres véhicules. Un poids lourd a une capacité très importante, un véhicule léger a une vitesse moyenne plus importante que les autres types de véhicules. Les coûts de sortie, horaire ou kilométrique varient en fonction du type de véhicule, mais également s'il s'agit d'une flotte en propre, louée ou sous-traitée.

Gestion des quantités

Le transport de biens peut impliquer l'utilisation de plusieurs capacités (poids et volume) ou de comptabiliser plusieurs demandes indépendantes (produits frais et surgelés) matérialisées dans les véhicules par des compartiments indépendants. Dans le cas d'activités telles que la collecte de déchets, le poids est rapidement limitant. En revanche, il est possible, en cours de tournée, d'aller vider les déchets collectés à un exutoire. D'autres domaines demandent de collecter et vider à un couple de nœuds source-destination précis. C'est par exemple le cas de la livraison de colis. Un paquet a un expéditeur et un destinataire uniques. Pour imager la situation, dans la livraison de sable, le point de collecte importe peu, l'important est de livrer du sable. Pour la livraison de colis, il n'est pas possible d'échanger les colis de plusieurs clients. Il s'agit ici du Pickup and Delivery VRP (PDVRP), cela implique de lier deux nœuds qui doivent être dans la même tournée, le nœud de collecte et le nœud de livraison (Desaulniers *et al.*, 2002).

Équilibrage des tournées

La France est un cas particulier où les contrats stipulent une durée de travail fixe. Il est certes intéressant de réduire le temps de route en complément de la distance to-

tale parcourue. Cependant fournir à son équipe un plan de tournées où un chauffeur A travaillera 7h et un chauffeur B travaillera 2h sera peut-être intéressant en termes de coût. Mais cette répartition du temps de travail paraîtra injuste et questionnera certains collaborateurs sur la pérennité de leur poste (He *et al.*, 2009; Zhou *et al.*, 2013).

Périodicité

Dans nombre de cas, l'activité fluctue d'un jour à l'autre et la visibilité ne va pas au-delà de quelques jours. En revanche, dans certains secteurs les demandes sont régulières dans le temps : c'est le cas des visites commerciales, de l'approvisionnement en produits d'hygiène ou en vêtements de travail. Pour ces secteurs d'activités, le plan stratégique doit être réalisé plusieurs mois à l'avance afin de remplir les objectifs et d'assurer la qualité de service fixée contractuellement avec les clients finaux. Des visites régulières, mais variant d'un site à l'autre sont fixées. L'objectif de l'optimisation est alors de proposer des tournées régulières qui prennent en compte la variation d'activité selon les jours, qui lissent la charge de travail et qui réduisent les coûts. On parle alors de Periodic VRP (PVRP) (Vidal *et al.*, 2012; Rahimi-Vahed *et al.*, 2014; Archetti *et al.*, 2017).

Contraintes de précédence

L'exemple du PDVRP a introduit un premier exemple de relation entre deux activités en cours de tournées (Bredström et Rönnqvist, 2008). Les visites sont parfois à réaliser dans un ordre particulier. L'ordre dans lequel sont visités les nœuds des tournées est parfois en grande partie préétabli. À ces tournées, quelques nœuds sont ajoutés, ils doivent être insérés sans modifier l'ordre dans lequel sont servis ces nœuds dans les tournées. Ces pratiques sont courantes dans le secteur du portage de presse où il est possible d'intégrer du fret externe s'il ne provoque aucune augmentation du coût d'exploitation pour les prestataires le réalisant. La relation peut également prendre la forme d'une synchronisation des tournées pour servir simultanément deux nœuds auprès d'un client final. Un véhicule peut avoir un chauffeur ainsi que d'autres agents pour satisfaire des demandes spécifiques. Ces rassemblements sont requis pour satisfaire des demandes où l'équipage d'un seul véhicule n'est pas suffisant (i.e : l'installation de mobilier lourd et volumineux).

Tournées dépendantes du temps

Les tournées sont planifiées en amont de leur réalisation. Une fois la tournée débutée, souvent seul l'ordre de visite des nœuds peut changer (en particulier les demandes chargées sont spécifiques aux clients finaux de la tournée). Il faut alors anticiper la congestion du réseau routier en amont, sans quoi la tournée pourrait être, en totalité ou en partie, irréalisable. Du point de vue des utilisateurs et du client final, la prise en compte du trafic est « simple ». En effet, ils ont accès au quotidien à des calculs d'itinéraires qui prennent en compte la variation du trafic de manière prévisionnelle et en temps réel. Mais ce calcul est effectué seulement entre un point de départ et une destination avec un horaire de départ ou d'arrivée fixés. Le calcul d'itinéraire demandé à beaucoup de variables déjà fixées. Dans le cas de l'optimisation de tournées, l'ordre de parcours des nœuds et les horaires sur lesquels les arcs du graphe seront parcourus n'est pas connu à l'avance et peut évoluer à chaque étape de la résolution. Cette problématique de dépendance des itinéraires au temps rajoute une complexité à un problème déjà très complexe. Cependant, la recherche s'oriente vers cette thématique afin d'y apporter des méthodes de résolution efficace, et se regroupent sous l'appellation Time Dependant VRP (TDVRP) (Taş *et al.*, 2014).

1.5.2 Attributs supplémentaires

Vidal *et al.* (2013b) ont proposé l'utilisation du terme attributs pour désigner de manière générale les caractéristiques et contraintes qui permettent de définir les nombreuses variantes du VRP.

Matrices asymétriques et inégalité triangulaires

Les instances de la littérature proposent généralement des coordonnées dans un espace en 2D et laissent le calcul de la matrice à la discrétion du développeur, la distance est alors dans la majorité des cas une distance euclidienne, voire une distance de Manhattan. Dans ces cas de figure, les distances sont symétriques et ne varient pas d'un véhicule à l'autre. Or, dans un cas d'usage urbain, nous pouvons convenir qu'il n'est pas réaliste de considérer une distance et un temps de parcours euclidiens. Boyacı *et al.* (2021) ont étudiés la pertinence de la distance euclidienne en tant qu'approximation pour résoudre le Steiner TSP [3] et le Steiner CVRP.

3. Un graphe connexe non orienté, à coûts positifs pour chaque arc et un ensemble de nœuds à visiter parmi un ensemble de sommets du graphe. L'objectif est de trouver la plus petite tournée qui parcours

Ces variantes utilisent un graphe non orienté, ce qui n'est le cas que dans les tournées à pied. Les vélos peuvent également parcourir ce graphe non orienté, mais pour les voies à sens uniques (sans autorisation de contre-sens) ils doivent mettre pied à terre, ce qui se fait au détriment de la vitesse. L'approximation proposée ne peut d'ailleurs être adaptée qu'à la distance kilométrique. La distance en temps [4] n'est pas évoquée, hormis pour proposer dans le cas des instances académiques d'appliquer un coefficient constant.

L'approximation euclidienne n'est pas exploitable pour la logistique du dernier kilomètre. Les types de voies, les voies à sens unique et la structure même des villes européennes rendent une projection telle que la distance euclidienne difficilement exploitable. De plus les matrices sont, pour la plupart, calculées avec des itinéraires au plus rapide, les calculs d'itinéraire au plus court permettant d'emprunter des voies en réalité peu intéressantes (i.e : voies forestières avec une vitesse moyenne inférieure à 20 km/h plutôt qu'une voie rapide à 80km/h). Ces calculs d'itinéraires au plus rapide s'opposent à l'objectif de réduire le kilométrage total, dans ces cas, l'inégalité triangulaire n'est pas assurée.

Restriction des clients visitables

Les sites de livraisons ne sont pas toujours adaptés aux véhicules qui peuvent les visiter, il est alors nécessaire de faire venir un véhicule spécifique, par exemple avec un hayon pour décharger les marchandises. Ce véhicule peut en dehors de cette spécificité être en tout point similaire aux autres. Cela peut également se matérialiser par des besoins relatifs aux biens transportés, comme les produits surgelés qui ont besoin d'une chambre froide pour respecter la chaîne du froid. Les particularités requises peuvent être relatives aux compétences du chauffeur effectuant la tournée, pour certaines interventions de santé, des accréditations particulières sont obligatoires pour appliquer des traitements. Du point de vue du modèle, cette restriction d'accès peut être représentée soit par une distance ou un coût « infini » pour les véhicules qui ne peuvent visiter certains points, soit par le retrait des véhicules des domaines des variables représentant l'affectation des véhicules. Cette variante est appelée skill VRP (Cappanera *et al.*, 2011).

chaque nœud. Il est possible de traverser les sommets plusieurs fois.

4. Une distance est une mesure qui sépare deux points. Dans une projection cartographique, on peut utiliser une distance qui représente leur écart dans l'espace. Mais il est également possible de mesurer leur distance comme étant le temps pour parcourir l'espace entre ces deux points.

1.6 Méthodes de résolution

La résolution du VRP peut être réalisée à travers de nombreuses approches qui peuvent être regroupées autour de deux paradigmes : les méthodes exactes et les méthodes approchées.

Les méthodes exactes visent à trouver et prouver que la solution trouvée au problème est optimale. Cette approche est très intéressante et montre une très grande efficacité sur les problèmes de taille restreinte, mais se heurte à la complexité du problème dès lors que la taille des instances augmente. Par exemple, Sadykov *et al.* (2020) résolvent, pour le HFVRP, des instances de Duhamel *et al.* (2011) allant jusqu'à 256 nœuds mais la résolution est limitée à 60 heures. Pour les instances les plus grandes et les plus difficiles, l'optimalité n'est pas prouvée dans ce laps de temps.

L'ordre de grandeur des problèmes et l'échelle de temps pour les résoudre n'est à l'heure actuelle pas compatible avec un usage industriel et en particulier pour les besoins de la *supply chain*, les problèmes changent rapidement, les décideurs peuvent aussi avoir besoin de tester de multiples scénarios dans un laps de temps réduit.

Les méthodes approchées visent à trouver des solutions de bonne qualité dans un temps de résolution limité. Elles ne garantissent pas l'optimalité, mais permettent de traiter des instances de très grande taille telles que celles utilisées par Arnold *et al.* (2019) avec des instances jusqu'à 30 000 nœuds.

1.6.1 Méthodes exactes

Les méthodes exactes ont pour finalité de trouver la solution optimale pour un problème. Lors de la résolution de problèmes difficiles ou de grande taille, ces méthodes permettent d'obtenir des bornes. La borne supérieure (UB) permet d'assurer que la solution optimale aura une valeur de la fonction objectif inférieure ou égale à UB. La borne inférieure (LB) assure que la solution optimale aura une valeur de la fonction objectif supérieure ou égale à LB. Trouver la valeur de la solution optimale revient alors à faire converger la borne UB et la borne LB à une même valeur de la fonction objectif. Attention toutefois, l'égalité entre LB et UB n'est pas toujours possible à atteindre selon les méthodes de résolution. Le VRP est un problème de minimisation, c'est-à-dire que la résolution du problème implique de fournir la plus petite valeur de la fonction objectif.

Les paragraphes suivants introduisent de manière succincte les différentes méthodes exactes pouvant s'appliquer à la résolution du VRP.

Programmation linéaire La programmation linéaire (LP) est un cadre de modélisation pour les problèmes d'optimisation continus à l'aide d'expressions linéaires (Van Dantzig, 1947; Vanderbei, 2020) et la méthode la plus utilisée est l'algorithme du simplexe. Les problèmes d'optimisation combinatoires, dont le VRP fait partie, ne considèrent qu'un ensemble fini d'objets. L'espace de recherche est alors discret. On parle alors de Programmation linéaire en nombre entier (ILP). Puisque la résolution de LP est réalisée dans un espace continu, il est alors possible d'appliquer la méthode des plans sécants (*cutting plane*), proposée par Gomory (1958), de manière à introduire des contraintes supplémentaires afin d'obtenir des solutions entières. Lors de l'ajout d'une telle contrainte, la solution courante devient irréalisable et l'algorithme du simplexe est à nouveau appliqué. Et ce jusqu'à ce qu'une solution entière soit obtenue.

Branch and Bound Le *branch and bound*, proposé par Land et Doig (1960) est basé sur l'énumération des solutions de l'espace de recherche (Clausen, 1999) ; cette dénomination a été proposée par Little *et al.* (1963). Cet espace est décomposé en sous-espaces (*branching*), chaque sous-espace est lui-même décomposé en sous-espace et ce récursivement. L'espace de recherche peut alors être représenté par un arbre de décision. Chaque nœud de cet arbre représente un sous-problème. Pour chaque nœud, il est possible d'évaluer (*bounding*) les bornes LB et UB du sous-problème correspondant. Lors du parcours de l'arbre, la plus petite UB_{min} est conservée, pour tout nœud si la LB est supérieure à UB_{min} alors le nœud correspondant peut être élagué (*pruning*). De cette manière l'arbre de décision est réduit. Bien qu'il soit difficile d'appliquer directement un *branch and bound* au VRP, il sert de base aux approches suivantes de *branch and price*, de *branch and cut* ou de *branch and cut and price*.

Branch and Cut Padberg et Rinaldi (1991) ont proposé de combiner les méthodes des plans sécants et de *branch and bound* (Mitchell, 2002). Le *branch and bound* est utilisé pour diviser l'espace de recherche. La méthode des plans sécants permet lorsqu'une solution comporte une variable non entière d'ajouter deux contraintes. La première contrainte demande à ce que la variable soit inférieure ou égale à la partie entière usuelle. La seconde contrainte oblige la variable à être supérieure à la partie entière par excès. Ceci afin d'obliger la recherche à obtenir des solutions en nombres entiers.

Génération de colonnes Les programmes linéaires de grandes tailles ont beaucoup de variables. Or, la solution optimale ne comporte, très souvent, qu'un petit nombre de

variables, les autres étant nulles. Dantzig et Wolfe (1960) ont alors proposé une méthode qui évite la représentation de toutes les variables (colonnes). Le problème est représenté par :

— un problème maître qui est le problème original avec seulement un sous-ensemble de variables considéré,
— un sous-problème qui génère de nouvelles variables, on parle alors de génération de colonnes, qui ont le potentiel d'améliorer la fonction objectif du problème maître.

Dès lors que de nouvelles variables sont intégrées au problème maître, un nouvel ILP est résolu. La résolution est arrêtée dès lors que plus aucune variable ne peut améliorer la fonction objectif et la solution obtenue est alors optimale.

Branch and Price La méthode de *branch and price* (Barnhart *et al.*, 1998) combine le *branch and bound* à la génération de colonne appliquée à chaque nœud de l'arbre de décision. Un problème maître est définit sous forme d'un problème linéaire relâché. À chaque nœud des colonnes sont ajoutées au problème maître en résolvant un sous-problème.

Branch and Cut and Price La méthode de *branch and cut and price* peut être vue comme une combinaison de la méthode de *branch and cut* pour lequel à chaque nœud, la génération de colonnes est appliquée.

Programmation par contraintes La programmation par contraintes (PPC) est un paradigme de programmation (Rossi *et al.*, 2006). Chaque variable d'un problème à un domaine qui représente l'ensemble des valeurs qu'il peut prendre. Les contraintes permettent de lier des variables et par ce lien réduisent les valeurs que peuvent prendre les variables. Les méthodes de résolution s'appuient principalement sur la propagation sur le mécanisme de propagation de contraintes au travers d'une exploration de l'arbre de décision. Ce paradigme est décrit plus en détail dans la section 3.5.

Les méthodes exactes peuvent être appliquées avec succès sur des instances de taille modeste (Sadykov *et al.*, 2020). De manière générale, effectuer une recherche exacte n'est pas adapté aux problématiques du dernier kilomètre. En effet, à mesure que la taille des problèmes croît, le temps pour fournir une solution optimale augmente considérablement. En particulier, les méthodes relatives à la programmation linéaire ne permettent

pas d'assurer qu'une solution intermédiaire soit une solution entière. En effet, puisque des variables non entières sont utilisées, il n'est pas assuré de pouvoir extraire rapidement des solutions entières situées à l'intérieur du polyèdre que forme l'espace des solutions réalisables. Dans la mesure où, dans les problématiques abordées, nous devons fournir des solutions rapidement, il est intéressant de fournir des solutions même sous-optimales dans le temps accordé à la résolution. Pour cela, la programmation par contraintes correspond au besoin. En effet, du fait de l'application des mécanismes de propagation de contraintes, qui retirent les valeurs irréalisables des domaines des variables, toute solution obtenue est réalisable.

1.6.2 Méthodes approchées

Obtenir des solutions de bonne qualité en un temps limité peut être atteint avec différentes approches et peuvent s'appliquer ou non selon l'objectif à atteindre et les variantes à traiter.

Méthodes d'approximation

Si l'on veut garder une garantie de résultat, il est possible de se tourner vers les méthodes d'approximation. Elles permettent de trouver une solution qui soit entre la solution optimale et une borne supérieure. L'approximation de Christofides (1976) par exemple propose un algorithme pour lequel le pire scénario est prouvé fournir une 3/2-approximation au TSP si le graphe respecte l'inégalité triangulaire. C'est-à-dire, qu'au pire des cas, l'algorithme fournit une solution 50 pourcents plus coûteuse que l'optimum. Le VRP comporte davantage de contraintes et de nombreuses variantes. Or, les approximations sont prouvées pour un problème spécifique ou un cas particulier et même s'il est possible de les généraliser, cela se fait au détriment du ratio d'approximation (Khachay et Zaytseva, 2015). Ce défaut d'adaptabilité explique que ces méthodes soient rarement utilisées.

Heuristiques constructives

Les heuristiques constructives définissent une procédure à appliquer pour construire une solution, les décisions prises ne sont, au cours de la construction, pas remises en causes. Dans le cas du VRP, on détermine dans quel ordre sont traités les nœuds du problème et à quelle tournée ils sont affectés selon différents procédés. Voici un panel d'heuristiques :

— **Nearest-Neighbor** Le nœud le plus proche du dépôt est affecté à une tournée. Puis le nœud le plus proche du précédent point inséré est ajouté à la tournée, on procède de proche en proche jusqu'à ce que la tournée soit pleine. C'est-à-dire jusqu'à ce que le nœud à insérer rende le total des demandes du véhicule supérieure à la capacité de celui-ci. Dès lors qu'une tournée est pleine, une nouvelle tournée est ouverte et on cherche le nœud non affecté le plus proche du dépôt. La procédure est appliquée jusqu'à ce qu'il ne reste plus de nœud à insérer ou que toutes les tournées soient pleines. La méthode est simple et rapide, cependant la qualité de la solution obtenue est souvent de mauvaise qualité.

— **Clarke et Wright (1964)** ont proposé une méthode où chaque nœud constitue une tournée. On calcule pour chaque couple de tournées la réduction de coût (appelé*savings* ou gain) que la fusion de leurs tournées entraîne en remplaçant un des arcs menant au dépôt par un arc joignant les deux tournées. Les tournées sont fusionnées jusqu'à ce qu'il ne reste plus de gain à obtenir.

Les heuristiques ont pour objectif de fournir un patron simple de construction adapté au problème à traiter.

Métaheuristiques

Les métaheuristiques sont des méthodes de résolution construites autour de deux mécanismes. Premièrement, l'intensification permet d'améliorer une solution en explorant son voisinage afin d'atteindre un minima local. Deuxièmement, la diversification permet d'échapper aux minima locaux. L'ouvrage de Potvin et Gendreau (2019) donne un aperçu exhaustif de l'état de l'art sur ces méthodes.

Le voisinage d'une solution peut être vu comme l'ensemble des solutions pour lequel une opération simple peut être appliquée pour passer d'une solution à l'autre. Nombre de métaheuristiques utilisent pour l'intensification des opérateurs de recherche locale afin d'améliorer la solution courante. Dès lors que plus aucun opérateur ne permet d'améliorer la solution, on considère qu'elle a atteint un minima local. De nombreux opérateurs existent dans la littérature et s'appliquent dans de nombreuses variantes du VRP, ils peuvent s'appliquer au sein d'une route (intra-route) :

— **2-Opt** deux arcs de la tournée sont supprimés et remplacés,

— **Relocate** on change la position d'un nœud dans la tournée,

— **Swap** on échange la position de deux nœuds dans la tournée.

D'autres opérateurs s'occupent des relations inter-routes :

— **Relocate** on relocalise un nœud dans une autre tournée,
— **Swap** on échange la tournée et la position de deux nœuds,
— **Crossover/2-opt*** on intervertit des séquences de nœuds entre deux tournées.

Les opérateurs de recherche locale permettent d'améliorer une solution existante. Il est donc intéressant de partir d'une solution de bonne qualité afin de converger au plus vite vers des minimas locaux. Pour cela, il est possible d'utiliser des approximations et des heuristiques constructives.

Adaptative Large Neighborhood Search (ALNS)

La méthode ALNS est utilisée par le solveur RADOS (*Routing And Delivery Optimisation Solver*) développé par Flavien Lucas (2020) dans le cadre de ses travaux de thèses sur lequel nous avons pu travailler conjointement afin de l'intégrer au projet Optimizer API.

Les opérateurs de recherche locale sont plus ou moins complexes, certains opérateurs dits de voisinage large ont une première étape de destruction impactant plusieurs nœuds ou tournées, puis une seconde étape de réparation. Leur application, puisqu'elle impacte une partie conséquente de la solution, est coûteuse, en particulier en terme des temps d'exécution. La méthode Adaptative Large Neighborhood Search (ALNS), adaptée aux problèmes de tournées de véhicules par Ropke et Pisinger (2006), permet de n'appeler que les opérateurs qui sont pertinents. Initialement, tous les opérateurs ont une fréquence d'utilisation identique. Progressivement les voisinages peu utilisés, puisqu'ils n'ont pas apporté d'amélioration, voient la fréquence à laquelle ils sont essayés être réduite. La fréquence des opérateurs qui sont appliqués régulièrement est augmentée. La phase de diversification revient à reconstruire une nouvelle solution à l'aide des nouvelles pondérations des opérateurs.

Greedy Randomized Adaptative Search Procedure (GRASP)

La méthode GRASP est accessible directement depuis la libraire de OR-Tools dédiée aux problèmes de tournées de véhicules.

Les heuristiques constructives sont déterministes, l'application d'une de ces méthodes à une même instance donnera une seule solution. La méthode Greedy Randomized Adaptative Search Procedure (GRASP) introduit de l'aléatoire dans ces méthodes. Puisque l'objectif des heuristiques constructives est de générer une solution rapidement, l'introduction d'un paramètre aléatoire permet de garder cette propriété et également d'apporter de la diversification dans les solutions rencontrées. Cette méthode a été introduite par Feo

et Resende (1989), puis améliorée en y intégrant de la recherche locale (Feo et Resende, 1995). Similairement, le mécanisme de diversification passe par la construction d'une nouvelle solution.

Recherche Tabou

La méthode Tabou est accessible directement depuis la libraire de OR-Tools dédiée aux problèmes de tournées de véhicules. Dans les autres métaheuristiques, les opérateurs de recherche locale ne sont appliqués que s'ils améliorent la solution courante. Ainsi, atteindre un minima local demande de passer par une phase de diversification. La méthode tabou permet de réunir les mécanismes de recherche locale et de diversification. La méthode tabou applique le meilleur voisinage, mais plutôt que de n'appliquer les opérateurs de recherche locale que lorsque la solution est améliorée. Ainsi, la méthode tabou permet d'échapper à un minima local lorsqu'il est atteint en autorisant un voisinage qui dégrade la solution courante (Glover, 1986). Afin d'échapper à ce minima, une liste de mouvements ou de solutions taboues est mise à jour. Ce mécanisme force ainsi l'exploration de nouvelles solutions.

Algorithmes génétiques

Cette méthode a été généralisée pour un grand nombre de variantes du VRP par Vidal *et al.* (2014) qui en fait une des méthodes principales de l'état de l'art.

Les méthodes citées précédemment n'améliorent et ne se basent que sur une seule solution. Or, il existe des méthodes à base de populations qui utilisent un utilisent et améliorent simultanément un ensemble de solutions. C'est notamment le cas des algorithmes génétiques introduits par Holland (1975) puis Goldberg (2006). Une population initiale d'individus est créée, chaque individu est représenté par un ensemble de chromosomes qui représente la solution. À chaque nouvelle génération, des individus parents sont sélectionnés, de manière probabiliste parmi les individus les plus adaptés. Par adapté, on entend ceux pour lequel la fonction objectif retourne le coût le plus intéressant. Les individus parents sont utilisés pour la phase de reproduction, on combine ces individus afin d'en obtenir de nouveaux, des individus enfants. Les individus enfants peuvent subir des modifications aléatoires par le biais de mutations. À chaque itération les individus enfants de l'itération précédente deviennent des individus parents. Des mécanismes de conservations des élites peuvent être mis en place afin de conserver les individus qui ont la meilleure valeur pour la fonction objectif.

1.7 Organisation

Le chapitre 2 s'intéresse à l'ensemble des services mis en place par Mapotempo pour répondre aux différents besoins rencontrés dans l'organisation de tournées pour le dernier kilomètre. Il a pour objectif de décrire la structure générale utilisée dans les différents services et comment ces services s'articulent entre eux pour former un ensemble cohérent dans lequel chaque outil à une fonction bien déterminée.

Le chapitre 3 décrit en détail le service Optimizer-API dédié à la résolution du VRP et de ses variantes. Nous détaillons son architecture, les attributs et les variantes prises en charge. Nous passons également en revue les différentes méthodes de résolution qui y sont intégrées ainsi que leurs interactions.

Le chapitre 4 passe en revue les différentes méthodes de Clustering existantes et nous y présentons leur intérêt dans les problèmes de routing au travers des approches *cluster-first route-second* et route-first cluster-second. De plus, nous proposons deux méthodes de résolution de type *cluster-first route-second* : *Découpe récursive* et *Dichotomious*

Le chapitre 5 propose une nouvelle variante du VRP riche qui vise à réduire la complexité des problèmes de grande taille en groupant les points proches dans les mêmes tournées sans pour autant forcer à respecter un ordonnancement fixe de ces groupes de points. Nous présentons l'approche utilisée ainsi que les outils qui ont permis de parvenir à une application concrète.

Le chapitre 6 analyse la structure des instances proposées dans la littérature pour les problèmes de routing. Nous débattons de leurs similarités et différences avec les problèmes rencontrés dans le dernier kilomètre. Nous proposons également un outil qui bénéficie de jeux de données ouverts pour générer des instances réalistes. La discussion autour de la littérature nous permet d'aborder la question des performances du projet Optimizer-API à la fois sur les instances académiques et les instances générées.

La conclusion discute des apports de la thèse à la fois du point de vue de Mapotempo et la communauté Recherche Opérationnelle.

Les références aux problèmes et leurs variantes sont identifiées en utilisant leurs appellations et abréviations anglaises.

Chapitre 2

Système d'Aide à la décision

Ce chapitre présente le système d'aide à la décision mis en place par Mapotempo pour répondre aux besoins de la logistique du dernier kilomètre. Ce système d'aide à la décision prend la forme d'un ensemble de *web services*, qui utilisent des sources de données et des outils variés. Une web application vient également s'intégrer à cet ensemble pour servir de démonstrateur des fonctionnalités offertes par les *web services*.

La présentation de ces outils permet de décrire le contexte dans lequel doivent s'intégrer les méthodes de résolution pour le VRP riche.

2.1 État de l'Art

Les systèmes d'aide à la décision (Decision Support Systems DSS) sont des systèmes d'informations qui fournissent une assistance aux humains impliqués dans des chaînes de décisions complexes. Les DSS permettent alors de fournir des outils en vue d'établir des processus métiers en accord avec les objectifs spécifiques de l'activité ciblée. La gestion de la Supply Chain demande, en particulier pour la logistique du dernier kilomètre, d'anticiper des changements et de tolérer des modifications dans la planification. Puisque la solution sur le terrain peut évoluer rapidement, chaque étape de la chaîne de décision doit tolérer de multiples boucles de rétroaction. Même si la première planification ne dépend pas d'un historique, elle doit anticiper l'état du trafic et fournir une solution qui pourra être adaptée selon les aléas pour permettre sa réalisation.

Keen (1987) anticipait la recherche sur les DSS pour la prochaine décennie autour des points suivants :

— Explorer des domaines nouveaux, émergents ou négligés où les compétences des créateurs de DSS peuvent être appliqués,

— Appliquer des modèles et méthodes analytiques ; Adopter une vision plus prescriptive dont les décisions peuvent être prises efficacement,

— Exploiter l'expérience des outils logiciels émergents de l'IA pour construire des systèmes semi-experts,
— Réintégrer la valeur des professionnels des DSS comme la combinaison de leur compréhension de la prise de décision et de leur capacité à prendre avantage des développements dans les domaines informatiques.

Shim *et al.* (2002) constataient l'intégration rapide des méthodes de Recherche Opérationnel (RO) au sein des DSS. En particulier au travers des méthodes liées à la programmation linéaire. Ils anticipaient également l'intégration des métaheuristiques qui formaient alors la nouvelle tendance de la recherche.

Crainic *et al.* (2009) analysent les différentes couches qui composent les systèmes de transports intelligents. Ils identifient les challenges et opportunités qui découlent de la combinaison de ces différentes couches. Ils dégagent 3 directions parallèles et complémentaires : 1) le développement des véhicules et de l'infrastructure, 2) l'électronique embarquée, les technologies de suivi de position et le matériel de communication, 3) la partie méthodologie (modèles et algorithmes) qui transforme les données en informations qui font sens et en conseils intelligents pour les systèmes avancés de gestion du personnel, des opérations et de contrôle. Les institutionnels et les industriels privilégient les deux premiers points. En effet, en combinant les deux premiers points et en fournissant les données extraites à un humain, ce dernier peut déjà prendre des décisions intéressantes avec peu d'outils d'aide à la décision. Ce placement central des méthodes de RO laisse envisager de nombreuses opportunités dans leur utilisation (Azadivar *et al.*, 2009).

Mendoza *et al.* (2009) présentent l'implémentation d'un DSS appliqué au service de eaux de Bogotá afin de résoudre le *distance-constrained* VRP rencontré. Ce système utilise des informations extraites de la base de données clients et d'un système d'information géographique (GIS). Cependant, la résolution d'une unique variante du VRP ne permet d'appliquer cette contribution qu'à une activité particulière dans le cadre d'une organisation précise.

Drexl (2012) synthétise les travaux de recherche relatifs aux problèmes de tournées de véhicules riches. Cette analyse est mise en perspective des besoins rencontrés en pratique dans les *commercial vehicle routing software* (CVRS) où l'ensemble des attributs du VRP peuvent être nécessaires pour la résolution d'un même problème. Certains attributs sont bien étudiés alors qu'ils n'ont pas encore trouvé d'application. C'est en particulier le cas des variantes intégrant la congestion du réseau routier. En effet, les données sont encore peu disponibles ou les logiciels métiers ne permettent pas de fournir des données suffi-

samment fiables et qualitatives. La communauté est ouverte à l'intégration de nouveaux attributs, preuve qu'il reste encore beaucoup de pistes prometteuses et de challenges à relever. Lin *et al.* (2014) ont proposé un DSS dans le cadre de l'optimisation dynamique de services de coursiers. Celui-ci permet d'interagir directement avec le matériel du coursier en lui poussant progressivement les courses à effectuer.

Chassaing *et al.* (2017) ont proposé une plateforme pour intégrer les méthodes de résolution développées dans un *web service* représentant les instances fournies ou générées sur un fond cartographique. Il est possible d'y créer des instances et calculer des matrices issues de calculs d'itinéraires au travers du réseau routier.

Keenan et Jankowski (2019) synthétisent les avancées des 3 dernières décennies en matière d'intégration des DSS dans les systèmes GIS et constituent une branche de recherche sous l'appellation Spatial Decision Support Systems (SDSS). Ils concluent que les contributions entres les différentes disciplines sont déconnectées, ce qui ralentit la diffusion des connaissances et des savoir-faire. L'innovation dans un sous-domaine peut rester inconnue dans les autres sous-domaines. Ils invitent à plus d'interactions entre les communautés de DSS et SDSS.

2.2 Contexte d'utilisation

Mapotempo se positionne en tant qu'éditeur de logiciels pour la planification et l'optimisation de tournées pour le dernier kilomètre. Ce positionnement permet, tel que décrit par Crainic *et al.* (2009), de bénéficier de l'interaction entre les véhicules et l'infrastructure avec l'électronique embarquée, les technologies de traçage et le matériel de communication. Il est alors possible d'obtenir, d'une part, des informations sur le réseau routier et les véhicules eux-mêmes et, d'autre de part, de récupérer et de transmettre des informations lors de l'exécution des tournées. Afin de bénéficier de ce positionnement au sein d'un DSS et dans les méthodes de RO, un ensemble d'éléments doit être considéré afin de répondre aux besoins du dernier kilomètre.

La flotte de véhicules peut être hétérogène ; elle est alors constituée de véhicules de natures différentes. Les restrictions applicables à chaque type de véhicule peuvent influer sur les voies que les véhicules peuvent emprunter. Le graphe routier ne sera pas le même pour un poids lourd ou un vélo. Leur nature impacte également la vitesse à laquelle ils peuvent parcourir ce graphe. Une variation de la vitesse peut intervenir selon l'expérience et le comportement du chauffeur sur la route. Le temps passé auprès de chaque client final

peut également varier d'une tournée à l'autre puisqu'elle dépend du type de véhicule. En effet, un poids lourd rencontre plus de difficultés à trouver une place de stationnement qu'une camionnette. Il est également plus long d'ouvrir une remorque et d'en sortir des colis ou palettes. Le temps passé chez un client final dépend aussi du chauffeur et de la politique de qualité de service de la société de transport. Ces paramètres sont généralement connus à l'avance.

Lorsque les tournées sont planifiées, il peut être difficile de les changer entièrement car cela demande une modification organisationnelle qui n'est pas économiquement viable. Par exemple, cela peut demander d'avoir du personnel et des machines disponibles au dépôt pour déplacer les produits auprès du bon quai de chargement. Dans certaines situations, ces changements ne sont pas socialement acceptables. Ajouter de nouvelles demandes à une tournée devient un processus itératif. Et même si des changements sont possibles, ils doivent se faire dans des limites fixes, propres à l'organisation en place (conservation des secteurs, limite au kilométrage supplémentaire). C'est particulièrement le cas lorsque les chargements sont préparés à l'avance pour faciliter le chargement des véhicules dès leur arrivée aux quais de chargements. Il peut également être difficile pour un responsable de renvoyer un chauffeur parce que sa tournée a été répartie sur d'autres tournées. Dès lors qu'une tournée a débuté, il peut être envisagé de transférer des demandes à un autre chauffeur du fait d'évènements sur la route, sauf si les demandes transportées par les véhicules sont propres à chaque client final, tels que des colis de produits commandés sur les plateformes de marchés en ligne.

Les DSS utilisées pour le dernier kilomètre doivent fournir une solution adaptée à la situation actuelle d'une ou plusieurs tournées, tout en proposant de préserver tout ou partie de la solution précédente. Les modules qui opèrent dans ce contexte doivent prendre en charge ces contraintes, états, priorités et préférences. Les outils à la disposition de l'opérateur doivent lui fournir une vision correcte de la réalisation des tournées planifiées. Le responsable de tournées peut avoir à ajuster les tournées et transmettre ces modifications à sa flotte de véhicules sur le terrain.

L'écosystème Mapotempo, tel que représenté dans la figure 2.1 propose deux niveaux de services. Premièrement, un ensemble de *web services* de bas niveaux relatifs au traitement de la donnée (cartographie et optimisation de tournées) dédiés aux ingénieurs informatiques. Deuxièmement, une application web clé en main qui fournit une interface graphique simple.

L'application web est un client léger accessible depuis un navigateur internet. Le choix

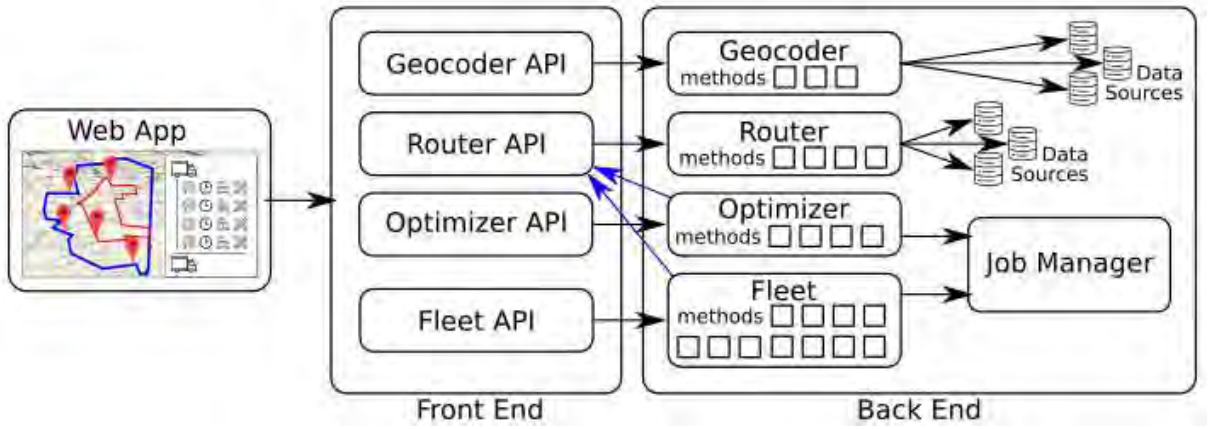

FIGURE 2.1 – Système d'Aide à la décision Mapotempo

d'un client léger a été réalisé pour faciliter la mise à jour des données et le déploiement des nouvelles fonctionnalités auprès des utilisateurs. En effet, les données cartographiques évoluent rapidement pour refléter l'apparition de nouveaux aménagements de la voirie. Maîtriser le déploiement des mises à jour auprès des utilisateurs permet entre autres de réduire les supports, puisqu'il n'existe qu'une seule version à maintenir. De plus, ce choix de client léger permet de réaliser les calculs lourds sur des serveurs et déployés en fonction des besoins. Là où réaliser ces calculs depuis un client lourd demande un investissement supplémentaire des utilisateurs qui doivent alors investir dans un matériel performant. Lorsqu'une action est réalisée sur le client, une requête est transmise au serveur de l'application web et une série de méthodes est déclenchée pour y répondre. Par ailleurs, le client léger permet aux utilisateurs d'utiliser Mapotempo Web depuis un poste fixe ou un smartphone.

Les *web services* sont constitués de deux couches principales. Le *front end* rassemble les méthodes accessibles aux ingénieurs informatiques qui veulent s'interfacer et bénéficier des fonctionnalités proposées par les *web services*. Le *front end* doit rester disponible en permanence pour recevoir les requêtes et y répondre. Il est donc constitué uniquement de méthodes légères qui ont pour visée de déclencher des processus dans le *back end*. Le *back end* traite les données et les opérations lourdes. Traiter ces opérations peut impliquer de déléguer le traitement à d'autres machines, qui sont considérées elles aussi comme faisant partie du *back end*

L'écosystème est compartimenté : chaque fonction principale (Geocoder, Router, Optimizer et Fleet) est autonome et constitue un *web service* indépendant. Cette séparation permet de faire évoluer chaque service indépendamment du reste du projet. Les clients,

dont l'application web est un exemple, s'appuient sur les *web services* et contactent le *front end* dès lors qu'ils ont besoin qu'une action spécifique soit réalisée.

2.3 Définitions

2.3.1 Application Programming Interfaces

Les premiers systèmes de calculs distribués dédiés à l'optimisation datent des années 90. Ils étaient basés sur une architecture client-serveur permettant aux utilisateurs finaux de transmettre leur problème et de recevoir leur solution en utilisant des protocoles tels que FTP (File Transfer Protoco), SMTP (Simple Mail Transfer Procotol) ainsi que des protocoles bas-niveaux TCP-IP (Transfer Control Protocol (TCP) et Internet Protocol (IP)). Les premières générations de serveurs ont rapidement évolué pour bénéficier de la puissance d'expression offerte par les nouveaux protocoles développés pour le World Wide Web (WWW) tel que souligné par Cohen *et al.* (2001). Cette possibilité d'interconnexion dans les modules constituant les modèles de processus métiers (Wang *et al.*, 2018) dans la gestion de la Supply-Chain est basée sur des Application Programming Interfaces (API) qui définissent un ensemble de procédures et méthodes qui peuvent être exécutés par un module au travers d'une connexion internet (Jin *et al.*, 2018; Jacobson *et al.*, 2012; Massé, 2011).

Les technologies Web ont transformé la conception, le développement, l'implémentation et le déploiement des systèmes d'aide à la décision. Ceux-ci peuvent êtres écrits dans n'importe quel langage de programmation qui prend en charge les connexions internet en utilisant TCP-IP. Ces interfaces sont appelées au travers de protocoles internet et permettent de construire des applications en utilisant des services situés sur des serveurs distants. De telles API sont façonnées de manière à fournir des données ou services spécifiques afin de déléguer une partie des traitements d'applications plus larges. Ils donnent accès à des boîtes noires sur lesquels les développeurs peuvent se reposer pour fournir un comportement particulier. La gestion des données et l'accomplissement de taches complexes est souvent délégué à des entités spécialisées qui sinon demanderaient d'avoir un développeur spécialisé ou un accès à des volumes de données pour alimenter de tels systèmes.

Les API sont une manière de mettre en place le paradigme Services-Oriented Architecture (SOA). Elles sont conçues de manière à faciliter l'intégration de sources hétérogènes

ainsi que de rendre des systèmes inter-opérables. L'une des clés pour construire une API est de comprendre comment les requêtes sont traitées côté serveur (modèle de traitement) et comment le client invoque et utilise le service (modèle d'interaction).

Le modèle de traitement d'une API peut être *business object-centric* ou *document-centric* :

— L'approche *business object-centric* est conduite par l'enchaînement d'une série d'appels de méthodes. Ces appels successifs appliquent la logique métier du service sur un ensemble de données détenues par l'API qui sont requises pour le traitement de la requête.

— Une API *document-centric* garde la logique métier séparée du contenu du document transmis. Le service reçoit un document qui contient uniquement des données et aucun lien explicite avec la logique métier à appliquer. Les méthodes liées au traitement de la requête ne sont pas invoquées par le client, elles sont déduites du contenu du document.

De nombreuses études sur la recommandation de structures adéquates aux API lèvent des points sensibles importants qui doivent être discutés dont les points suivants :

— La plupart des approches tiennent compte de l'**isolation** des services étudiés, ce qui a pour tendance de mettre en avant des services avec beaucoup de redondance dans les fonctionnalités proposées plutôt que des services fonctionnellement séparés mais inter-dépendants sur certaines fonctionnalités (Almarimi *et al.*, 2019).

— La **confidentialité** devient plus complexe à mesure que le nombre de services interconnectés augmente. Les réglementations ont également un impact fort sur ce point ; on peut en particulier mentionner la mise en place des règles relatives au RGPD [1].

— Un **accès raisonnable** pour les utilisateurs du service est à définir afin de garantir les performances et une équité entre les utilisateurs dans l'utilisation des services de l'écosystème.

— **Éviter les usages abusifs** de la part d'utilisateurs qui détourneraient l'usage du service de manière à remettre en question la pérennité de celui-ci. En téléphonie mobile, par exemple, téléphoner 20 heures par jour pendant 1 semaine avec un abonnement illimité est un usage abusif.

— Il peut être nécessaire de séparer les traitements afin qu'ils soient réalisés de manière

1. Règlement (UE) 2016/679 du Parlement européen et du Conseil du 27 avril 2016, relatif à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données (règlement général sur la protection des données).

synchrones ou asynchrones. Des techniques spécifiques pour la planification des traitements et la combinaison de services asynchrones ont été fournies par Zhao *et al.* (2012) afin de correspondre aux comportements attendus.

La tendance actuelle pour les *web services* est de s'appuyer sur de multiples services disponibles à travers le réseau. Cela repose sur le paradigme Service-Oriented Computed (SOC) qui permet un développement rapide de systèmes bon marchés, inter-opérables, évolutifs et massivement distribués. La promesse du SOC est d'assembler facilement des micro-services faiblement couplés afin de construire des processus métiers flexibles (Papazoglou *et al.*, 2007; Karakoc et Senkul, 2009). Ces éléments sont accompagnés d'un concept tout aussi important, qui est la qualité de service (QoS). Celui-ci est utilisé pour guider la sélection de services candidats au travers de l'évaluation de multiples mesures (Parejo *et al.*, 2014; Ramírez *et al.*, 2017) : coût, disponibilité, temps de réponse, fiabilité, sécurité, latence, documentation.

Dans l'écosystème Mapotempo, chaque service implémente un ensemble de méthodes exposées à travers son API qui sont associées à un ensemble de ressources. Les ressources correspondent aux concepts principaux autour desquels sont construits les services ou les données. Les méthodes et ressources sont résumées dans la Table 2.1. Les API peuvent communiquer entre elles, en particulier dans le cas de Optimizer et Fleet pour lesquelles le calcul d'itinéraire repose sur Router.

Project	Resource	Create	Read	Update	Delete	Synchronous
Web Application	depot	✓	✓	✓	✓	✓
	unit	✓	✓	✓	✓	✓
	vehicle	✓	✓	✓	✓	✓
	customer	✓	✓	✓	✓	✓
	plan	✓	✓	✓	✓	✓
	plan/tour		✓	✓		✓
	plan/optimize	✓				
Geocoder	geocode		✓			✓
	reverse		✓			✓
Router	route		✓			✓
	matrix		✓			✓
	isoline		✓			✓
Optimizer	solve	✓				
	solution		✓		✓	✓

TABLE 2.1 – Ressources du Front End

2.3.2 OpenStreetMap

Les cartes ont pour but de représenter les territoires, elles sont de puissants outils pour la mise en place de réformes juridiques et fiscales par les gouvernements. En France, les cartes de Cassini (1756-1815) sont la première occurrence de cartes du territoire français et sont le point de départ de la cartographie dite « moderne » (Habert, 2017). Elles instaurent des limites précises dans le découpage administratif, judiciaire ou cadastral du territoire. L'État français garde le monopole sur la production et la diffusion de ces cartes. L'usage est réservé aux élites : militaires, fonctionnaires des impôts, navigateurs ou explorateurs. Historiquement rattaché aux services de l'armée française, la seconde guerre mondiale a forcé le service géographique à devenir un organisme civil, l'Institut national de l'information géographique et forestière (IGN), afin d'éviter que les cartes et matériels de relevés topographique ne tombent aux mains de l'armée allemande occupant alors le territoire.

Du fait d'un manque de concertation avec les populations locales, de nombreux programmes d'aménagement et de gestion du territoire ont échoué, le processus de création des cartes étant initialement confié à des chercheurs ou ingénieurs experts qui imposent une vision du territoire détaché de la vision locale. Dans les années 70-80, l'approche participative a commencé à émerger dans les pays du Sud afin d'intégrer les populations à ce processus de création. Ces collectes de données sont organisées par des organisations de coopération internationale, des ONG ou des organismes gouvernementaux ou internationaux.

L'intégration de contributions venant de personnes qui ne sont pas issues du milieu de la cartographie permet aux communautés locales de représenter et de s'approprier leur territoire, notamment en représentant des savoirs traditionnels et revendications conflictuelles qui ne transparaissaient pas dans les cartographies jusqu'alors (Hirt, 2009).

Depuis lors, les organismes d'État perdent progressivement le contrôle des données géographiques. Le développement de l'imagerie satellitaire, le GPS, les drones et les projets tels que Google Earth ont fait perdre aux États la souveraineté sur ces données. La carte comprise comme un processus politique plutôt qu'un simple outil technique devient alors un contre-pouvoir politique tel que le montre Radjawali (2015).

Cette appropriation par les citoyens de la carte est d'autant plus facilitée par l'émergence de logiciels tels que QGIS, Philcarto, Magrit ou Khartis. Les acteurs privés également prennent avantage de l'aspect collaboratif pour co-construire ou mettre à jour leurs données ; c'est entre autres le cas de la société Waze Ltd.

Le projet OpenStreetMap (OSM) lancé le 9 août 2014 suit cette mouvance de co-construction. Contrairement aux initiatives précédentes menées par les États ou des compagnies privées, les données sont libres. Elles peuvent alors être créées, modifiées et supprimées par chacun.

Au travers de la directive INSPIRE (2007/2/CE) du 14 mars 2007 la communauté Européenne facilite la mutualisation et la diffusion des connaissances. En particulier, cette directive impose aux autorités publiques de rendre accessibles au public les données environnementales géographiques. Ce texte appuie la convergence des intérêts des États, des industriels et du citoyen par la création de ressources communes. Les données agglomérées par chacune des parties viennent compléter les données communes.

Les collectivités migrent progressivement leurs services vers des cartographies OSM. Les industriels qui font face à l'avance de Google et son produit Google Maps trouvent dans OSM une base solide pour construire leur solution (Apple, Facebook, Mapbox) et contribuer à la montée en qualité des données. Les citoyens contribuent et s'approprient le projet, en particulier pour la constitution de versions régionalisées des cartes (`https://www.openstreetmap.bzh/fr/`).

Le tableau 2.2 présente différentes solutions et projets disponibles pour répondre aux problématiques liées à la cartographie, aux données et aux outils qui y sont liés. Les API de données permettent de récupérer des informations relatives à la cartographie. L'IGN a ouvert, au 1er janvier 2021, l'accès sous licence libre à toutes ses données publiques. Les sociétés Google et Here Technologies utilisent des données propriétaires. En effet, elles disposent des équipements nécessaires pour cartographier seules le monde et valider ces informations à l'aide des remontées des utilisateurs qui utilisent leurs solutions.

Il serait trop lourd d'afficher une carte détaillée à l'aide d'une seule image. Ainsi, la carte est divisée en petits morceaux appelés tuiles. Les serveurs de tuile permettent de générer ces tuiles et de les obtenir à la volée lorsque la carte est parcourue. L'IGN, Google et Technologies distribuent des solutions propriétaires. Le communauté OSM a mis en place un certain nombre d'outils libres qui permettent de générer ces tuiles.

Le calculateur d'itinéraire de Google est largement utilisé, cependant il n'est pas adapté pour une utilisation professionnelle. En effet, Google ne fournit pas de solution pour les professionnels, le marché ciblé est le grand public. Ainsi les données agrégées le sont avec pour objectif de fournir un service de navigation aux particuliers utilisant leur voiture. Le calculateur d'itinéraire est alors pleinement utilisable seulement pour les véhicules légers. Les itinéraires proposés ne correspondent pas aux besoins rencontrés avec un poids

Source	IGN	Google	Here Technologies	OpenStreetMap
Données	Libre	Prop.	Prop.	Libre
Serveur de tuile	Prop.	Prop.	Prop.	Libre
Calculateur d'itinéraire		Prop.	Prop.	Libre
Géocodage	Prop.	Prop.	Prop.	Libre
Trafic		Prop.	Prop.	?

TABLE 2.2 – Comparatif des sources de données cartographiques et outils associés

lourd par exemple. Dans le cas d'un piéton, les trajets sont projetés sur les données cartographiés pour les véhicules légers et ne permet donc pas de bénéficier des passages uniquement empruntables à pied.

Les données OSM ne sont agrégées pour aucune utilisation spécifique, même si elles peuvent l'être dans le cadre d'une activité particulière. Les bonnes pratiques invitent les contributeurs à représenter ce qu'est la réalité, plutôt que de les adapter à l'usage. Les données sont collectées à partir de jeux de données ouverts, ou localement par des contributeurs bénévoles ou des entreprises spécialisées. Cette diversité de sources et d'usages augmente les domaines d'applications des données OSM. Les données sont construites selon le modèle suivant :

1. les institutionnels mettent à disposition des données, elles sont intégrées et forment une base,
2. les entreprises vérifient et alimentent les données collectées dans le cadre de leur activité,
3. les contributeurs bénévoles vérifient et alimentent les données selon les projets de la communauté et de leurs centres d'intérêts,
4. des outils de validation remontent les erreurs ou incohérences que les contributeurs de tout niveau peuvent corriger.

Il est possible à tout niveau que des biais soient introduits. Cependant, puisque des contributeurs de tous horizons y interviennent, le consensus final aboutit à des données fidèles à la réalité.

Concernant les données de trafic, il existe des outils propriétaires pour le projet OSM. Par ailleurs, un projet a été financé par la Banque mondiale pour agréger et distribuer ces données sous licence libre. Cependant, le projet est à l'arrêt depuis la fermeture de la société Mapzen qui était chargée de son développement en janvier 2018. L'accès au trafic devrait donc théoriquement être libre, mais dans les faits le projet est en attente.

Ceci est signifié dans la table 2.2 par un point d'interrogation. L'utilisation du projet OSM comme base, permet de couvrir la majorité des besoins rencontrés sur les données cartographiques. De plus, puisque les données et les outils sont librement accessibles, il est plus facile d'en maîtriser les coûts, là où les politiques tarifaires de solutions propriétaires ont montrées qu'elles pouvaient changer rapidement et drastiquement.

Les données OSM sont construites autour de 3 concepts :

Node Les *nodes* représentent des objets qui ont une position géographique.

Way Un *way* est constitué par un ensemble de *nodes* ordonnés qui sont reliés deux à deux par un segment de droite. Cela permet de représenter entre autres des axes routiers, des bâtiments, des cours d'eau, etc.

Relation Les relations sont des collections de *nodes* et *way* qui permettent de définir des liens logiques ou géographiques entre les éléments qui les composent.

Ces concepts sont couplés à une logique de libellés qui permettent de définir à quoi correspondent les éléments représentés.

2.4 Geocoder API

Les adresses postales sont la source principale de géo-référencement des structures humaines. Elles permettent d'identifier par un ensemble de conventions (propres à chaque pays) un point plus ou moins précis du territoire. Elles sont structurées de manière progressive de manière à partir de l'ensemble géographique le plus large vers le plus fin. Les données OSM seules, ne sont pas structurées et ne permettent pas à partir d'un texte comportant une logique géographique complexe de retrouver une position géographique. En France, une adresse postale est généralement formatée comme suit :

Destinataire

Précisions sur l'adresse (Appartement, bureau, etc...)

Numéro de rue, nom de la voie

code postal, VILLE

PAYS

Puisque la donnée est structurée, on peut alors la décomposer et la comparer aux bases de données associées. La Base Adresse Nationale permet de faire le lien entre des adresses

et des coordonnées, processus nommé géocodage. Le géocodage inversé permet quant à lui de convertir des coordonnées en adresse postale.

En Allemagne ou au Royaume-Uni, bien que la structure des adresses soit similaire, la construction du nom de la rue diffère. Là où en France, nous avons le type de voie suivi du nom de celle-ci, outre Rhin, le nom de la voie précède son type. Ceci n'est pas problématique pour la recherche d'adresse lorsque les règles sont propres à des pays entiers.

En revanche, ces distinctions peuvent engendrer quelques problèmes dans des pays appartenant à plusieurs aires linguistiques. Comme par exemple en Suisse qui est couverte par les aires linguistiques de l'allemand, du français, de l'italien et du romanche. La structure n'est alors pas uniforme, il faut alors être vigilant dans la reconnaissance des modèles dans la recherche d'adresses.

La Base Adresse Nationale (BAN) est une base de données recensant la correspondance entre adresse postale et coordonnées géographiques pour environ 25 millions d'adresses sur le territoire français. Elle a été inaugurée le 15 avril 2015 par la signature d'une convention entre 4 entités : L'IGN, le Groupe La Poste, OpenStreetMap France et la mission Etalab. Cette base est intégralement disponible sous Licence Ouverte depuis le 1er janvier 2020. La BAN est maintenue par les pouvoirs publics. Les contributions sont ouvertes mais la validation et la publication passent par les services de l'État. OpenStreetMap a pris l'initiative début 2014 de mettre en place une Base d'Adresses Nationale Ouverte (BANO) qui puisse être interopérable avec le projet OpenStreetMap.

Le projet Addok permet d'interpréter les adresses françaises, de les comparer à une base d'adresses, tel que BAN ou BANO et de restituer les coordonnées géographiques associées.

Les bases d'adresses libres n'étant pas disponibles partout, et chaque système d'adresse demande des compétences spécifiques ou un projet particulier pour géocoder les adresses. Il n'est à l'heure actuelle pas réaliste que Mapotempo internalise l'ensemble de ces compétences. Il est alors nécessaire de faire intervenir des partenaires. C'est en particulier le cas pour les adresses hors France. Geocoder API fait alors appel à d'autres services, dont celui de HERE Technology, et évalue la qualité des résultats obtenus afin de restituer à l'utilisateur le résultat le plus probant.

2.5 Router API

Le calcul d'itinéraire à besoin de trois éléments importants. Dans un premier temps, il est nécessaire d'avoir un graphe routier ou de transport qui soit connexe avec une qualité de données importante. Ensuite, les points de départ et de destination pour les itinéraires doivent correspondre aux besoins de l'utilisateur final. Et enfin, le mode de transport utilisé doit être connu.

Pour le premier point, les données OpenStreetMap sont, comme nous l'avons vu, issues des efforts d'une communauté hétéroclite. Le projet est suffisamment mature pour être utilisé dans les pays occidentaux (Europe et Amérique du Nord) ainsi que dans certaines parties du monde où les communautés locales sont arrivées à une maturité suffisante pour une exploitation industrielle.

Concernant l'obtention des points pour le calcul d'itinéraire, il est certes possible de demander à l'utilisateur final de positionner ces points sur une carte. Cependant, les sources de données sont souvent des adresses postales. Or nous avons vu précédemment qu'il était possible de les convertir en coordonnées parfaitement interprétables à l'aide du mécanisme de géocodage.

Le mode de transport utilisé est une donnée fournie par l'utilisateur.

Le projet Router permet de sélectionner le calculateur d'itinéraire à utiliser en combinant les coordonnées et les modes de transport utilisés. En effet, calculer un itinéraire au travers d'un réseau de transport en commun ne demandera pas les mêmes données qu'un itinéraire en véhicule léger et n'utilisera probablement pas les mêmes technologies. La diversité des modes de transport rend une couverture complète complexe. Il faut alors arbitrer en fonction des usages, si possible, quels sont ceux à internaliser ou ceux pour lesquels utiliser un service tiers est préférable.

Du point de vue de Mapotempo, le besoin principal est d'avoir un calculateur d'itinéraire pour véhicule léger, puisque le secteur d'activité ciblé est la logistique du dernier kilomètre. Les calculs doivent être possibles à l'échelle de l'Europe et proposer une possibilité d'extension en Afrique du Nord et Amérique du Nord. Par ailleurs, le secteur d'activité du service à la personne en France est un secteur historique pour Mapotempo. Pour répondre à cette problématique, nous avons également besoin d'un calculateur d'itinéraires en transport en commun pour les grandes agglomérations françaises.

Open Source Routing Machine (OSRM) [2] est un moteur de calcul d'itinéraire implé-

2. OSRM est un projet libre, distribué sous licence BSD simplifiée. `http://project-osrm.org/`

menté en C++, il permet d'obtenir les plus courts chemins dans un réseau routier. Le réseau routier est extrait des données OSM. Il est possible de modifier ce réseau afin, par exemple, de ne pas emprunter des voies forestières, des autoroutes payantes ou, au contraire, d'intégrer des tronçons restreints aux véhicules de services par exemple. L'un des principaux avantages de OSRM est que le caclul d'itinéraire est très performant. Cependant le graphe routier à charger en mémoire est important et il n'est pas possible de le modifier à la volée. De ce fait, la prise en compte du trafic en temps réel n'est par conséquent pas envisageable avec cet outil. Il est néanmoins possible dans la phase de préparation du réseau routier d'anticiper la congestion résiduelle liée à des zones particulières. En effet, Mapotempo applique sur le réseau routier des règles de vitesse qui dépendent du type de voie et de la densité de population au travers de l'utilisation du jeu de données CORINE Land Cover constitué en France par l'IGN et animé au niveau européen par l'initiative Copernicus (Büttner *et al.*, 2004). Une équivalence de ces données est disponible en Amérique du Nord. Le projet Copernicus couvre également une partie de l'Afrique du Nord.

Pour les transports en commun, nous utilisons le projet OpenTripPlanner (OTP). Les données sont proposées par chaque agglomération de manière individuelle. De manière similaire aux problématiques d'accès à la donnée rencontrée par les adresses postales, les données OSM et les données de densité de population ne sont pas présentes mondialement, du moins pas à un niveau satisfaisant. Il est alors possible de s'appuyer sur les services de partenaires qui possèdent ces données et les services de calcul d'itinéraire, en particulier HERE Technology.

Le projet Router permet donc d'aiguiller soit sur les calculateurs déployés dans le *back end* Mapotempo soit de faire appel aux API de partenaires technologiques.

2.6 Mapotempo Web

Les API permettent la construction de services métiers de haut niveau. Pour qu'elles soient utilisées, il est important de faire la démonstration de leur bon fonctionnement. Mapotempo Web agit à la fois comme application dédiée à la gestion des tournées de véhicules et à démontrer le bon fonctionnement des API utilisées. Cette application propose une interface simple pour visualiser des données sur un fond cartographique, de constituer des tournées, d'obtenir les données (horaires de passage, kilométrage, etc) relatives à celles-ci, de les optimiser puis de les exporter ou de les transmettre pour réalisation.

Mapotempo Web est une application web construite à l'aide de l'architecture Modèle-Vue-Controller (MVC) et qui se présente sous forme d'un client léger accessible à travers un navigateur internet (Des captures d'écran sont proposées en annexe). Cette architecture permet également à Mapotempo Web d'exposer une API pour intégrer certaines de ses opérations dans d'autres services métiers. Ces API permettent de manipuler les données ou appliquer de la logique métier (*business object-centric*). Elles permettent aussi d'obtenir des Vues de certains modules également présents dans l'application web pour les intégrer telles quelles dans d'autres logiciels métiers. L'intégration de Vues dans une interface permet de bénéficier à la fois d'une partie de l'interface interface utilisateur de Mapotempo Web ainsi que toutes les opérations qui y sont liées.

Initialement le *backend* de Mapotempo Web embarquait directement les logiques de calcul d'itinéraire, de géocodage et d'optimisation sans les exposer. Il fallait passer par toute la phase d'import d'un jeu de données et application de logique métier pour avoir accès à ces opérations et obtenir un résultat indirect. Ce format avait plusieurs inconvénients, tout d'abord cela masquait le savoir-faire de Mapotempo sur les outils relatifs à OSM et à l'optimisation de tournées de véhicules. Ensuite, cela représentait une source de revenus complémentaire qui n'était pas exploitée. Enfin, cette architecture obligeait à embarquer sur le même serveur les différents services, ce qui le rendait d'autant plus sensible à des interruptions de service et rendait le service sujet à des latences lors des pics d'affluence.

Depuis, les outils Router, Geocoder et Optimizer exposent leur propre API. Ils sont chacun présents et répliqués sur des machines différentes. Cela a pour conséquence de permettre à Mapotempo Web de monter davantage en charge, les serveurs hébergeant Mapotempo-Web n'ayant plus à effectuer des actions lourdes.

Ce projet intègre un ensemble de concepts qui permettent de représenter un problème de tournées de véhicules. La notation diffère de celle introduite pour le VRP. En effet, il s'agit ici des termes exposés aux utilisateurs et correspondent à leur usage dans le cadre de Mapotempo Web, plutôt que de coller à la dénomination utilisée par la communauté de recherche.

2.6.1 Dépôts

Un dépôt est un point géographique qui peut être utilisé pour représenter les points de départ et de fin des tournées des véhicules. Renseigner une adresse dans cet objet permet de faire appel au géocodage pour obtenir des coordonnées et inversement (Figure A1). Il

est représenté sur la carte par une icône et une couleur au choix.

2.6.2 Unités livrables

Durant les tournées des demandes sont à satisfaire. Les unités livrables permettent de nommer ces entités sans restriction que ce soit du poids, du volume, des arrêts ou tout autre dénomination (Figure A2).

2.6.3 Véhicules

Les véhicules dans Mapotempo Web représentent deux concepts :

— le moyen de transport qui possède des propriétés comme ses capacités (identifiées grâce aux unités livrables), le moyen dont il s'agit (vélo, véhicule léger, poids lourd, etc.) et les restrictions à appliquer pour parcourir le graphe routier (Figure A3),

— le chauffeur associé au moyen de transport et qui définit l'amplitude horaire de travail, les pauses, les coûts d'exploitation (coût fixe, coût horaire, coût kilométrique) et le ou les dépôts (Figure A4).

Un élément vient à l'intersection de ces deux concepts, il s'agit des compétences. Elles peuvent représenter les spécificités du véhicule : « Possède-t-il une chambre froide ? Un hayon ? », mais également celles du chauffeur associé : « Est-il habilité à prendre en charge telle ou telle pathologie ? » dans le cas de tournées de soins à domicile. « Est-il habilité à effectuer des maintenances électriques ? » dans le cas de tournées de techniciens. Les compétences sont liées au système de libellés. Celui-ci associe de manière optionnelle un libellé à une couleur et une icône. Cette fonctionnalité permet soit de distinguer certains points géographiques sur la carte avec une couleur et une icône particulière soit, lorsqu'ils sont affectés à un véhicule, de représenter une compétence (Figure A5).

2.6.4 Destinations et Visites

Les destinations sont similaires aux dépôts dans la mesure où elles ne représentent qu'un point géographique et qu'elles portent uniquement des propriétés visuelles en plus de l'adresse et/ou des coordonnées (Figure A6). Elles ont également quelques propriétés supplémentaires comme le numéro de téléphone et l'adresse mail du client final associé.

Les destinations servent de socle pour le concept de visite. Une destination peut avoir plusieurs visites. Cela permet de représenter les multiples demandes d'un client final et

les contraintes comme les fenêtres de temps. Les propriétés de la destination sont donc communes à toutes les visites d'un même client final (Figure A7).

2.6.5 Plans

Dès lors que les données de bases sont importées dans Mapotempo Web, il est possible de les manipuler à l'aide du client léger. À ce titre, la vue « plan » permet de représenter l'ensemble des données précédemment citées et de donner accès au concept de tournées (Figure A9). Les visites peuvent être affectées à des véhicules puis ordonnées soit manuellement soit de manière automatique. Pour ce point il existe plusieurs manières de procéder :

Import : les visites peuvent être dès l'import affectées à une tournée et ce dans l'ordre d'apparition des visites dans le fichier importé ou dans la requête associée.

Zonage : les destinations appartiennent à une zone pré-définie et les visites sont affectées aux véhicules associées dès l'application du zonage (voir section 2.6.6), mais ordonne les visites en fonction de leur ordre dans la base de donnée, donc sans les ordonner à l'aide d'une résolution d'un problème de voyageur de commerce (TSP).

Optimisation : l'ensemble des propriétés du plan sont transmises à Optimizer API (voir chapitre 3) qui les affecte et les ordonnent de façon à minimiser les coûts d'exploitation et respecter les contraintes renseignées.

La vue plan est le centre de l'application pour l'utilisateur. Elle permet de visualiser son activité et de l'organiser soit manuellement soit en faisant appel à des opérations automatiques. Pour celles-ci l'utilisateur reçoit un retour instantané avec l'affichage des itinéraires et l'actualisation des métriques qui donnent une vue globale de l'impact des modifications sur le temps de travail, la distance, les quantités transportées, le nombre de véhicules utilisés et les coûts d'exploitation qui en découlent. Ces informations sont consultables au niveau macroscopique en résumant l'ensemble des tournées, mais également directement au niveau de chaque tournée. Chaque visite porte également un certain nombre d'informations relatives à la tournée dans laquelle elle est insérée, comme l'horaire de passage ou son indice dans la tournée.

Il est possible d'analyser le détail des tournées sans support cartographique en étendant le panneau des tournées (Figure A10) présent sur le plan. Cette vue permet d'échanger les visites entre les tournées et visualiser directement l'impact des manipulations sur les deux tournées en termes de coût, d'horaires et de kilométrage. Cependant, cela ne permet

pas de constater directement la pertinence des permutations sur un fond cartographique, qui se retrouve cachée par le panneau déployé.

Il est à noter que des indicateurs visuels viennent alerter l'utilisateur lorsque des contraintes sont violées.

2.6.6 Zonages

L'outil de zonage (Figure A8) permet de grouper des points soit manuellement en dessinant des polygones, soit en utilisant des méthodes de clustering [3]. Générer ou construire des zones permet d'affecter des visites à un véhicule. Ceci correspond au concept de *cluster first, route second* présenté dans la section 4.4.2. En effet, un sous-problème est généré ainsi et les tournées peuvent être organisées individuellement. La création de zones s'accompagne d'indicateurs qui peuvent se révéler indispensables dans la création de zones équilibrées. Ces indicateurs permettent de comptabiliser les destinations et visites comprises dans une zone, ainsi que les durées de visites cumulées et le total des quantités comprises dans la zone délimitée. Les indicateurs peuvent être filtrés pour ne prendre en compte que certains points qui possèdent un libellé particulier. Les zones peuvent ensuite être affectées à un véhicule.

2.7 Export des tournées

Dès lors que le responsable de tournées est satisfait de la qualité des données dans Mapotempo Web, de la répartition des visites dans les tournées et de leur ordonnancement, il peut alors exporter ces données pour transmettre les tournées aux chauffeurs ou les conserver pour un usage ultérieur. L'application propose plusieurs modes d'export pour correspondre au besoin de ses utilisateurs. Les besoins basiques sont un export d'un fichier au format tableur, l'impression d'une feuille de route, l'extraction de traces GPS ou l'association des tournées à des applications agenda. Les supports sur lesquels sont exportées les tournées ne permettent pas de récupérer d'information durant leur réalisation pour en effectuer le suivi.

Il existe des solutions sur le marché qui permettent d'avoir un suivi avancé des tournées en suivant le statut des services et le tracé GPS des tournées. Il également possible

3. Le résultat d'une méthode de clustering peut être représentée en constituant un polygone recouvrant l'ensemble des éléments du cluster

d'effectuer le suivi de température à l'intérieur des véhicules afin de vérifier la continuité de la chaîne du froid.

2.7.1 Solutions du commerce

Les solutions de suivi GPS interfacées sont :

— Alyacom
— Notico Deliv'
— Praxedo
— Suivi de Flotte
— Teksat
— TomTom Webfleet
— Trimble

Hormis Notico Deliv' qui passe par une application sur Smartphone, les solutions requièrent un GPS professionnel dédié et un abonnement.

2.7.2 Fleet API et Mapotempo Live

Face à ces solutions qui demandent un matériel particulier ou qui ne sont pas entièrement adaptées au suivi de tournées de véhicules, nous avons choisi de développer notre propre solution. Cela passe par deux projets :

Fleet API reçoit d'un côté des tournées et de l'autre des statuts de missions, des informations liées et des traces GPS. Il synchronise les deux côtés et gère également l'envoi de notifications au client final par le biais de mail ou de SMS.

Mapotempo Live est une application Android (bientôt portée sur iOS) qui permet au chauffeur de récupérer des tournées à effectuer et d'assurer leur suivi, de remonter d'éventuels incidents et de lancer la navigation entre les missions. Actuellement la navigation se fait via une application tierce (Waze, Google Maps, Magic Earth, etc.). Celle-ci sera prochainement disponible directement au sein de l'application.

2.8 Conclusion

Le système d'aide à la décision mis en place par Mapotempo fournit un ensemble d'outils pour répondre aux besoins de la logistique du dernier kilomètre.

Un client léger est accessible et permet de manipuler l'application Mapotempo Web à l'aide d'un navigateur internet. Mapotempo Web propose de visualiser et manipuler des données sur un fond cartographique afin d'organiser des tournées.

L'organisation d'une tournée requiert un certain nombre d'outils. Ces outils sont des *web services* qui sont appelés au travers de l'application de manière transparente pour l'utilisateur. Ces *web services* permettent de transformer des adresses en coordonnées (Geocoder), de calculer des itinéraires (Router), d'appeler des méthodes de Recherche Opérationnelle pour résoudre les problèmes de tournées de véhicules (Optimizer), ou de suivre la réalisation des tournées (Fleet).

Ces *web services* sont chacun découpés en deux parties : 1) un *front end* qui rassemble les méthodes accessibles au travers d'une API pour s'interfacer et bénéficier des fonctionnalités proposées, 2) un *back end* qui traite les données et opérations lourdes invoquées au travers des méthodes présentes dans le *front end*.

Le choix a été fait d'utiliser en priorité des solutions *open source* et des données libres. Ainsi les données cartographiques sont, autant que possible, récupérées au travers des projets liés à OpenStreetMap.

Par ailleurs, les *web services* Geocoder, Router et Optimizer sont distribués en *open source*.

Chapitre 3

Optimizer API

Optimizer-API propose un environnement intégré de résolution pour les problèmes de tournées de véhicules.

Dans ce chapitre, nous détaillons les concepts à prendre en compte pour constituer ce type de *web service*, ainsi que les choix qui ont été effectués. Pour la résolution des problèmes de tournées de véhicules, nous utilisons principalement la programmation par contraintes . Cependant, pour certaines variantes ou pour certaines tailles d'instances, nous utilisons d'autres approches de type métaheuristiques.

3.1 Définition d'une API

L'utilisation d'un service disponible sur un serveur distant implique de définir un protocole pour échanger des données et exécuter des méthodes. Pour cela, nous pouvons nous reposer sur des technologies tel que HTTP qui sont largement diffusées dans l'industrie. Utiliser des principes connus permet aux développeurs de s'approprier plus facilement une API. Une API basée sur HTTP définit un ensemble de *endpoint* qui sont similaires aux URL des serveurs webs. Ces *endpoint* qui représentent des ressources, exposent une ou plusieurs opérations. Une ressource est une fonctionnalité ou un objet du modèle de données. Lorsque les opérations sur les *endpoint* sont invoquées elles permettent d'exécuter des méthodes liées aux ressources. Les données si elles ne sont pas fournies par l'API sont échangées à l'aide de ces opérations en utilisant des formats tels que JSON ou XML.

3.1.1 API Requête/Réponse

Les API de type requête/réponse supposent que le client reçoive une réponse ou qu'une méthode soit exécutée pour chaque requête qu'il effectue.

Pour ce faire, il existe de multiples technologies :

— **Representational State Transfer (REST)** expose un ensemble de *endpoint* en

tant que partie d'URL. REST est basée sur l'utilisation des verbes *Create, Read, Update, Delete* (CRUD) du standard HTTP qui définissent opérations disponibles pour exécuter des méthodes sur les ressources (voir Tableau 3.1). Chaque *endpoint* implémente au moins l'un des verbes CRUD. Les logiques qui sortent de ce patron sont plus difficilement représentables avec une telle structure. Cependant, il est possible d'ajouter aux données transmises avec le verbe *Put* ou *Patch* des paramètres supplémentaires afin d'exécuter des méthodes particulières. Il est également possible de créer des sous-*endpoint* qui représenteront ces méthodes particulières. Un sous-*endpoint* montre la relation avec le *endpoint* de niveau supérieur.

— **Remote Procedure Call (RPC)** similairement à REST utilise les verbes du standard HTTP, mais plutôt que de se focaliser sur les ressources, ce protocole se concentre sur les méthodes qui nomment ainsi les *endpoints*. Ce choix permet de générer un grand ensemble d'actions pour une même ressource mais rend plus difficile la lecture des opérations disponibles.

— **GraphQL** ne nécessite qu'un seul *endpoint* pour manipuler l'intégralité de l'API. Plutôt que de devoir manipuler de multiples *endpoint* et opérations, GraphQL propose de définir toute la logique dans le corps de la requête, généralement au format JSON ou XML. Il ne supporte que les verbes GET et POST. Ce standard rend plus facile la manipulation d'objets complexes et conditionnels.

Table 3.1 – Opérations CRUD et verbes HTTP

CRUD	HTTP verb	Description
Create	POST	Ajoute de nouvelles ressources
Read	GET	Permet de consulter des ressources
Update	PUT/PATCH	Edite ou remplace des ressources existantes
Delete	DELETE	Supprime des ressources existantes

3.1.2 API événementielle

L'architecture événementielle (Event-Driven) est basée sur des états. Un état est un ensemble d'informations conservées suite à un événement, en vue d'une utilisation ultérieure. Un événement est une transition entre deux états. Dans les API événementielles, le serveur conserve l'état courant dans les échanges avec un client et les informations nécessaire pour le contacter directement. Ainsi, chaque changement d'état peut être notifié au client par le serveur. Une architecture requête/réponse ne peut informer le client

directement, elle doit attendre que le client l'interroge pour lui transmettre les changements. L'architecture événementielle permet d'éviter au serveur d'être interrogé si aucun changement de statut n'est à signaler. Cela a également pour conséquence de réduire les échanges entre le client et le serveur, ainsi que le délai entre un changement d'état et la transmission de cette information au client, ce qui est particulièrement intéressant dans le cadre d'applications en temps réel. Ce procédé est notamment adapté aux environnements asynchrones.

WebHooks : Le client transmet au serveur une URL à laquelle il peut transmettre une requête POST HTTP pour le notifier de tout changement. Ce standard permet de basculer d'une API de type requête/réponse à une API évènementielle en ajoutant seulement un *endpoint* HTTP côté client. Les cas d'erreurs sont gérés nativement par le standard HTTP, et lorsqu'ils surviennent le serveur peut retenter de contacter le client après un délai. Malgré sa facilité de mise en place, il peut être difficile de transmettre l'information au client s'il est protégé derrière un pare-feu, puisque dans ce cas de figure, il peut ne pas recevoir de connexion entrante. Cette restriction rend *WebHook* plus adapté pour la communication entre serveurs.

WebSockets ouvre un flux bidirectionnel entre le serveur et le client en utilisant le *Transport Control Protocol* (TCP). La connexion est établie entre la couche applicative et le protocole internet. Puisque *WebSockets* est défini pour fonctionner sur les ports 80 et 443 cela le rend fonctionnel même au travers d'un pare-feu. Ce protocole permet la diffusion rapide de données sur une connexion avec une longue durée de vie. Il est attendu que le client maintienne la connexion ouverte tant que le serveur et lui-même ont des données à échanger.

HTTP Streaming a un comportement similaire à *WebSocket*, mais la connexion s'effectue au travers du protocole HTTP.

3.1.3 Échange de données

Les données transmises à une API doivent rester confidentielles. Pour cela, HTTP implémente le *Transport Layer Security* (TLS) au travers du protocole HTTPS (Figure 3.1). Il sécurise l'envoi de données entre le client expéditeur et le serveur web. En premier lieu, le serveur doit être connu d'une autorité de certification. Pour cela, le serveur génère une clé privée et une clé publique et effectue une demande de *Certification Signing Request* (CSR) auprès d'une autorité de certification pour obtenir un certificat d'identité

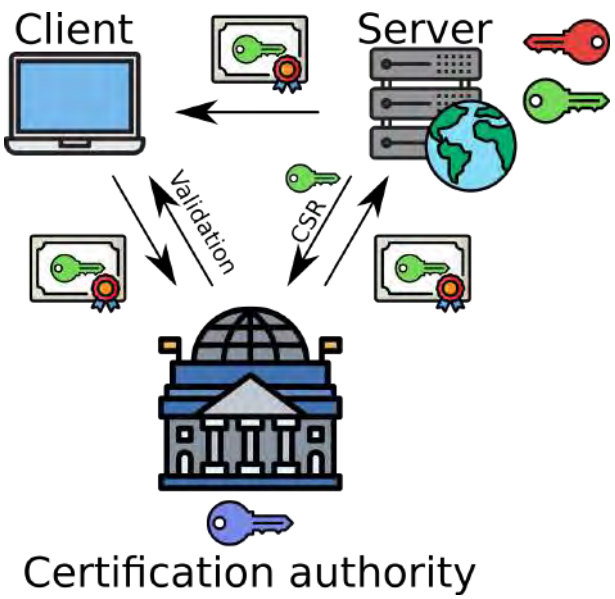

FIGURE 3.1 – Authentification TLS

numérique.

Une fois le serveur enregistré, des requêtes HTTPS peuvent être réalisées. Au début d'une nouvelle transaction le client demande au serveur de lui transmettre une chaîne qui indique auprès de quelle autorité de certification il est enregistré et avec quel certificat. Le client vérifie que l'autorité signifiée fait bien partie de sa liste blanche. Si c'est le cas, le client demande à l'autorité de certification la clé publique correspondant au serveur contacté en premier lieu. L'utilisation d'un tiers de confiance assure que le client et le serveur ont une identité unique vérifiée. À partir de là, les deux parties conviennent d'une clé de cryptage symétrique au travers d'une procédure dite de *handshake* (poignée de main). Puisque les deux parties sont déjà authentifiés et peuvent échanger des données encryptées, la clé symétrique peut être échangée sans risque.

3.1.4 Authentification

Authentification basique

L'une des manières les plus simples d'authentifier un utilisateur consiste à lui demander un identifiant et un mot de passe. Cela ne nécessite aucune libraire ou implémentation supplémentaire, hormis pour le cryptage du mot de passe. La solution est simple et largement diffusée, mais elle a des faiblesses importantes. En effet, lorsqu'une application est compromise, les identifiants sont exposés, ce qui signifie que les données privées de

l'utilisateur peuvent fuiter. Cela devient d'autant plus critique puisqu'il est connu que les mêmes mots de passe sont souvent utilisés au travers de multiples services. Un accès aux identifiants d'un seul service peut donner accès en réalité à de multiples services sur lesquels l'utilisateur est enregistré, même s'ils n'ont rien en commun. Ainsi, cette méthode d'authentification est déconseillée, particulièrement dans le contexte d'applications de gestion professionnelle.

OAuth

Plutôt que d'effectuer l'authentification directement au sein du service contacté, cette partie peut être déléguée à un service tiers, qui peut être une plateforme de confiance. De cette manière l'utilisateur a un accès centralisé à de multiples services avec les mêmes identifiants. L'utilisateur autorise, si la connexion a réussi, le service d'authentification à créer un jeton pour l'identifier. Éventuellement, les applications ou services peuvent demander à avoir accès à certains détails du compte ou à l'identité de l'utilisateur. L'utilisation d'un jeton permet à l'utilisateur de modifier son mot de passe sans perdre accès à tous les services auquel il a souscrit et inversement révoquer l'accès à certains services sans changer de mot de passe. Cette méthode est particulièrement avantageuse puisqu'elle délègue l'authentification à un tiers de confiance, de manière similaire à HTTPS, ce qui découple l'accès aux données personnelles et aux données professionnelles.

Clé d'API

Les clés d'API sont un autre moyen d'être authentifié auprès d'un service. Ces clés sont générées aléatoirement de manière à obtenir une longue chaîne de caractères, la rendant ainsi difficile à deviner même avec une attaque de type *brute-force*. Cette clé a l'avantage d'être unique, elle n'est utilisée que par le service qui l'a générée. Un point, qui peut être vu comme un avantage, est que l'authentification n'est pas déléguée à un service tiers. Cette solution est rapide à mettre en place et assure un bon niveau de sécurité. Elle est plutôt conseillée pour les services « jeunes » jusqu'à atteindre une base d'utilisateurs suffisante pour envisager d'autres méthodes d'authentification. La philosophie est similaire à celle de OAuth puisque l'identification se fait au travers d'une clé ou token unique. Cependant les clés d'API sont moins flexibles dans la mesure où il faut prévoir un autre mode d'authentification pour permettre à l'utilisateur de changer cette clé ou pouvoir rentrer en contact facilement avec celui-ci pour faire la bascule.

3.1.5 Quotas

Le client d'une API peut envoyer de multiples requêtes en simultané. Si elles sont nombreuses, cela peut avoir une conséquence sur les performances du service contacté. Des mécanismes de limitation d'usage doivent être mis en place du côté du service pour empêcher les usages abusifs. Ces mécanismes protègent en partie les API des attaques par déni de service (DoS). Un quota peut être défini sur le serveur indépendamment de l'application et de l'utilisateur. Il est possible de définir un nombre maximal de connections simultanées à la machine physique. La limite peut également être plus fine avec une limite définie par source ; ce mécanisme aura tendance à bloquer les utilisateurs qui dépassent les recommandations. Tout ceci vise à limiter les utilisateurs qui ont un gros impact sur la charge du serveur du fait de leur utilisation trop intensive des requêtes. Par ailleurs, nous devons également considérer les clients qui envoient peu de requêtes mais avec un volume de données important. Les quotas doivent-ils se limiter au nombre de requêtes ou également limiter le volume de données contenus dans ces requêtes ? De telles limitations doivent-elles être identiques pour tous les clients ?

— Le **Token-bucket** fixe un nombre maximal de jetons disponibles, chaque requête entrante consomme un jeton et les jetons sont réapprovisionnés régulièrement. S'il n'y a plus de jetons disponibles, la requête est rejetée.
— Le **Compteur à fenêtre fixe** définit pour une fenêtre de temps le nombre maximal de requêtes qu'il peut recevoir. À la fin de la période, le compteur est réinitialisé.
— Le **Compteur à fenêtre glissante**, plutôt que de considérer une fenêtre statique d'une minute ou d'une heure, considère une définition flottante. En effet, il n'est pas forcément judicieux d'oublier qu'à la fin de la période précédente l'API a reçu une arrivée massive de nouvelles requêtes. Ce mécanisme de fenêtre glissante permet de lisser l'arrivée des requêtes.

3.1.6 Répartition de charge

La saturation d'une API ne provient pas forcément d'un seul utilisateur. Un évènement exceptionnel peut inciter une grande partie des utilisateurs à effectuer des requêtes au travers de leur client simultanément et saturer le service. Pour anticiper et contrer ces cas de charge massive, il peut être nécessaire de répartir la charge sur de multiples serveurs, éventuellement déployés à la demande pour répondre au l'augmentation du trafic. La répartition de charge (*load balancing*) permet de traiter ces évènements et ouvre la voie à

des services soumis à de fortes variations de trafic (Bourke, 2001; Membrey *et al.*, 2012).

Répartition de charge DNS : Lorsqu'un client contacte un service, il utilise généralement un nom de domaine. Pour déterminer l'adresse IP du service associé, la couche réseau interroge un Domain Name Service (DNS). Cependant, héberger le service sur une unique machine peut ne pas suffire pour prendre en charge l'intégralité du trafic entrant. Ainsi, plutôt que de n'exposer qu'une seule adresse IP, il est possible de fournir plusieurs adresses IP au DNS qui s'occupera alors de distribuer équitablement les requêtes sur ces adresses. Cette méthode a plusieurs inconvénients. Premièrement, même si les clients sont répartis équitablement, il n'est pas assuré qu'ils aient un usage équivalent du service. Ceci peut, malgré tout, mener à un déséquilibre dans la charge des différents serveurs. Deuxièmement, la réponse du DNS sera mise en cache par le système ayant effectué la requête, ce qui forcera le client à utiliser la même adresse IP pour une longue période de temps.

Répartition de charge globale : Exposer un service sur internet le rend accessible mondialement. Cependant, son accès est soumis à des contraintes physiques, au design du réseau ou à la latence. Un utilisateur peut avoir une expérience différente selon sa localisation. Avoir un serveur hébergé à Paris, permet aux utilisateurs en France métropolitaine d'éviter les problèmes de latence, mais un utilisateur basé en Amérique du Nord rencontrera une latence élevée, ce qui lui rendra l'utilisation du service plus frustrante. Pour éviter ces inconvénients, le service peut être répliqué dans de multiples data-centers répartis à travers le monde. La redirection des clients vers le serveur le plus proche est également effectuée par DNS. Puisque le DNS est impliqué et qu'il n'a pas accès aux informations détenues par le service, un client pourrait être redirigé vers un serveur qui ne contient pas les données nécessaires à l'utilisateur. Cependant, cette méthode à d'autres avantages. La redondance à travers de multiples data-centers rend le service moins sensible aux interruptions dues aux coupures de courant et défaillances matérielles.

Clustering de serveurs : De manière similaire à la répartition de charge par DNS, le clustering de serveurs promet une haute disponibilité et mise à l'échelle. Mais contrairement à la répartition de la charge au niveau de la couche réseau, sa gestion est effectuée au niveau de la couche applicative. Le clustering de serveurs implique un groupe de serveurs qui accepte le trafic entrant et le réparti entre eux. L'application doit déterminer quels serveurs sont dédiés à quelles tâches. Certains sont dédiés à déterminer le taux de charge, d'autres sont dédiés à la réception du trafic

entrant et d'autres sont dédiés au traitement des tâches du service.

3.1.7 Traitements asynchrones

Dès lors que l'API reçoit une requête, elle doit déterminer les actions à mener. Est-ce que son traitement peut être réalisé directement, ou est-ce que son traitement peut attendre ? Le *front end* du service doit rester disponible en permanence afin de recevoir les nouvelles requêtes. La durée des connexions HTTP est limitée. Si le client est placé sur un téléphone portable, la connexion subira des coupures, en particulier s'il est embarqué dans un véhicule qui se déplace. Pour ces raisons, laisser le serveur qui reçoit les requêtes traiter toutes les opérations peut être dangereux, en particulier lorsque ces opérations sont lourdes. En effet, certaines opérations impliquent un grand volume de données, ou l'appel à des méthodes d'une complexité importante ou encore un temps de traitement important.

À contrario, certaines opérations qui peuvent être traitées rapidement, font appel à des méthodes avec une faible complexité et une faible quantité de données, et pourraient être plus longues à mettre en file d'attente que de les traiter directement.

Par ailleurs, il est possible, selon le processus métier, pour une même requête, de la traiter de manière synchrone ou asynchrone.

3.1.8 Mise en cache

Les opérations réalisées au sein d'un service peuvent prendre du temps. Dans la mesure ou certaines de ces opérations fournissent un résultat déterministe, il peut être intéressant de conserver un résultat afin d'éviter au prochain appel de cette méthode, sur les mêmes données d'avoir à effectuer à nouveau le traitement. Le résultat obtenu peut alors être mis en cache. Certaines opérations peuvent avoir, quant à elles, un résultat qui varie dans le temps. Selon la fréquence de cette variation, le résultat peut être mis en cache ou non. Et si c'est toutefois possible, le cache peut avoir une durée de vie limitée.

3.2 Positionnement et choix techniques

Optimizer API propose une interface unifiée pour la résolution de multiples variantes liées au VRP dans le cadre d'applications réelles. Pour cela, il fait tout d'abord appel à Router API (Section 2.5) pour le calcul de matrices issues du réseau routier. Ne considérer

que les matrices permet de représenter le réseau routier uniquement par la manière dont il est parcouru par les véhicules. Les itinéraires sont ainsi pré-calculés.

Dans le cas de problèmes trop volumineux à traiter, que ce soit par la taille de la matrice à calculer ou par la complexité du problème à résoudre vis à vis du temps alloué à sa résolution, des méthodes de type *cluster first, route second* (voir section 4.4.2) sont proposées et présentées dans la section 4.5. Enfin, différentes heuristiques sont implémentées et plusieurs solveurs sont interfacés afin de résoudre les problèmes soumis. Il a également été choisi de ne pas faire appel au projet Geocoder (Section 2.4). En effet, puisque le géocodage dépend grandement de la qualité des adresses et des données disponibles, ne faire intervenir aucun humain avant d'exécuter une requête d'optimisation peut conduire à résoudre un problème qui ne correspond pas à ce que l'utilisateur imaginait. Un géocodage de mauvaise qualité peut venir soit du fait de l'absence d'adresses dans les jeux de données interrogés, soit parce que l'adresse contient trop d'informations parasites. Le développeur qui souhaite s'interfacer doit, s'il veut tout de même géocoder avant d'optimiser, faire appel aux deux API séparément. Nous recommandons de donner à l'utilisateur accès au résultat du géocodage afin qu'il puisse le valider et le corriger si besoin avant d'effectuer des taches coûteuses en temps et en ressources.

Choisir un standard pour définir une API doit être fait à bon escient selon le contexte d'application. Nous avons vu dans la section 3.1.2 que les API événementielles sont plutôt utilisées dans le cadre de connexions stables tout au long de l'échange de données. Elles sont adaptées à des données légères avec un flux régulier ou continu. Dans le contexte de *web services* dédiés aux tournées de véhicules, l'API peut être utilisée sur le terrain, nous n'avons alors aucune garantie sur la qualité du réseau. Par ailleurs, les traitements se font de manière asynchrone. Le résultat est volumineux et n'est disponible qu'à la fin du traitement. Dans ces conditions, le maintien d'une connexion ouverte tout au long du processus n'a pas été retenu.

Le choix se porte alors plutôt sur les API de type requête/réponse (Section 3.1.1). Le stockage des données étant délégué au *back end* de Mapotempo Web ou au service appelant l'API, nous pouvons supposer un problème envoyé complet. Optimizer API n'a par ailleurs pas vocation à stocker de données. Le problème contient toutes les informations nécessaires à son traitement, nous n'avons besoin que de deux opérations élémentaires pour interroger le service et récupérer le résultat. Une API RPC n'est alors pas requise puisque le nombre d'actions à appeler est réduit. GraphQL permet l'appel et la modification d'éléments précis dans une base de données, puisque le projet Optimizer-API ne stocke ni

ne partage d'information sur les VRP traités, ce choix semble peu pertinent. L'architecture REST est simple, largement utilisée et convient à l'usage souhaité pour cette API. Il est à noter que REST, n'est pas un protocole, mais une convention d'architecture. Ce qui signifie que les règles qui le définissent peuvent ne pas être appliquées dans tous les *web services* qui se définissent comme suivant la convention REST. Le terme Restful, permet de qualifier les services qui se conforment à l'ensemble des principes énoncés. Les API Restful se concentrent sur la performance, mise à l'échelle, simplicité, modifiabilité, visibilité, portabilité et fiabilité :

- **Client-Server** : La responsabilité doit être séparé entre les différents domaines organisationnels. Dans notre cas d'utilisation, le serveur reçoit les requêtes et les stocke dans une file. Les *workers*[1] interroge celle-ci pour obtenir un problème à traiter.
- **Statelessness** : Une requête est auto-suffisante et ne requiert pas de conserver l'état de la session. Ici, le client envoi une requête complète et reçoit un identifiant en échange. Utiliser cet identifiant permet au client de vérifier le statut de la résolution indépendamment de la session en cours.
- **Cacheability** : Les réponses peuvent être mises en cache. Le résultat obtenu à partir d'un VRP pourrait être retourné sans avoir à effectuer à nouveau la résolution.
- **Layered system** : Un serveur peut déléguer une tâche à un autre serveur pour générer une réponse. Optimizer-API est organisé de telle façon que la résolution est déléguée à des *workers* qui pourraient tout à fait être placés sur une autre machine physique. Ce point aide au passage à l'échelle.
- **Code on demand (optional)** : Les serveurs doivent être capable de transmettre du code exécutable. Le projet en l'état n'implémente pas ce type de fonctionnalité.
- **Uniform interface** : Le modèle de données interne est conceptuellement découplé de la réponse envoyée au client, la réponse doit être auto-suffisante. Également, la présentation d'une ressource doit fournir suffisamment d'informations pour permettre au client de la modifier ou la supprimer. Dans notre cas, le modèle est indépendant de la réponse renvoyée en JSON, de manière à ce que la solution puisse être interprétée seule. Un problème soumis peut être supprimé à l'aide de l'identifiant retournée par l'API.

1. Les workers sont des processus qui traitent des tâches de manière asynchrone. Ils déploient un code similaire au serveur exposant l'API mais ne sont pas accessibles depuis l'extérieur du système.

Le choix d'un modèle *business-centric* ou *document-centric* pour un tel service n'est pas restreint par le standard utilisé mais aux besoins du projet lui-même. L'utilisation de REST permet d'utiliser les deux approches. Les opérations peuvent chacune correspondre à une définition *business-centric*, de manière à ce que le client doive appeler les bonnes méthodes pour traiter les données à sa convenance. Il est aussi possible de transmettre un document à l'aide du verbe POST avec toutes les données nécessaires et laisser l'API déduire les opérations internes à appliquer. Il est également possible de combiner les approches afin d'avoir des opérations dépendantes des requêtes du client et d'autres indépendantes.

Cette propriété est particulièrement intéressante dans notre contexte. En effet, nous allons recevoir de grands et complexes VRP à résoudre. L'objectif de l'API est de fournir un service de type boîte noire qui retourne des tournées organisées. Il n'est pas attendu que l'utilisateur sache quelle suite d'opérations lui permette de résoudre son problème. Le projet Optimizer-API détermine la série d'opérations à appliquer pour chaque problème reçu (*document-centric*). Néanmoins, l'utilisateur pourrait être amené à manipuler des opérations de haut niveau.

Le calcul des matrices et la résolution d'un VRP sont coûteux en temps, à ce titre la résolution asynchrone est privilégiée. Dans ce cas de figure, l'utilisateur pourrait vouloir interagir avec le processus traitant son problème avec des opérations *business-centric*. Ici, il peut interroger régulièrement le service pour déterminer le statut de la résolution. Puisque la logistique du dernier kilomètre intervient dans un environnement qui peut évoluer rapidement, un problème donné à résoudre peut ne plus correspondre au besoin une fois terminé. L'utilisateur peut alors demander à interrompre la résolution. Il peut également vouloir lister l'ensemble des problèmes dont le traitement est en cours. Ces trois opérations sont *business-centric* puisqu'elles n'attendent pas de l'API l'invocation de logique complexe et cachée, mais d'opérations simples que l'utilisateur invoque en fonction de son processus métier.

Le scénario d'utilisation est alors le suivant :

1. Le client transmet un problème à l'API au travers du *endpoint* **submit** et du verbe POST.
2. L'API retourne un identifiant unique en échange de ce problème.
3. Cet identifiant permet, grâce au *endpoint* **solutions** et au verbe GET, de vérifier le statut de la résolution et lorsqu'elle est terminée de récupérer le résultat.
4. L'identifiant permet également d'interrompre la résolution à l'aide du *endpoint*

solutions et du verbe DELETE, si l'utilisateur n'en a plus l'utilité.

Il n'existe actuellement aucun mécanisme de notification pour alerter le client d'un changement de statut. En effet, avant d'envisager cette fonctionnalité, certaines interrogations doivent être levées. Nous imaginons plutôt l'API utilisée par un responsable des tournées depuis un client web comme c'est le cas pour Mapotempo Web. Dans ce cas, les mécanismes de notifications seraient parfaitement fonctionnels. Cependant, si l'API est appelée durant une tournée par le matériel embarqué d'un véhicule, alors son adresse IP pourrait être amenée à changer. Ce problème peut être gommé en passant par un service intermédiaire qui s'occupe de gérer les transactions à sa place, tel que Fleet API. Un autre scénario envisageable est d'avoir un responsable des tournées qui utilise un client web depuis son poste, mais voudrait que le résultat soit envoyé directement sur le matériel embarqué d'un véhicule en cours de tournée. La question est ouverte, intéressante, mais nous n'avons pas eu à ce jour de demande en ce sens.

La sécurité des échanges de données, entre le serveur et le client, est assurée par l'utilisation de TLS au travers du protocole HTTPS. Ce choix implique l'augmentation du nombre de connexions au serveur dans la mesure où la procédure de *handshake* demande plusieurs aller-retours. Cependant, ces échanges sont légers et sont un moindre inconvénient au regard des bénéfices apportés. Le cœur de cible du projet sont les éditeurs de logiciels métiers. À cet égard, l'authentification basique semble peu pertinente dans la mesure où le risque d'exposition de données confidentielles est important. Le protocole OAuth est intéressant, mais demande soit de passer par un tiers de confiance, soit d'avoir en interne un tel système déjà établi au travers des différents projets. Nous ne sommes pas encore arrivés au niveau de maturité où la mise en place de cette solution est nécessaire. En revanche, les clés d'API correspondent au niveau de sécurité attendu tout étant simples à mettre en place. C'est la solution retenue. Les clés d'API s'accompagnent de limites sur la taille des problèmes que peuvent transmettre les entités [2] et utilisateurs associés. Comme énoncé dans la section 3.1.5, la taille d'une seule requête n'est pas le facteur limitant, le serveur est limité en nombre de connexion entrantes. Nous avons mis en place des compteurs à fenêtre glissante afin de limiter les requêtes sur plusieurs échelles de temps (jour, mois, année). L'ensemble de ces limites peuvent varier selon le *web service* ou l'opération demandée.

Nous n'avons à ce jour pas mis en place de mécanisme d'équilibrage de charge. En

2. Par entité nous entendons des structures tels que des entreprises, associations ou équipe de recherches qui peuvent avoir de multiples utilisateurs qui partagent les mêmes limites

effet, les éditeurs, pour certains, ont fait le choix d'avoir leur propre instance du service, le flux d'entrée est réparti sur plusieurs serveurs. Pour la partie propre aux utilisateurs provenant de Mapotempo-Web ou utilisant ce même serveur, dit mutualisé, le nombre de connexions entrantes simultanées n'a pas provoqué de congestion. Il pourra être envisagé dans un premier temps une répartition de la charge en répartissant les clients par leur origine géographique par DNS. Ensuite, puisque le service ne retient aucune donnée une répartition de la charge par DNS pourra être mise en place.

À la vue du nombre de connexions entrantes, de la complexité des taches à effectuer et de la taille des instances, une partie des requêtes d'optimisation est traitée de manière synchrone. La limite actuelle est fixée à 200 points, si le véhicule lié est un véhicule léger (puisque nous maîtrisons le service pour le calcul de ces matrices) et si le solver appelé est VROOM. En effet, VROOM nous permet d'assurer pour les instances cette taille de retourner une solution dans un délai inférieur à 100ms pour le TSP et le CVRP. Pour la plupart, les requêtes sont traitées de manière asynchrone, si elles ne correspondent pas aux cas particuliers permettant de déclencher un traitement synchrone. En effet, dès lors que nous estimons que le calcul de matrice prendra trop de temps ou que la résolution sera trop longue, nous libérons le serveur qui expose l'API pour accepter de nouvelles requêtes entrantes. Le serveur exposant l'API et les *workers* sont actuellement placés sur la même machine physique. Cependant, nous avons commencé à mettre en place des mécanismes correspondant au clustering de serveurs à l'aide de technologies liées à Docker (Modak *et al.*, 2018).

3.3 Modèle de données

Le VRP est composé de différents éléments : nous avons d'une part les véhicules et d'autre part un ensemble de clients finaux qui expriment des demandes qui sont représentés par des nœuds à des points géographiques donnés. Les concepts exprimés ici font écho aux concepts évoqués dans la section 1.5. Le nombre de nœuds qu'un véhicule peut servir est limité par des capacités correspondant aux demandes. Les chauffeurs associés aux véhicules ont une amplitude de travail limitée et des pauses à prendre durant leur temps de travail. Les nœuds ont des fenêtres de temps. À partir de cette description succincte nous pouvons déjà dégager une structure qui lie les différents concepts entre eux :

— les **Fenêtres de temps** délimitent un créneau horaire avec un début et une fin ;
— les **Points** représentent une position sur un graphe, soit en utilisant une position

composée des coordonnées, soit un indice si une matrice est fournie ;
— les **Unités** permettent de désigner une unité de mesure, par exemple, un poids ou un nombre de palettes.

Ensuite, nous avons des concepts qui ne sont pas auto-suffisants :
— les **Capacités et Quantités** sont liées à une unité. Les quantités correspondent aux demandes. La capacité permet de donner une limite aux quantités transportées ;
— les **Pauses** représentent une interruption du temps de travail, avec une fenêtre de temps et une durée ;
— les **Matrices** représentent les distances (temporelle, kilométrique ou de valeur) entre les points où sont localisés les nœuds qui composent le problème. En effet, nous n'avons pas l'utilité de représenter l'intégralité du graphe ici, nous pouvons pré-calculer ces distances ;
— les **Véhicules** parcourent le graphe afin de servir les clients finaux et satisfaire leurs demandes. Un véhicule comporte une fenêtre de temps, il référence un ensemble de matrices (temps, kilométrique et de valeur) ou possède les propriétés nécessaires à leur calcul. Il peut également avoir un ensemble de capacités et possède des points de départ et de fin pour représenter ses dépôts. Il embarque également des pauses ;
— les **Activités** représentent chacune un nœud, elles sont composées d'un point, d'une durée pour servir la demande et de fenêtres de temps,
— les **Services** sont composés chacun d'une ou plusieurs activités. Ces objets portent également les demandes associées aux nœuds qui les composent.

Les véhicules et les services se retrouvent alors liés par différentes dimensions comme le temps par le biais des fenêtres et des matrices de temps, ou par les dimensions matérialisées par les unités. Des éléments supplémentaires viennent ajouter des attributs au VRP ou simplifier la définition de ceux-ci par les développeurs s'interfaçant avec l'API :
— les **Routes** permettent à l'utilisateur de transmettre l'état initial des tournées à partir duquel la résolution doit commencer ;
— les **Shipments** facilitent la définition des Pickup and Delivery (PDVRP) ;
— les **Relations** donnent la possibilité de lier des missions (services ou shipments) ou des véhicules ;
— les **Zones** lient un secteur défini par un polygone à un véhicule ou plusieurs véhicules.

Une fois ce modèle défini, nous devons nous assurer que les données qui sont transmises

en entrée à l'API respectent sa structure.

3.3.1 Validation du contenu

Les différentes éléments du modèle ont leur propre cohérence, chaque champ qui les compose doit être validé individuellement afin de valider son type et éventuellement son domaine. Les champs relatifs au temps par exemple peuvent être fournis selon différents formats, en secondes à partir d'un zéro particulier, en format Heure-Minute ou Heure-Minute-Seconde. Ces différents formats sont courants et nous pouvons les tolérer puisque leur conversion vers un nombre entier est possible. Il est également possible de vérifier que les champs composant un élément sont cohérents localement. En effet, pour un point, nous attendons au moins une position ou un indice, un point sera inutilisable s'il ne possède pas au moins l'une de ces informations. Par ailleurs, certains champs peuvent être optionnels, comme la fenêtre de temps pour un service, ou obligatoires comme l'unité d'une quantité. Si le contenu du problème transmis ne peut être validé l'ensemble des éléments en erreur sont remontés à l'utilisateur et le problème est rejeté.

3.3.2 Validation de la structure

Dès lors que nous avons validé les éléments individuellement, nous pouvons vérifier qu'ils composent bien un problème dans lequel ils sont correctement reliés. Par exemple, si les points référencés par les véhicules ou les services n'existent pas, alors nous ne pouvons construire le problème dans son intégralité. Le problème est identique pour l'utilisation des unités par les capacités et les quantités. Un problème dont la structure est incorrecte est rejeté et indique à l'utilisateur l'élément qui a provoqué ce rejet.

3.3.3 Validation de la cohérence générale

Un problème transmis peut avoir une structure correcte, les éléments qui le composent peuvent être validés et être cohérents localement. Cependant, ce même problème peut n'avoir aucun sens au global. Les fenêtres de temps associées aux véhicules peuvent être disjointes des fenêtres de temps liées aux activités. Nous pouvons déterminer même sans appeler une procédure de résolution que le résultat sera composé de routes vides. Il en va de même pour des quantités qui dépasseraient les capacités des véhicules. Les éléments qui provoquent ce manque de cohérence générale ne sont pas forcément simples à identifier

et n'empêchent pas la construction du modèle. Il est cependant intéressant de les filtrer, de manière à éviter aux méthodes de résolution de tenter d'insérer des éléments qui sont détectés en amont comme incohérents.

3.3.4 Réalisabilité

Lorsque le modèle est construit et validé, aussi bien au niveau local qu'au niveau de sa structure. Sa cohérence générale a été vérifiée et au besoin des éléments ont été filtrés. Nous pouvons alors déterminer si le problème décrit peut être résolu au sein de l'API. Le projet intègre différents solveurs, nous avons pour chacun une liste d'attributs ou de combinaisons de ceux qu'il peut traiter. Si aucun solveur n'est éligible pour le problème, ce dernier est rejeté et l'API renvoi pour chaque solveur quelle combinaison d'attributs a provoqué ce rejet. En revanche, si un solveur est éligible nous déterminons si le problème peut être traité de manière synchrone ou asynchrone.

3.4 Rapport d'erreurs

La prise en charge d'un grand nombre d'attributs augmente le nombre de combinaisons possibles dans l'utilisation de ceux-ci. Ainsi, il est humainement difficile d'anticiper chacune de ces combinaisons. L'utilisation de données réelles et leur projection sur un réseau routier sont également sujets à des erreurs, que ce soit des erreurs de saisies (inversion des latitudes ou longitudes), des erreurs liées à leur projection (un point projeté sur un graphe routier non connexe) ou la cohérence vis à vis du problème (incompatibilité d'un créneau horaire avec les disponibilités véhicules). Bien que la plupart de ces éléments soient filtrés, il n'est pas exclu qu'une combinaison de ceux-ci génère un blocage au sein de l'API. Il est alors nécessaire d'avoir un suivi des erreurs qui sont rencontrées par les utilisateurs. D'une part, afin de pouvoir les suivre et les comptabiliser, d'autre part, pour assurer un suivi client qui soit pro-actif. À cet usage, le projet Sentry (Figure 3.2) est utilisé. Il permet en cas d'alerte ou d'erreur de générer un rapport détaillant le contexte dans lequel l'erreur a été rencontrée :

1. fichier et ligne de l'erreur,
2. l'ensemble des méthodes de plus haut niveau ayant mené à celle-ci,
3. informations relatives aux données manipulées.

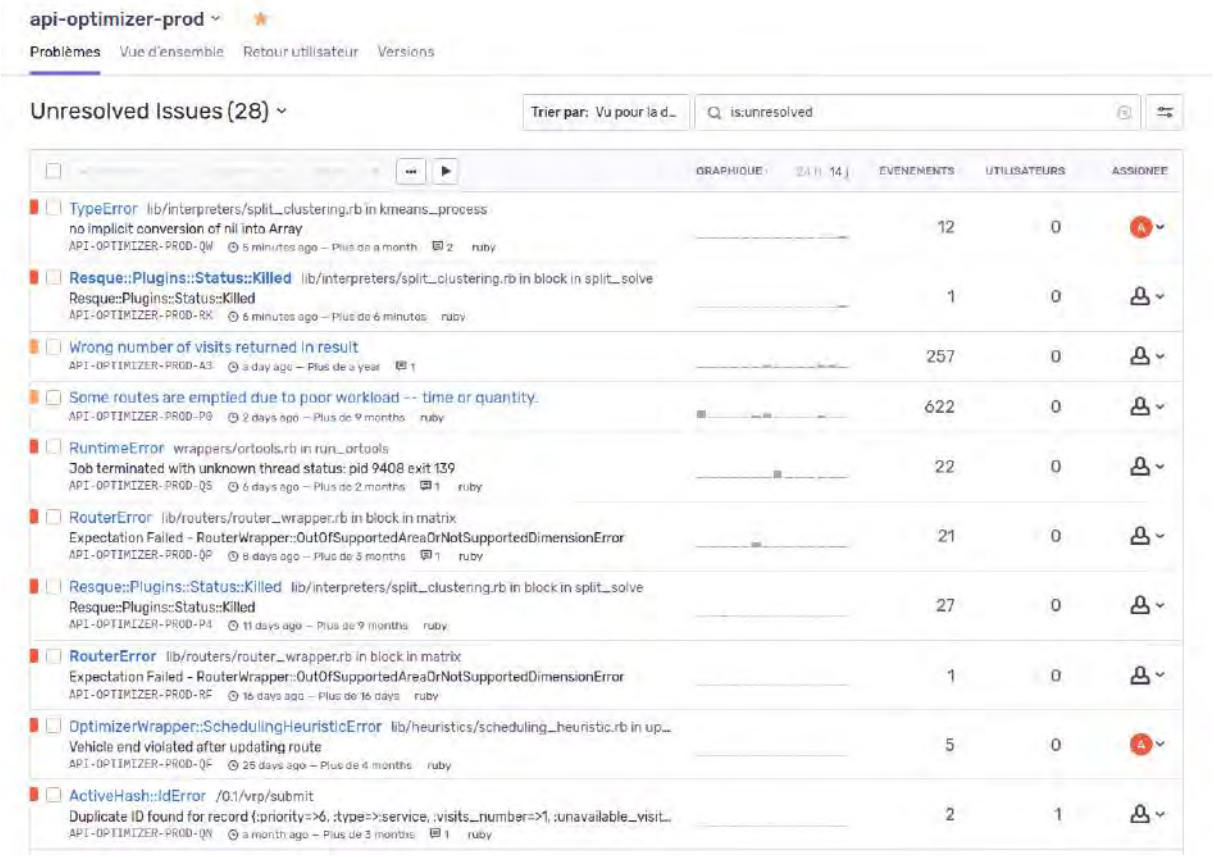

FIGURE 3.2 – Sentry

3.5 Programmation par contraintes

D'après Rossi *et al.* (2006), « les contraintes sont un concept omniprésent qui, dans son sens le plus large, se rapporte à notre expérience quotidienne : elles représentent les conditions qui réduisent notre liberté de décision ». C'est donc une notion naturelle à manipuler, elle trouve son application directe pour la résolution de problèmes combinatoires au travers du paradigme de programmation par contraintes (PPC). Ici, les contraintes permettent de réduire de l'espace de recherche, là où en programmation linéaire des coupes sont ajoutées dès lors qu'une contrainte est violée et fournit une solution irréalisable.

Cette propriété est particulièrement intéressante dans la mesure où toutes les solutions fournies par la PPC sont réalisables. De plus, la réduction de l'espace de recherche permet une résolution plus rapide. Donc, plus un problème est contraint, plus l'optimalité sera atteinte facilement.

En programmation linéaire (PL), l'espace des solutions réalisables peut être repré-

senté par un polyèdre et la résolution passe sans distinction de l'intérieur de ce polyèdre (faisabilité) à son extérieur (infaisabilité) jusqu'à trouver l'optimalité. Il peut donc être complexe en PL d'atteindre des solutions réalisables.

Ainsi dans le cadre d'un service d'aide à la décision pour le dernier kilomètre, il est pertinent de s'orienter vers un paradigme qui puisse fournir des solutions réalisables rapidement.

De plus, la PPC permet de définir un état initial pour les variables intervenant dans la solution. Cette initialisation permet soit d'utiliser d'autres méthodes pour fournir une solution de départ, soit d'utiliser la solution d'un problème légèrement différent. Par exemple, Bourreau *et al.* (2020) utilisent une une solution obtenue par un TSP et découpé à l'aide de l'algorithme Split (Lacomme *et al.*, 2001) pour initialiser un VRP avec synchronisation.

Un problème traité en programmation par contraintes est composé des éléments suivants :

Variables

Un problème est défini par un ensemble de variables de décision $\mathcal{X}$.

$$\mathcal{X} = \{x_1, ..., x_n\}$$

Domaines

Soit $\mathcal{D}$ l'ensemble des domaines des variables. Un domaine est l'ensemble des valeurs que peut prendre une variable.

$$\mathcal{D} = \{\mathcal{D}_1, ..., \mathcal{D}_n\},\ \forall k \in [1; n]\ x_k \in \mathcal{D}_k$$

Contraintes

L'ensemble des contraintes $\mathcal{C}$ énoncent des propriétés qui doivent être respectées.

$$\mathcal{C} = \{\mathcal{C}_1, ..., \mathcal{C}_m\}$$

Les contraintes créent des relations entre variables, elles peuvent être de type arithmétique $x < y$, logique $x \Rightarrow y$ ou plus complexe pour lier davantage de variables.

Les contraintes permettent de représenter le problème sous forme de graphe.

Un nœud du graphe représente alors une variable. Une contrainte peut s'appliquer uniquement à un nœud. De cette manière son domaine est réduit. La *node consistency* assure que toutes les valeurs du domaine d'un nœud satisfont toutes les contraintes unaires (à un opérande).

Il est possible de relier les nœuds par des arcs, on ne relie alors pas les variables, mais les valeurs formant les domaines de ces variables. Un arc représente alors un couple de valeurs qui permet de respecter la contrainte. Une variable x est dite *arc consistent* lorsque dans une contrainte $\mathcal{C}$ qui lie deux variables x et y s'il existe une valeur dans le domaine de y tel que la contrainte binaire $\mathcal{C}(x, y)$ est satisfaite. La contrainte est *arc consistent* s'il existe une valeur dans les domaines de x et y.

Cette notion est illustrée à l'aide de l'exemple extrait du cours « Programmation par Contraintes » de David Savourey. Il s'agit d'un problème de coloration, quatre pièces (P1, P2, P3 et P4) sont à colorier et il existe trois couleurs (bleu, jaune et rouge). Dans la figure 3.3 chaque pièce, ou variable est représentée par une ellipse. Le domaine de ces variables est représenté par les couleurs qui y sont contenues.

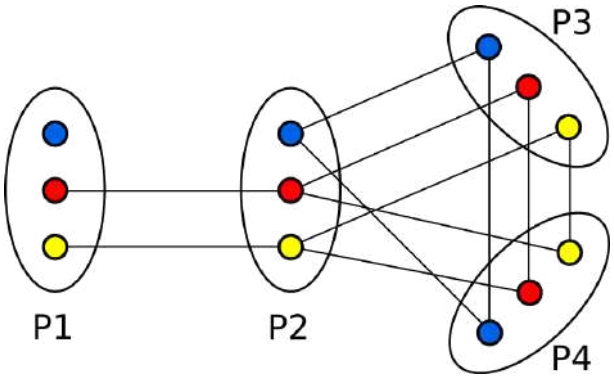

FIGURE 3.3 – Problème de coloration : graphe de compatibilité des couleurs

Les arcs du graphe représentent les choix de couleurs compatibles. Nous pouvons constater que la couleur jaune de P1 est reliée par un arc à la couleur jaune de P2 et la couleur rouge de P1 est reliée à la couleur rouge de P2. Cependant la couleur bleue de P1 n'est reliée à aucune couleur de P2 et d'aucune autre variable. Alors affecter la couleur bleue à P1 ne permet pas d'affecter une couleur compatible à P2. La couleur bleue peut alors être supprimé du domaine de P1.

La *path consistency* considère un ensemble de paires de variables. Une paire de variables (x, y) est *path consistent* avec une troisième variable z si pour chaque valeur *arc*

consistent pour la contrainte binaire $\mathcal{C}(x, y)$ il existe également une valeur pour z telle que les contraintes binaires $\mathcal{C}(x, z)$ et $\mathcal{C}(y, z)$ sont satisfaites et assurent l'*arc consistency*.

Les mécanismes et algorithmes de réduction des domaines sont appelés filtrage. À chaque réduction sur une variable, il convient d'évaluer à nouveau l'ensemble des contraintes dans laquelle elle est impliquée, puisqu'elle pourrait mener à de nouvelles déductions, on parle alors de propagations.

Pour revenir à l'exemple de coloration, nous avons supprimé la couleur bleue de P1. La couleur bleue de P2 n'est reliée à aucune couleur de P1. Elle peut également être supprimée de P2. Cependant, la couleur bleue de P2 est reliée aux couleurs bleues de P2 et P3, qui sont elles-mêmes reliées entre elles. Ainsi, si la couleur bleue de P2 est supprimée, nous pouvons propager cette information. En effet, si nous supprimons les arcs reliants les couleurs bleues de P2 et P3, puis P2 et P4, alors les couleurs bleues de P3 et P4 ne sont plus reliées à aucune couleur de P2. Ainsi, les couleurs bleues de P3 et P4 peuvent être supprimées. Ceci amène au graphe simplifié présenté dans la figure 3.4.

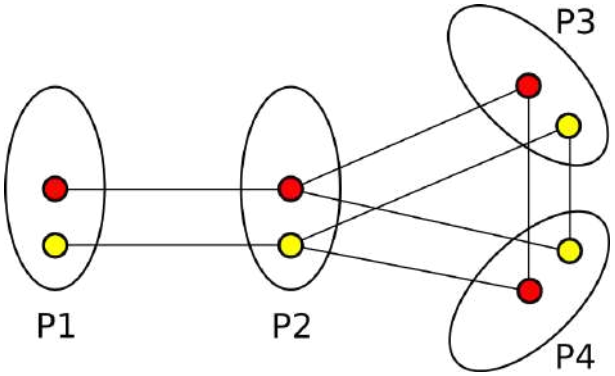

FIGURE 3.4 – Problème de coloration : graphe « simplifié » de compatibilité des couleurs.

Il est possible de faire intervenir plus que deux variables dans une contrainte. Il est ainsi possible de généraliser l'*arc consistency*. Ceci permet alors d'introduire les contraintes globales (Beldiceanu *et al.*, 2010) qui permettent d'appliquer une contrainte sur un ensemble de variables. Les contraintes si elles lient deux variables permettent dans le graphe des contraintes de propager des réductions des domaines entre les deux variables impliquées. L'ensemble des contraintes définies par couple de variable doivent alors être parcourues une à une à chaque réduction de domaine. C'est alors que les contraintes globales ont un intérêt. En effet, plutôt que de définir un ensemble de contraintes pour lier plusieurs variables, les contraintes globales permettent de définir une seule contrainte pour les lier toutes. Ce lien permet de mettre en place des algorithmes de filtrage et de propagation

efficaces.

La contrainte globale AllDifferent permet pour un ensemble de variables $\mathcal{X}$ d'assurer que toutes leurs valeurs soient différentes. Il est, certes, possible de définir pour chaque couple de variables une contrainte disant que leurs valeurs sont différentes. Cependant, la définition est plus simple avec la contrainte globale AllDifferent, mais également ces contraintes globales permettent d'appliquer des algorithmes de filtrage bien plus efficaces que pour des couples de variables (Gent *et al.*, 2008).

Selon les solveurs de PPC la liste des contraintes globales peut varier. La majorité d'entre eux implémentes les contraintes globales suivantes :

AllDifferent toutes les variables doivent avoir une valeur différente,

Sum la somme des valeurs des variables doit être égale à une valeur donnée,

Scalar le produit scalaire valeurs des variables doit être égal une valeur donnée,

Element cette contrainte permet de lier la valeur d'une variable aux valeurs présentes dans un tableau.

Cumulative pour chaque intervalle de temps t le cumul des demandes ne peut dépasser la capacité C définie.

L'espace de recherche déduit de la propagation initiale des contraintes est parcouru en affectant successivement une valeur aux variables. À chaque affectation, les mécanismes de filtrage et de propagations sont appliqués. Si une affectation mène à rendre vide le domaine d'une variable, un échec survient, le choix est remis en cause grâce à un mécanisme de *backtrack* (retour arrière) déclenché et une affectation différente est appliquée.

Progressivement toutes les variables sont affectées et une solution est constituée dès lors que toutes les variables ont une valeur fixée. Le mécanisme de *backtrack* est sensible à la stratégie de branchement. En effet, il est possible avec la PPC de définir une politique d'exploration de l'arbre qui prend en compte les spécificités du problème traité. Ce point est mis en évidence par Bourreau *et al.* (2020) au travers de l'exemple du VRP que l'utilisation de contraintes globales associées à une stratégie de branchement adaptée permet de réduire considérablement le nombre de nœuds de l'arbre de décision exploré ainsi que le nombre de *backtrack* nécessaire pour obtenir la preuve d'optimalité et donc d'accélérer le temps de résolution.

Il est également possible d'appliquer des métaheuristiques à cette recherche de solution. Les métaheuristiques bénéficient alors du cadre de la PPC pour valider la faisabilité des solutions explorées par recherche locale. Elles sont initialisées à l'aide d'une solution

obtenue aléatoirement ou à l'aide d'une heuristique. Celle-ci peut être irréalisable ou incomplète, il est alors possible d'appliquer des opérateurs de recherche locale pour réparer et améliorer progressivement cette solution. Cependant, puisque la recherche n'est pas exhaustive, l'optimalité n'est pas garantie.

3.6 Modélisation PPC

Une tournée peut être représentée comme une suite de nombres uniques (à l'aide de la contrainte AllDifferent), puisque qu'un client final ne doit être servi qu'une seule fois. Chaque nombre indique un nœud et l'ordre d'apparition détermine l'ordre dans lequel les nœuds sont servis. Ainsi, le nœud au rang $i + 1$ sera le successeur du nœud au rang i ; cette relation doit être matérialisée par une contrainte. À ces principes de base, nous pouvons ajouter un ensemble de contraintes pour assurer que les tournées encodées par cette succession de nombres soient réalisables et correspondent bien au problème ciblé. Les contraintes font intervenir un ensemble de données qui permettent de modéliser : les distances entre les nœuds, les fenêtres de temps, les quantités associées aux demandes, etc.

3.6.1 Données

$$G \text{ Ensembles des clients finaux} \tag{3.1}$$
$$N \text{ Ensemble des nœuds} \tag{3.2}$$
$$V \text{ Ensemble des véhicules} \tag{3.3}$$
$$B_v \text{ Ensemble des nœuds de départ, chacun associé à un véhicule} \tag{3.4}$$
$$F_v \text{ Ensemble des nœuds de fin, chacun associé à un véhicule} \tag{3.5}$$
$$D \text{ Ensemble des dimensions} \tag{3.6}$$

Ces ensembles forment la structure qui définit le problème. (3.1) est l'ensembles des clients finaux dont la demande doit être satisfaite. Plusieurs nœuds peuvent être nécessaires pour représenter un même client final en fonction des contraintes ou des variantes à traiter (3.2). L'ensemble des véhicules (3.3) est directement lié aux ensembles de nœuds représentants les dépôts de départs (3.4) et de fin de tournées (3.5) puisque pour chaque véhicule il

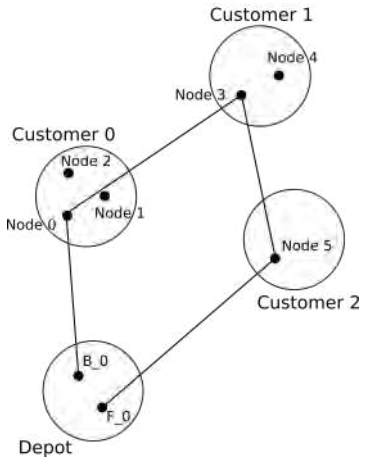

FIGURE 3.5 – Représentation schématique d'une route

existe un nœud dans chaque ensemble. Les dimensions (3.6) permettent de représenter les unités qui sont manipulés par les quantités (demandes) et capacités ainsi que les distances séparant les différents nœuds (temps, kilomètres et valeur). La figure 3.5 montre le départ d'une tournée d'un dépôt avec le nœud B_0 ; le véhicule satisfait la demande du client final $Customer$ 0 par le biais du nœud $Node$ 0. La demande du client $Customer$ 1 est satisfaite via le nœud $Node$ 3 et le client final $Customer$ 5 par le nœud $Node$ 5. La route se termine par un retour au dépôt par le nœud F_0. Cette illustration montre que chaque client final peut être représenté par plusieurs nœuds. La demande d'un client final est satisfaite dès lors que l'un des nœuds qui le compose est servi par un véhicule.

$$N_k \text{ Ensemble des nœuds appartenant à un client final } k \in G \tag{3.7}$$

$$TW_i \text{ Ensemble des fenêtres de temps associé au nœud } i \in N \tag{3.8}$$

$$O_i^u \text{ Horaire d'ouverture de la u-ème fenêtre de temps du nœud } i \in N \tag{3.9}$$

$$M_i^u \text{ Horaire de fermeture de la u-ème fenêtre de temps du nœud } i \in N \tag{3.10}$$

$$S_v \text{ Horaire de début de l'amplitude de travail du véhicule } v \in V \tag{3.11}$$

$$E_v \text{ Horaire de fin de l'amplitude de travail du véhicule } v \in V \tag{3.12}$$

$$C_v^d \text{ Ensemble des capacités du véhicule } v \in V \text{ pour toutes les dimensions } d \in D \tag{3.13}$$

Chaque nœud peut avoir une ou plusieurs fenêtres de temps (3.8), chaque fenêtre étant

constituée d'un horaire d'ouverture (3.9) et d'un horaire de fermeture(3.10). Une fenêtre de temps contraint l'horaire d'arrivée des véhicules pour servir ce nœud. Chaque véhicule est également soumis à un ensemble de capacités (3.13) qu'il ne peut pas dépasser en cumulant les quantités associées aux nœuds qu'il visite.

$$T_{i,j}^{d_v} \text{ Mesure séparant les deux nœuds } i, j \in N \cup S_v \cup E_v \quad (3.14)$$
$$\text{dependante de la dimension } d \in D \text{ et du véhicule } v \in V$$
$$U_i^{d_v} \text{ Valeur associée au nœud courant } i \in N, \quad (3.15)$$
$$\text{dépendante de la dimension } d \in D \text{ et le véhicule } v \in V$$
$$(3.16)$$

Les matrices contiennent les distances séparant les nœuds (3.14). On peut voir ces distances comme des quantités accumulées sur les arcs d'un graphe orienté. Ces matrices peuvent représenter une distance en temps [3] ou en kilomètres, ou directement le coût associé aux arcs dans le cas de la matrice de valeur. Si dans un VRP, un véhicule n'a pas de dépôt définit alors nous définissons un dépôt fictif pour lequel les distances associées sont fixées à zéro.

Lorsqu'un véhicule sert un nœud, il peut y déposer des quantités ou en récupérer. Ces quantités correspondent aux demandes, le véhicule accumule alors des quantités durant la tournée. De manière similaire, lorsque le véhicule sert un nœud, cela prend un certain temps, appelé temps de service (3.15). Le temps est une dimension similaire aux quantités associées aux demandes, à la différence que le temps de trajet entre deux nœuds fait

3. Une distance est une mesure qui sépare deux points. Dans une projection cartographique, on peut utiliser une distance qui représente leur écart dans l'espace. Mais il est également possible de mesurer leur distance comme étant le temps pour parcourir l'espace entre ces deux points.

également augmentée le cumul de celle-ci.

F_v Coût associé à l'utilisation du véhicule $v \in V$ (3.17)

K_v^d Coût associé au *makespan* du véhicule $v \in V$ pour la dimension $d \in D$ (3.18)

J_i Coût associé à la non-affection du client final $i \in G$ (3.19)

L_i^d Coût associé au dépassement de la dimension $d \in D$ au nœud $i \in N \cup B_v \cup F_v$ (3.20)

Enfin, les coûts permettent de déterminer le coût de la solution. Tout d'abord, l'utilisation d'un véhicule implique un coût fixe d'utilisation (3.17), ensuite, les dimensions telles que le temps ou le kilométrage ont un coût qui augmente linéairement (3.18). Si certains clients ne peuvent être servis, nous appliquons alors un coût d'exclusion (3.19). Ce coût est appliqué si pour un client aucun des nœuds qui le compose n'est actif dans la solution. En effet, il n'est pas assuré que les problèmes soumis à l'API aient une solution réalisable. Dans certains cas, nous pouvons tolérer le dépassement d'une fenêtre de temps ou de la quantité maximale transportée par un véhicule, à cela nous pouvons associer une pénalité (3.20). Certains nœuds peuvent également limiter le poids des véhicules qui viennent les visiter du fait de restrictions sur les voies d'accès.

3.6.2 Variables

$$z_i = \begin{cases} v, & \forall i \in N,\ v \in \{V \cup \{-1\}\} \\ v, & \forall i \in B_v \cup F_v,\ v \in V \end{cases} \tag{3.21}$$

La variable z_i (3.21) lie les nœuds à l'indice d'une route. Si un nœud n'est affecté à aucune route, alors z_i est fixée à -1. Chaque véhicule possède un nœud de départ et un nœud de fin dont la variable z_i à la tournée correspond à l'indice du véhicule.

$$n_i = \begin{cases} j, & \forall i \in N \cup B_v, j \in N \cup F_v,\ \forall v \in V \\ \emptyset, & \forall i \in F_v,\ \forall v \in V \end{cases} \tag{3.22}$$

La variable n_i (3.22) permet de lier un nœud à son successeur dans la tournée à laquelle il est associé. La variable de successeur permet donc d'encoder la tournée et de lui donner un ordre. Les nœuds de fin n'ont pas de successeurs.

L'utilisation des chaines de Lin-Kerninghan a pu montrer que les solutions de bonnes qualités ont une propriété commune (Applegate *et al.*, 2003; Toth et Vigo, 2003; Arnold et Sörensen, 2019). En effet, les nœuds qui se succèdent dans une tournée sont parmi les nœuds les plus proches. Nous pouvons alors utiliser ce constat pour réduire le domaine des successeurs des nœuds.

$$a_i = \begin{cases} b, & b \in \{0,1\}, \forall i \in N \\ 1, & \forall i \in B_v \cup F_v,\ \forall v \in V \end{cases} \tag{3.23}$$

Pour savoir si un nœud est actif dans la solution courante, nous utilisons la variable booléenne a_i (3.23). Les nœuds de départ et de fin sont toujours actifs.

$$c_i^d \in \mathbb{Z}^+,\ \forall i \in N \cup B_v \cup F_v,\ \forall d \in D, \forall v \in V \tag{3.24}$$
$$t_{i,n_i}^{d_v} \in \mathbb{Z},\ \forall i \in N \cup B_v,\ \forall d \in D,\ \forall v \in V \tag{3.25}$$
$$w_{i,n_i}^d \in \mathbb{Z}^+,\ \forall i \in N \cup B_v,\ \forall d \in D \tag{3.26}$$
$$w_{i,n_i}^d = 0,\ \forall i \in F_v,\ \forall d \in D,\ \forall v \in V \tag{3.27}$$
$$l_i \in \mathbb{Z}^+,\ \forall i \in N \tag{3.28}$$

Chaque nœud est associé à une variable cumulative (3.24) qui décrit la valeur courante de la dimension considérée lorsque le véhicule le sert. La variable de transit (3.25) fait appel aux données (3.14) et (3.15) de la dimension considérée pour déterminer la valeur à ajouter à la variable de cumul du nœud successeur. Une variable de relai (3.26) est également introduite pour pouvoir représenter les temps d'attente générés par les fenêtres de temps par exemple. Une variable permettant de mesurer les dépassements (3.28) est également ajoutée au modèle, elle ne sera utile que dans le cas de contraintes non strictes.

3.6.3 Contraintes

Contraintes basiques du VRP

$$a_i = 0 \Leftrightarrow n_i = i \Leftrightarrow z_i = -1 \tag{3.29}$$

$$n_i = j \Rightarrow z_j = z_i \tag{3.30}$$

$$n_i = j \Rightarrow c_j^d = c_i^d + t_{i,j}^d + w_{i,j}^d \tag{3.31}$$

$$t_{i,n_i}^{d_v} = T_{i,n_i}^{d_v} + U_i^{d_v} \tag{3.32}$$

$$\text{AllDifferent}(n) \tag{3.33}$$

Un nœud qui n'est pas actif dans la solution (dont la variable a_i est égale à zéro) implique que ce nœud est son propre successeur et il n'est affecté à aucune tournée (3.29). Et réciproquement, s'il n'est affecté à aucune tournée, alors son successeur est lui-même et il n'est pas actif dans la solution. Cette propriété permet de respecter la contrainte AllDifferent, en assurant également que chaque tournée commence à un dépôt B_v et finisse à un dépôt F_v tout en permettant que des nœuds qui ne permettent pas de fournir des solutions réalisables soient inactifs.

Si le successeur de i est j, alors i et j appartiennent à la même tournée (3.30). De plus, cela implique, tel que représenté dans la figure 3.6, que la variable de cumul de j est la somme de la variable de cumul de i, de la variable de transit de i à j et de la variable relai entre i et j (3.31).

Cette contrainte de cumul couplée aux successeurs permet de propager les réductions de domaines des variables liées aux nœuds en fonction de leur ordre dans la tournée. Ainsi, si j est successeur de i nous pouvons propager un certain nombre d'informations dès que i est affecté. La propagation se fait alors du début de la tournée vers la fin de celle-ci.

La contrainte globale (3.33) permet de représenter que tous les successeurs sont uniques. Et dans la mesure où les variables de cumul d'un nœud sont liées à son successeur (3.32) alors les sous-tours sont automatiquement filtrés. En effet, si deux nœuds i et j appartiennent à la même tournée, si pour une dimension d : $c_i^d < c_j^d$ alors il est impossible que i soit successeur de j ou que i se situe après j dans cette tournée.

Time windows

$$c_i^d \in \bigcup_{u=1}^{TW_i} [O_i^u; M_i^u],\ d = Time,\ \forall i \in N \tag{3.34}$$

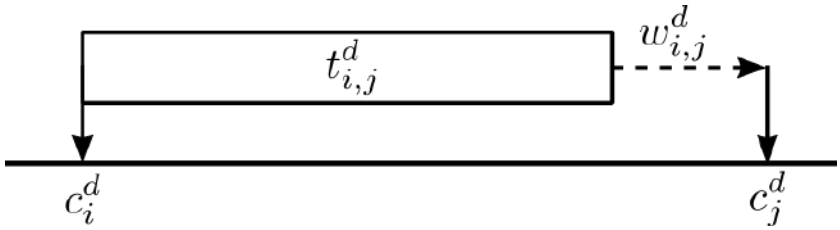

FIGURE 3.6 – Représentation de la variable cumulative entre i et son successeur j.

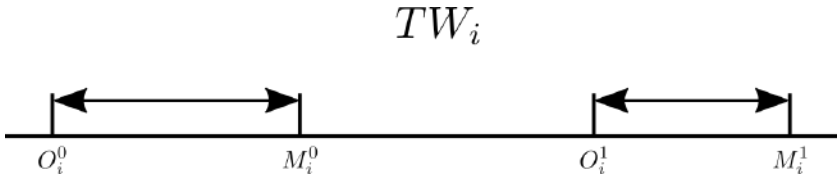

FIGURE 3.7 – Hard Time windows.

Chaque fenêtre de temps TW_i est composée d'un couple O^u_i et M^u_i qui définissent les horaires d'ouverture et de fermeture du nœud i. Si O^0_i n'est pas défini il est considéré égale à zéro. Si M^n_i (n étant le dernier créneau horaire du nœud) n'est pas défini, il est égal à une valeur entière suffisamment grande. Les fenêtres horaires d'un nœud sont triées par ordre croissant et ne peuvent avoir d'intersection.

La contrainte (3.34) assure que la variable de cumul du nœud i sur la dimension de temps appartient à au moins l'un des intervalles définis (Figure 3.7).

Semi-Soft Time windows

$$\sum_{i \in N_k} a_i \leq 1, \ \forall k \in G \tag{3.35}$$

$$c^d_i \geq O^0_i, \ \forall i \in N, \ d = Time, \ \forall i \in N \tag{3.36}$$

$$\|TW_i\| = 1 \tag{3.37}$$

$$l_i = \begin{cases} c^d_i - M^0_i, & c^d_i > M^0_i \\ 0, & otherwise \end{cases} \quad \forall i \in N, \ d = Time \tag{3.38}$$

La variante Semi-Soft Time windows ne permet pas de définir des fenêtres de temps qui soient complètement disjointes (Figure 3.8). Nous pouvons alors dupliquer les nœuds et générer pour chaque fenêtre de temps sur un nœud indépendant pour représenter la demande d'un même client final, mais à des horaires différents. Nous assurons que pour un même client, il n'existe qu'un nœud actif (3.35) similairement à ce qui est présenté

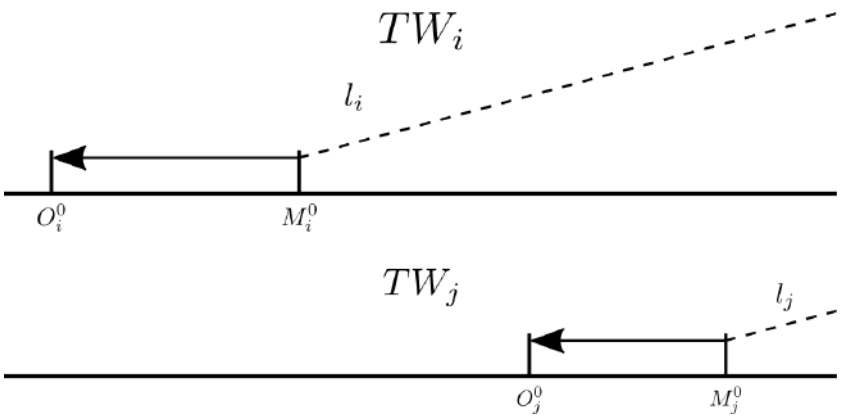

FIGURE 3.8 – Semi-soft Time windows.

dans la figure 3.5. Ainsi, chaque nœud a une unique fenêtre de temps (3.37). Cela assure de ne considérer qu'un seul coût de retard pour chaque client final au sein de la fonction objectif. Chaque fenêtre de temps a un horaire d'ouverture O_i^0 (3.36) mais l'horaire de fin peut être dépassé de manière à ce que le dépassement de M_i^0 induise l'augmentation de la variable l_i (3.38) et génère un coût additionnel dans la fonction objectif.

Amplitude horaire

$$c_i^d < E_v,\ \forall i \in N \cup B_v \cup F_v,\ d = Time, \forall v \in V \tag{3.39}$$

$$c_i^d > S_v,\ \forall i \in N \cup B_v \cup F_v,\ d = Time, \forall v \in V \tag{3.40}$$

Contrairement aux clients, les véhicules ne peuvent avoir qu'une seule fenêtre de temps qui est impactée sur les fenêtres horaires des nœuds de départ et de fin représentant les dépôts associés à la tournée. Il est également possible d'appliquer la variante semi-soft sur le nœud de fin d'une tournée.

Capacités

$$c_i^d < C_v^d,\ \forall i \in N \cup B_v \cup F_v,\ \forall d \in D - \{Time\}, z_i = v, v \in V \tag{3.41}$$

La gestion des quantités est similaire à celle de la dimension de temps. À la différence qu'il n'est pas nécessaire de considérer des fenêtres pour servir les nœuds. Nous devons juste nous assurer qu'en tout nœud de la tournée la quantité cumulée est supérieure ou égale à zéro (puisque définie dans $\mathbb{Z}^+$) et inférieure ou égale à la capacité du véhicule. Il

est également possible d'ajouter une variante autorisant le dépassement sur la capacité du véhicule, similairement à la variante Semi-Soft Time windows. Attention toutefois, la proposition d'une tournée avec un dépassement de charge trop important peut engager la responsabilité du chauffeur, du responsable de tournée et du service ayant proposé la solution. Cette possibilité n'est à utiliser que pour des dimensions qui n'ont pas de réalité matérielle ou dont le dépassement ne contrevienne pas à la loi.

Relations

$$A_{i,j} = a_i * a_j,\ i,j \in N \tag{3.42}$$

$$a_i \leq a_j,\ i,j \in N \tag{3.43}$$

$$v_i * A_{i,j} = v_j * A_{i,j},\ i,j \in N \tag{3.44}$$

$$PathPrecedence(n_k, \{i,j\}),\ \forall k \in N,\ i,j \in N \tag{3.45}$$

$$n_i * A_{i,j} = j * A_{i,j},\ i,j \in N \tag{3.46}$$

Pour modéliser les contraintes des utilisateurs, il est parfois nécessaire de lier des nœuds entre eux. Forcer une égalité sur le véhicule auquel appartiennent deux nœuds peut rendre l'application d'opérateurs de recherche locale difficile ou nécessiter un *backtrack* important, ce point est abordé dans la section 3.7.1. De même, l'un des nœuds appartenant à une relation peut être impossible à servir, rendre la relation obligatoire dans la solution, et force alors tous les nœuds de la relation à être inactifs. Pour activer une contrainte de type relation, nous avons besoin d'une variable qui détermine si deux nœuds sont actifs dans la solution (3.42). Lorsqu'une relation implique plus de deux nœuds, nous devons nous assurer que des sous parties ne puissent s'activer indépendamment (3.43). Nous pouvons alors appliquer différentes relations, pour forcer des nœuds à être affectés à une même tournée (3.44). Pour forcer un ordre, mais laisser des visites s'intercaler entre les nœuds, nous devons tout d'abord assurer que les nœuds appartiennent à la même tournée (3.44), puis que les nœuds respectent bien la contrainte globale de *path precedence* (3.45). Une relation de séquence, définit un ordre, mais ne laisse pas s'intercaler des visites intermédiaires (3.46).

3.6.4 Objectif

$$\sum_{d\in D}(\sum_{v\in V}(c^d_{F_v}-c^d_{B_v})*K^d_v+\sum_{k\in G}\sum_{i\in N_k}(1-a_i)*J_i+l^d_i*L^d_i) \tag{3.47}$$

La première partie de la fonction objectif somme le coût du *makespan* de chaque tournée et de chaque dimension. C'est à dire que l'on calcule pour chaque couple dimension/tournée la différence des variables cumulatives entre les nœuds de départ et de fin de tournées que l'on multiplie par le coût associé. De plus, nous devons comptabiliser le coût des clients inactifs dans la solution, ainsi que les coûts associés aux dépassements des fenêtres de temps et les dépassements sur les autres dimensions.

Il est à noter qu'il existe d'autres dimensions en particulier relatives au temps. En effet, il peut être nécessaire de comptabiliser une dimension de temps dans laquelle nous ne prenons pas en compte les temps d'attente. Alors la dimension de temps sera dupliquée mais sur ce duplicata les variables de relai seront fixées à zéro et les fenêtres de temps ne seront pas appliquées.

3.7 Solveurs interfacés

Proposer un service dans le cadre du dernier kilomètre oblige à rechercher la performance en complément de traiter un large panel de variantes. Ainsi, il est nécessaire de rester en alerte sur les avancées des différentes solutions disponibles. Nous donnons priorité aux projets *open source*, puisqu'ils correspondent à la philosophie du projet Optimizer-API. Cependant, le besoin de performances ne doit pas nous rendre agnostique des solutions propriétaires. Nous utilisons principalement le solveur de PPC OR-Tools et en particulier sa librairie *routing* qui fournit un outil qui nous permet de couvrir la plupart des variantes proposées au sein de l'API. Par ailleurs, nous utilisons d'autres solveurs qui ne sont pas basés sur le modèle. VROOM, qui implémente des heuristiques et métaheuristiques issues de la littérature, vise à fournir de bonnes solutions dans un court temps de traitement, mais ne tolère qu'un nombre réduit d'attributs. RADOS, qui est une métaheuristique développée par Lucas (2020), propose également une méthode de résolution concentrée sur la vitesse de résolution.

3.7.1 OR-Tools

OR-Tools, développé et maintenu par Google, propose un « wrapper » vers de multiples solveurs MIP (Mixed-Integer Programming), tels que SCIP, GLPK, CPLEX ou Gurobi. À cet égard, il permet à partir d'un modèle unique d'appeler de multiples outils pour le résoudre. OR-Tools intègre également un solveur de programmation par contrainte (PPC), qui est progressivement migré vers le nouveau solveur CP-SAT qui propose de résoudre les problèmes par le biais de l'équivalence avec le problème de satisfaisabilité booléenne.

OR-Tools propose également une librairie *routing* afin de modéliser le VRP et ses variantes et de le résoudre à l'aide du solveur PPC. Le modèle mis en place pour l'utiliser est présenté dans la section 3.6. Cette librairie, propose pour résoudre le VRP de générer tout d'abord une première solution à l'aide d'une heuristique parmi un panel déjà intégré, puis d'appliquer une métaheuristique : GRASP, Guided Local Search, Recuit simulé ou recherche tabou.

Heuristiques

Une solution initiale de bonne qualité permet lors de la recherche locale de faciliter la convergence vers des solutions intéressantes. OR-Tools propose un certain nombre d'heuristiques [4] adaptées au modèle PPC et aux problèmes de tournées de véhicules. Cependant comme chaque problème a ses particularités, il est difficile de déterminer à l'avance quelle sera l'heuristique la plus pertinente en faisant débuter la recherche locale dans un voisinage prometteur.

- **Path cheapest arc** Une route est ouverte à partir de son dépôt, le nœud qui génère le plus petit coût est ajouté, la route est étendue de proche en proche à partir de chaque nœud ajouté (plus proche voisin).
- **Path most constrained arc** Similaire à Path cheapest arc mais favorise les arcs les plus contraints.
- **Savings** Algorithme de Clarke et Wright (1964).
- **Sweep** Algorithme de Wren et Holliday (1972).
- **Christofides** Algorithme de Christofides (1976).
- **All unperformed** Rend tous les nœuds inactifs dans la solution (ne fonctionne que si les nœuds sont optionnels)

4. La documentation relative aux heuristiques de OR-Tools est accessible à l'adresse suivante : `https://developers.google.com/optimization/routing/routing_options`

— **Best Insertion** Insère progressivement le nœud le moins coûteux à la position la moins coûteuse. La fonction de coût est basée sur la fonction objectif.
— **Parallel cheapest insertion** Similaire à Best insertion, mais la fonction de coût est basée sur les arcs.
— **Local cheapest insertion** Insère les nœuds dans l'ordre de leur apparition dans le modèle à la position la moins coûteuse, la fonction de coût est basée sur les arcs.
— **Global cheapest arc** Connecte progressivement deux nœuds qui génère le segment de route le moins coûteux.
— **Local cheapest arc** Sélectionne le premier nœud qui n'a pas de successeur lié et le connecte au nœud qui génère le segment de route le moins coûteux.
— **First unbound min value** Sélectionne le premier nœud qui n'a pas de successeur lié et le connecte au premier nœud disponible.

Nous déterminons empiriquement pour chaque problème transmis un panel d'heuristiques pertinentes. Chaque méthode du panel est appliquée pour obtenir une première solution. Nous sélectionnons, la solution initiale qui a le coût le plus bas. Il n'existe certes aucune garantie que le voisinage exploré autour de cette solution soit le meilleur, cependant, puisque le temps de résolution est restreint, nous avons pu constater de manière empirique qu'il s'agissait de la meilleure stratégie.

Résolution

Une fois la solution initiale obtenue, il est possible d'appliquer une métaheuristique parmi un ensemble prédéfini :

— **Guided local search** Applique un cœfficient perturbateur sur les opérateurs de recherche locale afin d'échapper aux minima locaux (Voudouris et Tsang, 2003).
— **Simulated annealing** S'inspire du procédé de métallurgie. Ici, la recherche de solution autorise des solutions avec un coût détérioré avec une probabilité qui est réduite progressivement au fil des itérations (Osman, 1993).
— **Tabu search** Cette méthode autorise également des solutions avec un coût détérioré, mais intègre une liste de mouvements interdits afin de ne pas revenir sur le même bassin de solutions (Glover, 1986).
— **Greedy descent** Bien que la descente gloutonne ne soit pas à proprement parler une métaheuristique dans la mesure où la recherche s'arrête dès qu'un minima local est atteint (Feo et Resende, 1995). Elle est classifiée au sein de OR-Tools au même niveau que les méthodes précédentes, puisqu'elle implémente les mêmes logiques

de gestion des opérateurs de recherche locale.

Nous utilisons exclusivement *Guided local search* pour la résolution du VRP.

3.7.2 RADOS

Le solveur RADOS est issu des travaux de thèse de Flavien Lucas (2020). Nous avons pu collaborer afin de l'intégrer au sein du projet Optimizer-API et l'adapter afin qu'il puisse correspondre aux attentes du projet. Cette méthode a pour objectif de fournir dans un laps de temps très court des solutions de bonne qualité. Elle permet actuellement de traiter les problèmes de CVRP et de HFVRP. En effet, les travaux de thèse de Flavien Lucas portent, entre autres, sur l'impact des méthodes d'apprentissage pour l'exploration de voisinages dans la résolution du VRP. Afin d'alimenter le moteur d'apprentissage, un grand nombre de solutions est nécessaire. La solution initiale est générée à l'aide de l'algorithme de Clarke et Wright (1964), puis une recherche locale (Desaulniers *et al.*, 2014) est appliquée à l'aide d'un ensemble d'opérateurs intra-route : Relocate, Swap-Exchange, 2-opt ; inter-routes : Relocate, Swap, Crossover/2-opt*, Path-move, Split ; ou encore les chaînes d'éjection. Une fois les voisinages appliqués, la solution se trouve dans un minimum local. Afin d'explorer d'autres voisinages, deux méthodes sont applicables. Premièrement, il est possible de multiplier les solutions initiales à l'aide d'un Greedy Randomized Adaptive Search Procedure (GRASP). Deuxièmement, il est possible de procéder à une recherche tabou qui autorise à explorer des solutions voisines de moins bonnes qualités afin d'échapper aux minima locaux.

3.7.3 VROOM

Développé et maintenu par Verso, le solveur Vehicle Routing Open-source Optimization Machine (VROOM). Il permet de traiter les variantes suivantes :

— **TSP** Travelling Salesman Problem
— **CVRP** Capacitated VRP
— **VRPTW** VRP with Time Windows
— **MDVRPTW** Multi-depot VRP with Time Windows
— **PDPTW** Pickup-and-Delivery Problem with Time Windows

Il avait été initialement intégré pour la qualité des solutions pour le TSP retournées dans un temps inférieur à 300 millisecondes pour des instances allant jusqu'à 1000 points avec un gap moyen à la meilleure solution connue de 3.04%. Progressivement le projet a étendu

les problèmes qu'il pouvait traiter avec toujours pour objectif de conserver la vitesse de résolution en gardant un écart à la meilleure solution connue le plus petit possible. Il n'embarque en revanche à l'heure actuelle pas de notion de coût. La fonction objectif est donc directement liée à la matrice de distance fournie. La notion de coût de fixe est absente.

3.7.4 Autres

D'autres solveurs ont également été candidats à l'intégration au sein de Optimizer-API.

Jsprit Solveur développé et maintenu par Graphhopper, il permet une souplesse similaire à OR-Tools et une interface de définition des problèmes plus simple que ce dernier. Il s'agit d'un bon point de départ pour s'initier aux projets Open-Source liés au VRP. Nous l'avons utilisé principalement afin d'explorer de nouvelles variantes avant de maîtriser OR-Tools. Il n'a jamais été utilisé en production, en effet, il est gourmand en ressources et n'atteignait pas les performances attendues.

LocalSolver Nous avons en 2018 exploré la possibilité d'utiliser LocalSover 8.0. Nos premiers essais ont pu montrer qu'il était performant sur les instances académiques et permettait de converger vers des solutions de bonne qualité plus rapidement qu'avec OR-Tools. Cependant, une fois des problèmes réalistes transmis, le rapport c'est inversé. Nos essais préliminaires n'étant pas concluants, n'avons alors pas poursuivi les investigations.

CPLEX CPO Également courant 2018, nous avons tenté de prendre en main CPLEX CPO, qui propose un solveur par contraintes. Puisque le cœur est similaire à OR-Tools, nous avons tenté de porter les bases du modèle que nous utilisons. Nous n'avons malheureusement pas réussi à obtenir de résultats compétitifs dans le temps alloué à son essai.

3.8 Résultats numériques

La table 3.2 résume les résultats pour différents ensembles d'instances issues de la littérature. Les calculs sont réalisés avec un CPU Intel i7-7700HQ 2.8GHz avec pour système d'exploitation Linux Mint 20. Ces résultats sont obtenus en utilisant le modèle

PPC et OR-Tools v7.5. Les distances sont calculées en utilisant une distance euclidienne, sauf pour les instances Duhamel *et al.* (2011) pour lesquelles les matrices ont été calculées en utilisant le calculateur d'itinéraire de Google. Cependant les matrices sont symétriques. Le temps maximal de résolution est fixé à 10 minutes, sauf pour les instances traitant la variante MDVRPTW pour lesquels la durée maximale est fixée à 1 heure. La table

Variante	Source	Instances	Nœuds	Temps moy.	Écart moy.
HFVRP	Duhamel *et al.* (2011)	DLP	20-95	289,22	2,63 %
HFVRP	Duhamel *et al.* (2011)	DLP	102-150	306,48	5,65 %
HFVRP	Duhamel *et al.* (2011)	DLP	152-196	369,12	7,22 %
HFVRP	Duhamel *et al.* (2011)	DLP	203-256	419,77	7,63 %
MDVRP	Cordeau *et al.* (1997)	p & pr	50-100	281,16	0,51 %
MDVRP	Cordeau *et al.* (1997)	p & pr	140-360	276,96	5,81 %
MDVRPTW	Cordeau *et al.* (2001)	p & pr	48-144	261,74	1,49 %
MDVRPTW	Cordeau *et al.* (2001)	p & pr	192-288	1145,53	6,28 %
VRPTW	Gehring et Homberger (1999)	C, R & RC	200-400	243	9,3 %
			Moy.	340,57	6,58 %

TABLE 3.2 – Écart à l'état de l'art

3.3 présente les résultats obtenus avec le modèle PPC pour le problème de tournées de véhicules avec dépôts multiples et fenêtres de temps (MDVRPTW) en les comparant aux résultats obtenus par Vidal *et al.* (2013a) et à la meilleure solution connue (BKS). Nous pouvons constater que le modèle PPC permet d'atteindre les meilleures solutions connues pour les petites instances. Cependant, l'écart augmente à mesure que la taille des instances augmente. Également le temps pour obtenir des solutions de bonne qualité est élevé face à l'état de l'art, il s'agit des conséquences de sa généricité.

Le détail des résultats obtenus pour les variantes MDVRP (Table A1), HVRP (Tables A2, A3 et A4) et VRPTW (Table A5) est présenté en annexe.

Nous pouvons constater que le modèle PPC fournit des solutions de bonne qualité pour les petites instances. Par ailleurs, l'écart moyen à la meilleure solution connue augmente avec la taille des instances. Cependant puisque le temps pour retourner une solution est strictement limité, l'écart reste raisonnable.

				Modèle PPC			Vidal et al.(2013)		BKS
Instance	n	m	d	Coût	Temps	Écart	Coût	Temps	Coût
pr01	48	2	4	**1074,12**	226,83	0,00 %	**1074,12**	4	**1074,12**
pr02	96	3	4	1769,48	169	0,41 %	1762,61	14,84	**1762,21**
pr03	144	4	4	2397,05	378	0,99 %	**2373,65**	22,58	**2373,65**
pr04	192	5	4	2950,71	273	4,82 %	**2815,11**	76	**2815,11**
pr05	240	6	4	3181,81	194	7,41 %	**2962,25**	112	**2962,25**
pr06	288	7	4	3772,82	1 572	5,13 %	**3588,78**	173,29	**3588,78**
pr07	72	2	6	**1418,22**	222	0,00 %	**1418,22**	6,58	**1418,22**
pr08	144	3	6	2156,99	186	2,87 %	**2096,73**	30,84	**2096,73**
pr09	216	4	6	2803,44	416	3,35 %	**2712,56**	67,1	**2712,56**
pr10	288	5	6	3780,09	585	9,10 %	**3464,65**	196,39	**3464,65**
pr11	48	2	4	1018,02	226	1,22 %	**1005,73**	2,71	**1005,73**
pr12	96	3	4	1526,22	327	4,21 %	**1464,5**	21,68	**1464,5**
pr13	144	4	4	2017,34	321	0,78 %	**2001,81**	37,94	**2001,81**
pr14	192	5	4	2391,65	279	8,94 %	**2195,33**	84,52	**2195,33**
pr15	240	6	4	2549,73	1 686	4,79 %	**2433,15**	162,06	**2433,15**
pr16	288	7	4	2962,37	2 614	4,43 %	**2836,67**	206,06	**2836,67**
pr17	72	2	6	1245,53	209	0,75 %	**1236,24**	13,55	**1236,24**
pr18	144	3	6	1853,23	352	3,64 %	**1788,18**	42,58	**1788,18**
pr19	216	4	6	2299,76	2 309	1,89 %	**2257,13**	110,84	**2257,13**
pr20	288	5	6	3325,73	654	11,45 %	**2984,01**	286,19	**2984,01**
				Moyenne	660	3,81%		83.59	

Table 3.3 – MDVRPTW

3.9 Autres méthodes de résolution

3.9.1 Découpe récursive

Le calcul des matrices de temps et kilométriques est coûteux en termes de ressources et de temps. Le router possède une limite de temps associée aux requêtes qu'il peut traiter, au-delà de cette durée limite, il interrompt la connexion. Puisque nous souhaitons avoir un temps de réponse rapide, il n'est pas envisageable de consacrer trop de temps au calcul des matrices. D'ailleurs, les tables 3.4 et 3.5 montrent que le temps de calcul des matrices peut varier selon les calculateurs d'itinéraires utilisés. Or ils dépendent du moyen de transport associé aux tournées. Nous devons donc rester, autant que possible, sur des tailles de matrices inférieures au temps limite de résolution. Également, traiter de grandes instances au sein d'une seule résolution ne permet pas de fournir des résultats de bonne qualité dans un laps de temps réduit.

Distance/Temps	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Moyenne (s)	0.85	1.69	3.11	4.26	6.09	8.1	10.61	13.29	16.28	19.27

TABLE 3.4 – Temps de calcul pour obtenir des matrices bi-dimensionnelles (Distance/-Temps) avec OSRM 5.21

Temps	**10**	**20**	**40**	**50**	**100**	**200**	**300**	**400**	**500**
Moyenne (s)	0.58	0.73	2.89	3.98	15.90	60.00	127.80	212.60	295.90

TABLE 3.5 – Temps de calul pour obtenir une matrice de temps avec HERE Matrix API, dans un rayon de 100km

Pour répondre à cette problématique, il est possible de fournir un seuil au-delà duquel le problème fourni est découpé en deux parties, et ce de manière récursive, jusqu'à ce que chaque partie contienne moins d'éléments que le seuil demandé. Par exemple, un problème de 1200 services et un seuil de « découpe », de 500 pourra être coupé en deux sous-problèmes de 750 et 450 services à l'aide d'une méthode de clustering (Section 4.4). Le sous-problème de 750 sera ensuite redécoupé en deux parties, par exemple, de 350 et 400 services. Chaque sous partie est traitée de manière séquentielle, car la flotte est partagée par tous les sous-problèmes. Alors, puisque la taille de la flotte est réduite, le premier sous-problème aura besoin d'un nombre réduit de véhicules. Les véhicules inutilisés sont transmis aux sous-problèmes suivants. Nous pouvons également supposer que les tournées avec un taux de charge trop faible, seraient plus utiles dans un autre sous-problème, si celle-ci est sous-dimensionnée.

Ce découpage en amont de la résolution n'est jamais remis en question, il est défini pour découper le problème et de ne calculer que des matrices de taille réduite en formant des sous-problèmes.

3.9.2 Dichotomious

La méthode « Découpe récursive » divise et ne remet jamais en question la séparation des sous-problèmes. Par ailleurs, les distances utilisées sont euclidiennes. Cela a peu d'impact lorsque nous travaillons à une échelle macroscopique. Cependant, pour des problèmes en milieu urbain, il est nécessaire d'avoir un niveau de détail important. À cet égard, nous avons mis en place une méthode qui permet de bénéficier des méthodes de clustering bas niveau en utilisant les matrices de temps et kilométriques. Cette méthode au fil de la résolution remet en question régulièrement le clustering. Cette approche peut

être utilisée en aval de la « Découpe récursive » pour éviter le goulot d'étranglement que peut représenter le calcul des matrices.

L'approche « Dichotomious » demande en premier lieu des matrices de temps et kilométriques. En utilisant celles-ci, le problème à traiter passe par une méthode de clustering en coupant également de manière récursive le problème en deux parties. Cette fois-ci les sous-problèmes sont équilibrés et les véhicules sont répartis entre les sous-problèmes plutôt que partagés. Cette découpe se poursuit jusqu'à des problèmes comportant une centaine de services ou entre 3 et 5 véhicules. Chaque sous-problème ainsi généré est résolu individuellement. Une fois la résolution terminée, nous remontons l'arbre de découpe. À chaque nœud, les tournées sont réparties de manière à recomposer des sous-problèmes toujours de 100 services ou 3 à 5 véhicules. Les tournées sont ainsi réassemblées de proche en proche. Il est à noter que plus les nœuds sont bas dans l'arbre, plus le coût pour désaffecter des services est faible, et progressivement en remontant ce coût augmente. De cette manière, nous évitons de générer des détours incohérents dans les tournées de bas niveau. En remontant, ces points qui étaient esseulés par les découpes précédentes se voient potentiellement affectés à des sous-problèmes qui correspondent mieux à leurs propriétés.

Ce calcul de coût est actuellement paramétré pour le cas d'usage de quelques clients de Mapotempo.

3.9.3 Heuristique PVRP

Les problèmes de tournées périodiques demandent de lier des visites sur plusieurs tournées. Or, OR-Tools et VROOM n'intègrent pas d'opérateurs de recherche locale qui puissent déplacer simultanément des nœuds qui sont sur des tournées différentes. Ainsi, même si la modélisation est possible dans OR-Tools, l'amélioration des solutions initiales se retrouve bloquée du fait de ce manque d'opérateurs. Par ailleurs, la génération de solutions initiales n'est pas adaptée à ce type de variantes.

Nous avons alors entrepris de créer une heuristique pour sa résolution. Les demandes d'un client final peuvent être à répéter plusieurs fois sur horizon de temps de plusieurs jours ou semaines, nous les nommons services afin de les distinguer des demandes ponctuelles. Un service est constitué de plusieurs visites. Nous considérons que la charge de travail doit être équilibrée sur la période considérée. De plus, il est préférable de placer les visites d'un service à intervalles réguliers. Nous définissons pour l'heuristique un ordre de priorité dans lequel les services sont traitées qui est une fonction d'intérêt calculée à partir de

leur priorité, le nombre de visites et leur proximité à la tournée choisie. L'heuristique est itérative, un service est inséré à chaque itération. Lorsqu'une tournée est vide, on pondère la fonction d'intérêt avec la distance aux autres routes déjà construites. Ainsi nous priorisons la construction de tournées sur des secteurs encore non servis.

La procédure d'insertion se déroule ainsi :

— sélection de la route la moins remplie,
— sélection du service avec la fonction d'intérêt la plus élevée et vérification de l'insertion des visites suivantes sur les autres jours de la période selon l'écart idéal,
— insertion des visites du service.

Une fois cette procédure appliquée, il peut rester des services non affectés. Pour cela, nous avons plusieurs post-traitements applicables.

Certains utilisateurs demandent initialement que tous les services auprès du même client final soient traités le même jour. Nous pouvons alors en post-traitement relâcher cette contrainte et poursuivre l'insertion comme décrit précédemment. La phase d'insertion initiale considère un espacement régulier des visites, cette contrainte peut être relâchée afin de donner plus de latitude pour l'insertion des services encore non affectés.

Il arrive également que les routes aient une charge de travail peu importante du fait de la sélection des routes de manière à équilibrer cette charge de travail. Il est alors possible à l'utilisateur de demander à ce que les visites des routes peu chargées soient regroupées ou qu'elles soient transférées sur des routes plus complètes.

3.10 Conclusion

Optimizer-API expose une API REST dans un environnement client-serveur. La réception d'un VRP déclenche une série d'opérations en fonction du contenu de l'instance transmise (*document-centric*). Il est toutefois possible de vérifier son bon déroulement ou de l'interrompre à l'aide d'opérations *business-centric*. La réception d'un VRP passe tout d'abord un ensemble d'étapes :

— vérification de l'identité du client,
— vérification des autorisations du client,
— validation du problème soumis
— sélection de la méthode de résolution

Une fois ces étapes de préparation terminées, nous faisons appel à différentes méthodes de résolution intégrées à Optimizer. Un modèle PPC pour la résolution du VRP riche

est présenté, il est en particulier implémenté avec le solveur OR-Tools. Nous utilisons également VROOM, qui propose des heuristiques et métaheuristiques pour différentes variantes du VRP, ainsi que RADOS qui propose une métaheuristique pour le CVRP et le HFVRP. En amont de l'appel à ces solveurs, nous avons des méthodes de clustering pour séparer le problème complet en sous-problèmes afin de limiter les temps de calculs des matrices et assurer de fournir des solutions de bonne qualité dans un temps raisonnable. Une heuristique pour le PVRP est également présentée.

Chapitre 4

Clustering

Le clustering est une méthode de séparation des données qui divise un ensemble de données en différents « paquets ». Nous souhaitons explorer les différentes approches disponibles et déterminer de quelle manière elles sont compatibles avec le VRP. Clustering et VRP sont utilisées conjointement depuis longtemps. Ce chapitre présente les différentes approches connues. Nous proposons par ailleurs une méthode de clustering KMeans adaptée au contexte de tournées de véhicules ainsi que son utilisation dans les méthodes de résolution *Découpe récursive* et *Dichotomious*

4.1 Intoduction

Le clustering vise à modéliser des problèmes de partitionnement des données en classes (clusters) de manière à ce que les éléments qui les composent soient similaires (compacité). Le clustering est particulièrement étudié en tant que méthode statistique, mais il peut également être perçu comme un problème d'optimisation combinatoire. En effet, les éléments qui forment un cluster doivent être partitionnés en optimisant un ou plusieurs critères. Les clusters doivent être compactes, au regard de la répartition des données dans l'instance traitée. Ceci afin qu'il soit possible de les distinguer des éléments des autres clusters, on parle alors d'homogénéité intra-cluster des données et de séparabilité inter-cluster.

Les applications des méthodes de clustering sont multiples dans de nombreux domaines, tels qu'en biologie, sociologie, géologie, géographie, etc (Brieden *et al.*, 2017). Le clustering se rapporte aux méthodes de classification non supervisées, c'est-à-dire que les classes des éléments ne sont pas connues à l'avance (Saxena *et al.*, 2017).

4.2 Approches

Le clustering n'est pas une méthode unique, il existe une grande diversité d'algorithmes. En effet, il existe de nombreux critères qui peuvent être optimisés (voir section

4.3). Par ailleurs, les données peuvent avoir des structures variées. Cette variété d'objectifs et de structures permet d'envisager de nombreuses méthodes pour résoudre ces problèmes.

Cependant, il est possible de catégoriser les algorithmes de résolution.

4.2.1 Partitionnement

Les algorithmes de clustering à partitionnement ont pour objectif de diviser les données en k classes de façon à minimiser ou maximiser un critère particulier. Ces méthodes utilisent un élément, appelé centroïde, fictif ou un échantillon, pour représenter chaque cluster. Ces algorithmes procèdent de manière itérative jusqu'à stabilisation du centroïde, dont voici quelques exemples :

K-Means L'algorithme est initialisé en supposant k éléments choisis aléatoirement comme étant les centroïdes de clusters. On parcourt ensuite tous les éléments du problème et on les affecte au centroïde dont ils sont le plus proche. À chaque itération, la moyenne de chaque cluster est calculée et les nouveaux centroïdes sont ces éléments fictifs moyens. L'algorithme se poursuit jusqu'à convergence ou jusqu'à atteindre un critère d'arrêt.

K-Medoïds Cet algorithme, aussi appelé PAM, est similaire à K-Means, mais plutôt que de représenter chaque cluster par un point moyen, un point médian (medoïd) est utilisé.

CLARA Dans les cas où les volumes de données sont importants, il peut être difficile de classifier les données frontalement. CLARA permet de ne traiter dans un premier temps que des échantillons des données sur lequel l'algorithme PAM est appliqué. Les k médoïdes obtenus sont évalués sur l'ensemble des données et la meilleure solution est conservée.

Fuzzy C-Means Il s'agit d'une version de K-Means où un élément peut appartenir à plusieurs clusters. Ou l'appartenance à un cluster est donné par une valeur entre 0 et 1. Et la somme de ces valeurs pour chaque élément est 1.

4.2.2 Hiérarchique

Les algorithmes de clustering hiérarchiques permettent de classer les données en utilisant des approches itératives descendantes en divisant les données, ou ascendantes en agglomérant les données. L'approche ascendante considère chaque donnée comme étant

un cluster, et les clusters sont étendus de proche en proche. À l'inverse, l'approche descendante, considère le problème comme n'étant qu'un seul cluster et sépare progressivement les données en plusieurs clusters. Ces deux approches peuvent être représentées à l'aide de dendrogrammes (Murtagh et Contreras, 2012).

Single-linkage Deux clusters sont combinés s'ils possèdent la paire d'éléments la plus proche n'appartenant pas déjà au même cluster.

Complete-linkage Deux clusters sont combinés, si la distance maximale entre une paire d'éléments est la plus petite des distances inter-clusters.

Average-linkage Deux clusters sont combinés si la distance moyenne entre leurs éléments est la plus petite, entre toutes les distances inter-clusters.

DIANA L'algorithme avec tous les éléments dans un cluster. Les clusters sont divisés en deux selon la plus grande mesure de dissimilarité.

Les 4 approches sont présentées dans les chapitres 5 et 6 de (Kaufman et Rousseeuw, 1990). Single-linkage, complete-linkage et average-linkage sont rassemblés sous la dénomination AGNES.

4.2.3 Densité

Les algorithmes basés sur la densité constituent des clusters par propagation (Kriegel *et al.*, 2011) qui s'étendent de proche en proche tant que le voisinage est suffisamment important. D'ailleurs, Single-linkage présenté dans les approches hiérarchiques peut être vu comme un algorithme qui se propage avec une densité nécessaire de 1 élément.

DBSCAN Les éléments sont parcourus séquentiellement. Si un élément n'est pas encore classifié et qu'il a au moins m éléments voisins dont la distance est inférieure à ϵ, alors un nouveau cluster est ouvert et cet élément est considéré comme un point central, sinon il est classifié comme étant du « bruit » [1]. Pour chaque point faisant parti du voisinage, on détermine son voisinage. Si la cardinalité de ce voisinage est supérieure ou égale à m éléments, alors l'élément courant est considéré comme un point central et on peut alors tenter de propager le cluster par ses voisins. Cette méthode ne permet pas de fixer le nombre de clusters à l'avance.

1. On ne peut pas construire ou propager de cluster à partir de cet élément, cependant s'il fait partie du voisinage d'un élément appartenant à un cluster, alors il peut appartenir à celui-ci.

4.3 Évaluation

Chaque méthode de clustering a un comportement différent selon les instances rencontrées. Ainsi, il est intéressant de s'interroger sur la pertinence des méthodes. Pour cela, nous pouvons nous appuyer sur un ensemble de mesures de validation séparées en deux ensembles : les indicateurs externes et les indicateurs internes. Les indicateurs externes mesurent la qualité des clusters à l'aide d'informations obtenues à partir d'un résultat déjà connu. Cela permet de comparer les résultats obtenus par différents algorithmes de clustering à ce qui est attendu et de sélectionner celui qui correspond à la structure de l'instance considérée.

Dans le cas du VRP, il est certes possible de dimensionner des secteurs géographiques pour chaque véhicule à l'aide de clusters. Cependant, si une méthode a pu donner un résultat pertinent dans une configuration particulière, peut-elle donner un résultat satisfaisant ne serait-ce qu'en rajoutant quelques demandes ? Quelques demandes supplémentaires peuvent en effet changer la charge de travail sur certains secteurs et rendre la sectorisation obtenue à l'aide du clustering irréalisable. De même, des nouveaux véhicules peuvent être nécessaires pour satisfaire toutes les demandes. Cette augmentation de la taille de la flotte rend obsolète les clustering précédemment déterminés. Ainsi, nous avons estimé peu pertinent d'intégrer des indicateurs externes dans cette thèse. En effet, il faudrait alors étudier l'intérêt des méthodes liées à l'apprentissage automatique pour déterminer quels opérateurs sont intéressants selon les configurations pour les tournées de véhicules.

4.3.1 Indicateurs Internes

Les indicateurs internes eux, mesurent la pertinence des clusters sans référence externe : l'évaluation se fait uniquement à partir des données à traiter et aux clusters construits par la méthode de clustering appliquée.(Rendón *et al.*, 2011; Legány *et al.*, 2006; Maulik et Bandyopadhyay, 2002). Ces indicateurs ne sont donc pas dépendants de l'historique, nous pouvons alors les utiliser dans tous types de situation, que la zone géographique couverte soit connue ou non.

Somme des erreurs au carré La somme des erreurs au carré (SEC) mesure la distance $d(x_i, \mu_k)$ au carré du cluster k au vecteur moyen μ_k qui le représente. Cette mesure permet d'estimer la déviation du vecteur moyen aux données qu'il représente. (Tsay, 2005)

$$\mu_k = \frac{1}{|N_k|} \sum_{i \in C_k} x_i$$

$$SEC = \sum_{k=1}^{K} \sum_{\forall i \in C_k} d(x_i, \mu_k)^2$$

Scatter criteria Le Scatter criteria (S_k) (Duda *et al.*, 1973) permet d'estimer la matrice de covariance d'un cluster par rapport à son vecteur moyen μ_k, ce critère se formule ainsi :

$$S_k = \sum_{x \in C_k} (x - \mu_k)(x - \mu_k)^T$$

Cette mesure permet de lier les dimensions corrélées dans le cluster, là où la somme des erreurs au carré propose un indicateur où les dimensions sont décorrélées.

Davies-Bouldin $\overline{d_k}$ mesure la moyenne de distances des éléments d'un cluster à son centre.

$$\overline{d_k} = \frac{1}{|N_k|} \sum_{i \in C_k} d(x_i, \mu_k)$$

Pour chaque cluster on mesure sa similarité avec le cluster le plus proche. Moins la similarité entre les clusters sera importante, plus l'indice de qualité sera bon.

$$BD = \frac{1}{|K|} \sum_{k=1}^{K} \max_{k' \neq k} \left(\frac{\overline{d_k} + \overline{d_{k'}}}{d(\mu_k, \mu_{k'})} \right)$$

Indice de Dunn La distance séparant deux clusters k et k' est mesurée par la distance entre leurs points les plus proches

$$\delta(k, k') = \min_{i \in C_k, j \in C'_k} d(x_i, x_j)$$

δ_{min} est la plus petite de ces distances.

$$\delta_{min} = \min_{k \neq k'} \delta(k, k')$$

On note $\Delta(k)$ la plus grande distance séparant deux points du cluster k ; elle est

également appelée diamètre du cluster.

$$\Delta(k) = \max_{i,j \in C_k} d(x_i, x_j)$$

Δ_{max} est la plus grande de ces distances

$$\Delta_{max} = \max_{k \in K} \Delta(k)$$

L'indice de Dunn S_D est défini comme le quotient de δ_{min} sur Δ_{max}

$$S_D = \frac{\delta_{min}}{\Delta_{max}}$$

Communément les indicateurs sont adaptés aux clustering construits autour de centroïdes. En effet, les clusters sont attendus compacts et avec une bonne séparabilité, cependant les mesures associées se font généralement à partir du vecteur moyen. Les modèles et méthodes de clustering par densité ne peuvent pas alors bénéficier de ces indicateurs de validité. Il existe cependant des indicateurs basés sur les arbres de recouvrement minimum qui permettent de les évaluer (Rojas-Thomas *et al.*, 2017; Xie *et al.*, 2020)

4.4 Clustering et VRP

En RO, l'utilisation de méthodes visant à résoudre une version simplifiée du problème à traiter n'est pas nouvelle. C'est d'ailleurs le principe de la programmation linéaire. Le clustering a pour vocation de classer un ensemble d'éléments, cela permet de représenter chaque élément par une classe. Puisqu'une classe représente un ensemble d'éléments, alors exprimer les classes revient à simplifier l'expression des éléments. La convergence de ces deux domaines a donné lieux à diverses méthodes de résolution ou heuristiques pour la résolution du VRP.

4.4.1 Route first, cluster second

Nous avons tout d'abord des méthodes qui résolvent en premier lieu une version simplifiée du problème initial, puis utilisent des méthodes de clustering pour reconstruire une solution au problème initial. C'est en particulier le cas de la méthode *split* proposée par Lacomme *et al.* (2001).

Split Un problème de tournées de véhicules, peut être vu comme un TSP pour lequel il faudrait définir des segments de routes, afin de découper la charge de travail et la répartir sur plusieurs jours. La méthode SPLIT part de ce principe, dans une première phase un tour géant est résolu. Puis, la seconde phase décompose la tournée obtenue en plusieurs routes respectant les capacités des véhicules. Cette découpe est réalisable en temps polynomial, lorsque la taille de la flotte est fixe (Ulusoy, 1985; Lacomme *et al.*, 2001).

4.4.2 Cluster first, route second

Une autre approche pour combiner résolution du VRP et clustering consiste à réaliser la séparation des données en premier puis de résoudre le problème simplifié obtenu.

Fisher et Jaikumar (1981) ont proposé de résoudre le VRP en résolvant tout d'abord un Generalized Assignment Problem (GAP) et en effectuant à partir de chaque ensemble affecté une résolution d'un TSP. L'algorithme est initialisé avec k clusters pour lesquels un point est sélectionné pour le représenter, appelé graine. Pour chaque nœud à servir, on calcule le coût de son affectation aux graines.

$$d_{ij_k} = \min(c_{0i} + c_{ij_k} + c_{j_k0}, c_{0j_k} + c_{j_ki} + c_{i0}) - c_{0j_k} + c_{j_k0}$$

Ici 0 est le dépôt, on calcule donc le coût du détour engendré par l'insertion du nœud i dans la tournée « dépôt-graine-dépôt » Le GAP est alors résolu à l'aide des coûts d_{ij_k} des demandes q_i et des capacités des véhicules C. Les clusters ainsi générés forment des TSP et sont ensuite résolus individuellement.

Sweep Algorithm Le dépôt est ici perçu comme le centre du problème, il est alors possible de diviser l'espace en cônes avec le dépôt pour sommet. L'algorithme Sweep de Gillett et Miller (1974) créé des cônes qui correspondent chacune à l'espace à traiter par une tournée. Chaque cône contient un ensemble de nœuds qui forment un cluster dont le véhicule associé doit satisfaire la demande. Le cône sera alors plus ou moins large en fonction des demandes contenues dans la part attribuée. Chaque part forme un TSP qu'il est possible de résoudre seul.

Algorithme en Pétales Plutôt que de ne former qu'une route par cône, il est possible d'avoir plusieurs tournées au sein de chaque sous-problème ainsi généré. Renaud *et al.* (1996) proposent une méthode pour générer dans chacun des cônes deux tournées en formes de pétales. Deux nœuds sont initialement sélectionnés et les tournées sont constituée en insérant les autres nœuds par leur coût d'insertion le plus faible, tout en maintenant la faisabilité.

Taillard (1993) Similairement à l'algorithme *Sweep*, Taillard (1993) a proposé une méthode où l'espace est divisé en cônes dont le sommet est le dépôt. Ces cônes sont eux-mêmes divisés en secteurs concentriques. Les nœuds de chaque secteur concentrique forment un cluster est utilisé pour constituer un sous-problème qui peut être résolu indépendamment, mais la méthode proposée introduit des mouvements périodiques pour échanger des nœuds entre clusters adjacents.

Bujel *et al.* (2018) ont mis en place une méthode qui utilise DBSCAN de manière récursive de manière à étendre les clusters dans les régions peu denses et réduire le nombre de nœuds dans des régions denses. L'objectif de cette procédure est d'obtenir autant de clusters que de tournées. Nous avons vu que DBSCAN ne permet pas seul d'atteindre cet objectif. Aussi, leur méthode augmente ou diminue les critères de propagation de la méthode de manière à converger vers le nombre de clusters souhaités.

4.5 Propositions

Résoudre un VRP implique de consacrer du temps à sa résolution. Or, même en utilisant des méthodes approchées, le temps pour obtenir des solutions de bonne qualité augmente avec la taille des données à traiter. Également, puisque nous utilisons des itinéraires issus du réseau routier, le temps de calcul des matrices augmente. Il faut qu'il reste raisonnable vis-à-vis du temps alloué à la résolution du VRP.

Au regard de ces impératifs temporels, nous avons développé et intégré au projet Optimizer des méthodes afin de conserver en toute situation des temps de traitements réduits et des résultats qui soient de bonne qualité. Ces méthodes se basent sur l'utilisation d'un algorithme KMeans modifié, que nous notons VRP-Kmeans, afin d'obtenir les propriétés suivantes :

— Respect des contraintes de capacité des véhicules

— Équilibrage de la charge de travail
— Respect des skills

L'algorithme KMeans ne tolère pas dans sa version classique des limites à la taille des clusters. Limiter cette taille peut générer des clusters qui se chevauchent. En effet, dans notre cas, les éléments sont principalement représentés par leurs coordonnées. Le principal critère alors à minimiser est la distance totale intra-cluster. Or, un cluster central dont le cumul des éléments à atteint la capacité de celui-ci refuse l'insertion de nouveaux éléments, alors un autre cluster situé à l'opposé de ces nouveaux éléments par rapport à ce cluster central peut les accepter alors qu'ils sont refusés par le cluster central, quand bien même ce nouveau cluster ne soit pas le cluster avec la plus petite distance. Ce type de situation augmente très nettement le critère à minimiser et provoque lors de la représentation des éléments sur une carte un recouvrement des clusters générés. Bien que des recouvrements ne soient pas totalement problématiques puisqu'il est toujours possible d'y réaliser des tournées de véhicules, cela peut dégrader le coût de la solution. Par ailleurs, cela peut poser problème d'un point de vue opérationnelle.

Imaginons qu'un chauffeur ne puisse assurer sa tournée, celle-ci doit être répartie sur les tournées à proximité. Alors, avoir des tournées bien séparées visuellement facilitera la répartition par le planificateur. Pour pallier à ce défaut, nous mettons à jour l'ordre dans lequel les éléments de l'instance sont parcourus afin que les éléments qui ont généré au chevauchement lors de leur insertion soient parcourus en priorité. À l'initialisation, les nœuds avec la plus forte demande sont considérés comme les premiers éléments à traiter.

Pour équilibrer la charge de travail entre les clusters, à chaque itération de l'algorithme un coefficient est calculé afin de pondérer la distance aux centroïdes. L'affectation aux clusters pleins sera alors plus difficile et favorisée pour les clusters avec un faible taux de remplissage. Il est également possible de fournir la matrice des temps de trajet afin d'affiner l'équilibrage sur le temps de travail ; nous estimons à l'aide de ces données la vitesse moyenne au sein des clusters. Cet algorithme a récemment été extrait du projet Optimizer-API pour former une librairie indépendante afin de rendre accessible cette fonctionnalité au travers de l'outil de Zonage de Mapotempo Web entre autres (voir section 2.6.6).

4.5.1 Découpe récursive

Au-delà d'un certain seuil en nombre de points géographiques, calculer les matrices devient trop coûteux. De même, plus le nombre de nœuds augmente, plus le temps pour

converger vers une bonne solution augmente. La méthode de résolution *Découpe récursive* demande de définir un seuil qui est comparé au concept de services, propre au projet Optimizer, pour déterminer si elle doit être invoquée. Lorsque la *Découpe récursive* est invoquée, telle que représentée par l'algorithme 1, VRP-Kmeans est appliqué afin de construire deux clusters depuis les éléments du VRP (Algorithme 2). Ces deux clusters permettent de constituer deux nouveaux VRP qui possèdent chacun un ensemble disjoint de nœuds du problème complet.

Algorithme 1: Pseudo-code de l'approche Découpe récursive

```
Function recursive-slicing(vrp)
    if vrp.services.size() > ϵ then
        sub-vrp1, sub-vrp2 = slice(vrp,2)
        result1 = recursive-slicing(sub-vrp1)
        remove-used-vehicles(sub-vrp2, result1)
        result2 = recursive-slicing(sub-vrp2)
        return merge-results([result1, result2])
    else
        vrp.compute-matrix()
        return process(vrp)
```

Algorithme 2: Pseudo-code de la fonction de découpe et limite de flotte

```
Function slice(vrp, nb-clusters)
    clusters = vrp-kmeans(vrp, nb-clusters)
    sub-vrps = []
    foreach cluster ∈ clusters do
        sub-vrp = filter-vrp(vrp, cluster)
        limit-fleet-size(vrp, sub-vrp)
        sub-vrps ← sub-vrp
    return sub-vrps ;
```

La flotte, cependant, est identique pour les deux sous-problèmes. Seule une limite sur le nombre de tournées réalisable est définie pour chaque sous-problème au prorata du nombre de nœuds.

Les sous-problèmes générés sont traités de manière séquentielle en invoquant à nouveau la *Découpe récursive*, ainsi les sous-problèmes, s'ils sont au-delà du seuil défini, sont à nouveau divisés. Si leur taille est inférieure au seuil attendu, alors les matrices sont calculées et ils peuvent être résolus (Algorithme 3). Une fois la résolution d'un sous-problème

terminée, les véhicules sont retirés du sous-problème suivant. Ce fonctionnement est illustré par la figure 4.1. Si le nombre de véhicules actifs est inférieur à la taille de la flotte définie, alors la taille de la flotte autorisée pour le sous-problème suivante est augmentée en conséquence. On considère qu'un véhicule est inactif si aucun nœud n'est affecté à sa tournée. Pour les tournées très peu remplies, si leur taux de remplissage est très bas, elles sont détruites et les nœuds associés sont marqués en tant que « non-affectés ». Ceci afin d'éviter que des véhicules qui pourraient être nécessaires dans un sous-problème suivant ne soient pas mobilisés par une tournée peu pertinente.

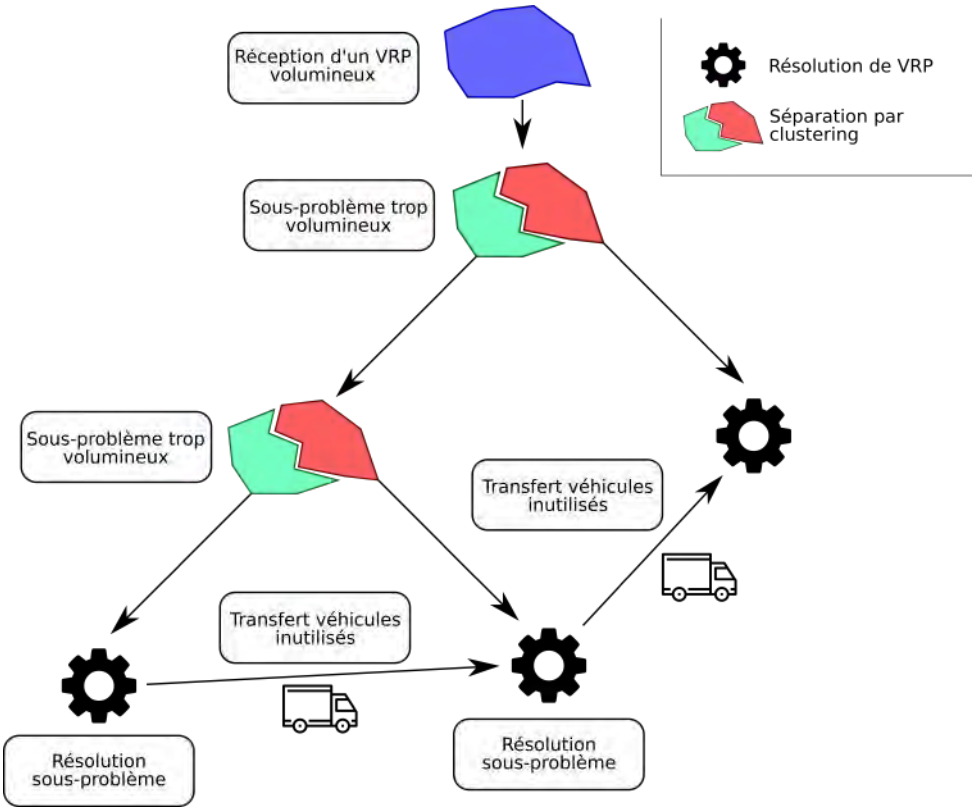

FIGURE 4.1 – Schéma de fonctionnement de la méthode Découpe Récursive

La nature récursive de la *Découpe récursive* et le traitement séquentiel des sous-problèmes générés permet d'assurer que les véhicules pour les tournées d'un sous-problème ne seront pas réutilisés ultérieurement. En revanche, l'un de ses inconvénients est que puisqu'il n'existe pas de résolution de haut niveau, les sous-problèmes traités sont totalement indépendants. Des nœuds qui n'auraient pas été insérés dans un sous-problème ne peuvent pas être affectés dans un autre. Les véhicules avec des skills ou une configuration particulière peuvent être utilisés par défaut dans un sous-problème alors qu'ils sont nécessaires

Algorithme 3: Pseudo-code de la fonction process

```
Function process(vrp)
    if vrp.services.size() > δ and vrp.vehicles.size() > τ then
        results = dichotomious([vrp])
        return merge-results(results)
    else
        return solve(vrp)
```

pour servir les nœuds d'un autre sous-problème. Cette approche peut être catégorisée en tant que *cluster first, route second*.

Résultats numériques

La table 4.1 présente les instances et les résultats de la variante MDVRPTW proposés par Vidal *et al.* (2013a) ; les temps de résolutions sont normalisés pour correspondre aux performances de la machine utilisée pour nos expérimentations (Table A6). Pour nos expérimentations, nous avons utilisé à la fois le modèle PPC au travers du solveur OR-Tools v7.5 seul, ainsi qu'en tant que méthode de résolution suite aux découpes proposées par *Découpe récursive*. Le temps maximal de résolution fixé à 10 minutes. Les calculs sont réalisés avec un CPU Intel i7-7700HQ 2.8GHz avec pour système d'exploitation Linux Mint 20. Les distances sont calculées en utilisant une distance euclidienne. Le seuil de découpe est fixé à 400 nœuds. Les résultats comparatifs des deux approches sont présentés dans la table 4.2. Les coûts en gras mettent en avant la meilleure solution obtenue entre l'utilisation du modèle PPC seul ou de la solution obtenue à l'aide de la *Découpe récursive* en amont du modèle PPC.

Nous pouvons constater que l'utilisation de la méthode *Découpe récursive* permet sur la durée de résolution restreinte de réduire l'écart à la meilleure solution connue pour 20 des 24 instances. Le gain sur l'écart est en moyenne de 2,54% sur ces instances. En pratique, il existe un gain complémentaire en termes de temps, puisque le temps de calcul des matrices n'est ici pas comptabilisé. Cette différence n'aurait été que d'environ 20 secondes si le calculateur d'itinéraire avait été OSRM (Table 3.4), mais aurait dépassé le temps maximal de résolution si le calculateur d'itinéraire appelé avait été HERE Matrix API (Table 3.5).

L'utilisation de la méthode *Découpe récursive* ne permet pas d'assurer l'accès à l'optimum. Cependant, dans un laps de temps restreint, cette approche permet d'accéder à

				Vidal et al.(2013)	
Instance	n	m	d	Coût	Temps (s)
pr12a	480	13	4	8179,8	1216,22
pr13a	600	16	4	9667,2	2223,65
pr14a	720	19	4	11124,01	2661,49
pr15a	840	22	4	13013,97	5369,19
pr16a	960	26	4	14299,87	5417,43
pr18a	520	10	6	8308,32	1793,92
pr19a	700	13	6	10677,61	3017,03
pr20a	880	16	6	11963,91	4352,84
pr21a	420	4	12	6260,53	1135,14
pr22a	600	6	12	7985,37	3083,11
pr23a	780	8	12	9937,43	5583,24
pr24a	960	10	12	11923,72	7993,38
pr12b	480	11	4	6063,26	1179,32
pr13b	600	14	4	7254,17	2877,97
pr14b	720	17	4	8732,29	4010,27
pr15b	840	20	4	10438,72	5249,19
pr16b	960	23	4	11483,22	6904,46
pr18b	520	9	6	6525,72	1599,32
pr19b	700	12	6	8227,25	3265,54
pr20b	880	15	6	10325,8	6111,08
pr21b	420	4	12	4866,57	1489,86
pr22b	600	6	12	6488,5	2970
pr23b	780	7	12	8523,41	6648,24
pr24b	960	8	12	10860,08	12101,76
		Moy.		9297,11	4093,90

TABLE 4.1 – Instances MDVRPTW

des solutions de bonne qualité avec peu ou aucune contrepartie par rapport à méthode de résolution appliquée sur le problème entier. Par ailleurs, nous pouvons constater sur les représentations des solutions obtenues pour l'instance pr24a dans les figures 4.2 et 4.3 que la séparation apportée par l'utilisation de *VRP-KMeans* sépare bien graphiquement les différents clusters. Cette séparation visuelle permet d'améliorer la qualité de service du planificateur. En effet, il est plus facile manipuler les tournées si elles sont visuellement séparées. Ainsi, bien que pour cette instance l'écart à la meilleure solution connue soit plus important, cette perte est compensée par l'amélioration de la qualité de service. Les tournées sont représentées sans les itinéraires depuis et vers les dépôts.

	Modèle PPC		Découpe récursive		
Instance	Coût	Écart	Coût	Écart	Différence
pr12a	9462,55	15,68 %	**9213,49**	12,64 %	-3,04%
pr13a	11113,42	14,96 %	**11060,86**	14,42 %	-0,54%
pr14a	13302,7	19,59 %	**12927,65**	16,21 %	-3,38%
pr15a	15283,89	17,44 %	**15058,17**	15,71 %	-1,73%
pr16a	**16819,89**	17,62 %	17050,29	19,23 %	1,61%
pr18a	9452,77	13,77 %	**9382,78**	12,93 %	-0,84%
pr19a	12347,07	15,64 %	**12099,53**	13,32 %	-2,32%
pr20a	14102,85	17,88 %	**13707,78**	14,58 %	-3,3%
pr21a	7155,36	14,29 %	**6830,04**	9,10 %	-5,19%
pr22a	9304,98	16,53 %	**8726,24**	9,28 %	-7,25%
pr23a	11802,03	18,76 %	**11270,11**	13,41 %	-5,35%
pr24a	**13811,67**	15,83 %	14103,24	18,28 %	2,45%
pr12b	7029,73	15,94 %	**6876,72**	13,42 %	-2,52%
pr13b	8555,78	17,94 %	**8175,28**	12,70 %	-5,24%
pr14b	9972,42	14,20 %	**9745,56**	11,60 %	-2,6%
pr15b	**11569,67**	10,83 %	11640,26	11,51 %	0,68%
pr16b	12951,06	12,78 %	**12838,33**	11,80 %	-0,98%
pr18b	7344,47	12,55 %	**7206,84**	10,44 %	-2,11%
pr19b	9253,9	12,48 %	**9082,26**	10,39 %	-2,09%
pr20b	**11603,54**	12,37 %	11624,31	12,58 %	0,21%
pr21b	5426,83	11,51 %	**5274,87**	8,39 %	-3,12%
pr22b	7632,17	17,63 %	**7256,08**	11,83 %	-5,8%
pr23b	9993,97	17,25 %	**9471,14**	11,12 %	-6,13%
pr24b	12243,12	12,74 %	**11984,87**	10,36 %	-2,38%
Moyenne	10730,66	15,26 %	10525,28	12,72 %	-2,54%

TABLE 4.2 – Résolution du MDVRPTW limité à 10 minutes

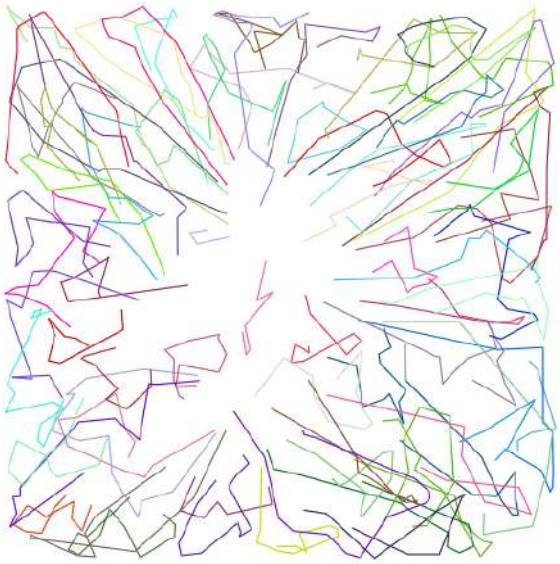

FIGURE 4.2 – Modèle PPC : Représentation de la solution pr24a

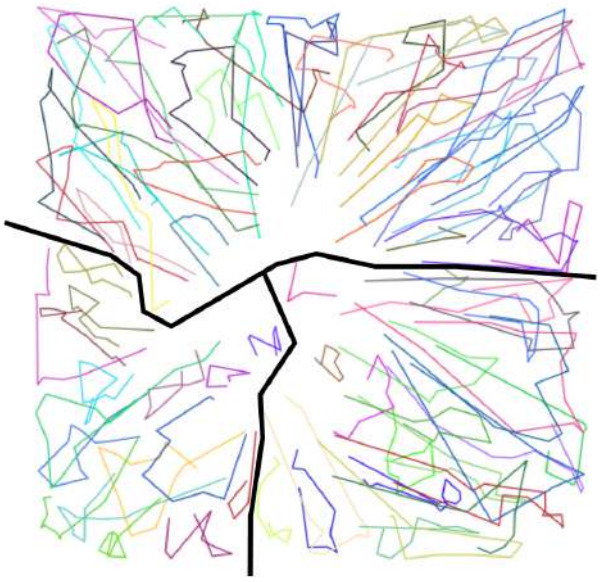

FIGURE 4.3 – Découpe récursive : Représentation de la solution pr24a

4.5.2 Dichotomious

La méthode de résolution *Dichotomious*, dont le pseudo code est présenté dans l'Algorithme 4, n'intervient pas lorsque le calcul des matrices est problématique, mais lorsque le temps pour obtenir une solution de bonne qualité pour un VRP est important. Nous pouvons encore une fois réaliser un clustering à l'aide de la méthode VRP-Kmeans. Puisque le calcul des matrices n'est pas problématique, nous pouvons utiliser cette information pour obtenir des clusters plus précis. La méthode VRP-KMeans est appelé pour construire 2 clusters à partir des éléments du problème. Un sous-problème est créé à partir de chacun de ces clusters (Algorithme 5). Et contrairement à la *Découpe récursive*, les véhicules sont répartis entre les deux sous-problèmes afin de correspondre au mieux aux propriétés attendues pour satisfaire les demandes associées aux nœuds. À noter que *Dichotomious* peut être appelé dès lors que la *Découpe récursive* n'est plus nécessaire (Algorithme 3).

Algorithme 4: Pseudo-code de l'approche Dichotomious

```
Function dichotomious(vrps)
    results = [ ]
    foreach vrp ∈ vrps do
        if vrp.services.size() > δ and vrp.vehicles.size() > τ then
            sub-vrps = divide(vrp)
            sub-results = dichotomious(sub-vrps, null)
            current-vrps = shuffle(sub-vrps, results)
            foreach current-vrp ∈ current-vrps do
                results ← solve(current-vrp)
        else
            results ← solve(vrp)
    return results
```

Tant que chaque sous-problème n'a pas atteint une taille équivalente au seuil, choisi empiriquement, équivalent à la charge de travail de trois tournées, la méthode *Dichotomious* est appelée récursivement. Le seuil actuel est fixé à 100 nœuds ou 3 véhicules. La séparation récursive en deux sous-problèmes forme un arbre binaire. Une feuille est atteinte, lorsque le problème courant est sous le seuil défini. Ce problème est alors résolu. Lorsque deux feuilles issues d'un nœud de l'arbre sont résolues, il est possible de remonter d'un niveau. Ce fonctionnement est illustré par la figure 4.4. Les sous-problèmes provenant du niveau inférieur sont recombinés en appliquant à nouveau VRP-Kmeans. Cependant,

Algorithme 5: Pseudo-code de la fonction séparation et préparation des véhicules

```
Function divide(vrp)
    clusters = vrp-kmeans(vrp, 2)
    sub-vrps = [ ]
    foreach cluster ∈ clusters do
        sub-vrp = filter-vrp(vrp, cluster)
        sub-vrps ← sub-vrp
    dispatch-vehicles(sub-vrps, null) return sub-vrps ;
```

les nœuds présents dans une même tournée sont conservés dans le même sous-problème et le véhicule associé est également affecté au nouveau sous-problème généré. Cette remontée dans l'arbre de décision se fait donc progressivement en effectuant une recombinaison des tournées et des nœuds non-affectés à chaque niveau (Algorithme 6).

Algorithme 6: Pseudo-code pour la recombinaison des sous-problèmes

```
Function shuffle(vrps, results)
    clusters = vrp-kmeans(vrp, results.size, results) ; sub-vrps = [ ]
    foreach cluster ∈ clusters do
        sub-vrp = filter-vrp(vrp, cluster)
        sub-vrps ← sub-vrp
    dispatch-vehicles(sub-vrps, results) return return sub-vrps
```

Il est à noter, que plus le niveau est bas, plus la pénalité pour rendre inactif un nœud dans la solution est basse. En effet, nous considérons que le coût de désaffectation doit être faible à bas niveau puisqu'il est possible qu'il soit dans un sous-problème inadapté et que l'espérance d'affectation dans une tournée d'un autre sous-problème est importante. Cette pénalité est augmentée progressivement en remontant dans l'arbre de décision. L'approche dichotomique enchaîne de multiples phases *cluster first route second*.

La logique derrière cette recombinaison progressive est similaire à l'approche proposée par Taillard (1993). En effet, nous effectuons tout d'abord un découpage à priori, puis progressivement nous explorons s'il est possible d'améliorer la solution en effectuant des échanges avec les tournées proches.

Dichotomious permet, pour des problèmes de taille intermédiaire, de résoudre le VRP en appliquant des logiques de résolution par cluster, avec des échanges locaux entre clusters. Cette logique permet d'obtenir des tournées sectorisées, sans toutefois empêcher les

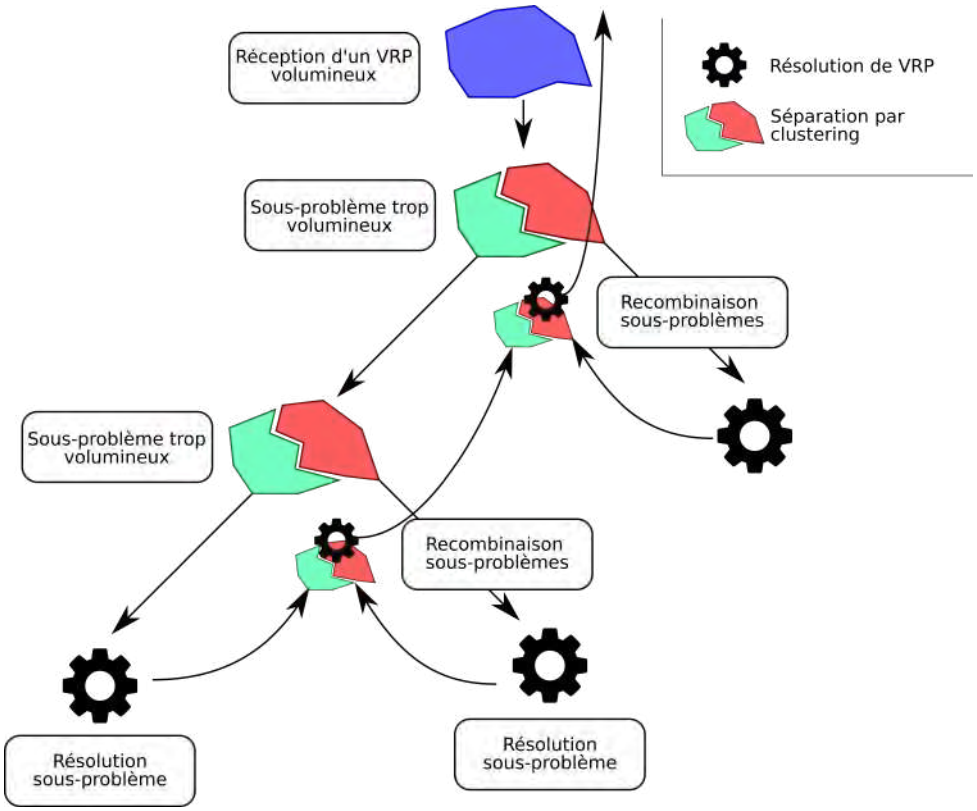

FIGURE 4.4 – Schéma de fonctionnement de la méthode Dichotomious

tournées de se croiser. Elle permet également d'effectuer une intensification locale de la recherche de solution au travers de la recombinaison des sous-problèmes. Cependant cette méthode est limitée par la taille des matrices qu'il est possible de calculer en amont de son application. De plus, les logiques d'améliorations se font localement, il n'existe pas de réflexion au niveau du problème complet pour guider les recombinaisons de sous-problèmes.

Les performances de cette méthode de résolution ne sont pas présentées dans cette thèse. Il s'agit d'un algorithme encore à un stade expérimental. Son fonctionnement actuel est spécifiquement paramétré pour une activité particulière en association avec l'un des clients de la société Mapotempo. La généralisation de la méthode *Dichotomious* passera en particulier par l'amélioration de calcul des coûts d'exclusion aux différents niveaux de l'arbre de découpe binaire. Cette amélioration n'entre d'ailleurs pas dans le cadre des travaux de cette thèse.

4.6 Conclusion

Ce chapitre nous a permis de passer en revue les différentes approches possibles pour séparer des données. Nous avons vu que les méthodes de clustering ne sont pas directement adaptées aux problèmes de tournées de véhicules. Cependant, il existe un ensemble de méthodes dans la littérature qui s'inspirent et adaptent des méthodes de clustering pour les appliquer au VRP. Ceci nous a permis de montrer plus en détails les mécanismes mis en place dans les méthodes *Découpe récursive* et *Dichotomious*.

Chapitre 5

Rich Soft Layered-Clustered-VRP

Comme évoqué dans le chapitre introductif, la qualité de service perçue par le chauffeur est directement liée au groupement des nœuds qu'il doit servir. Pour le planificateur, cette qualité est liée à la facilité d'exploitation du résultat. Ces deux points se matérialisent par des tournées dans lesquelles il existe des zones qu'un chauffeur doit servir entièrement et qui soient compactes géographiquement. Pour répondre à ces contraintes, nous devons alors envisager de constituer des tournées qui tiennent compte de ces besoins.

5.1 État de l'Art

Le Clustered VRP (CluVRP) a été introduit par Sevaux et Sörensen (2008) et est décrit comme suit : pour les VRP de très grande taille, il peut être intéressant de décomposer le problème en secteurs et d'utiliser les secteurs pour résoudre le problème de tournées de véhicules plutôt que les clients finaux individuellement.

Le terme secteur est utilisé pour désigner une zone géographique dans lequel des nœuds sont présents et regroupés. Le terme cluster, ici, fait référence à ces groupements, dont la création n'est pas mentionnée. Il s'agit d'une donnée du problème.

Dans cette variante, un cluster visité doit être terminé par le véhicule qui le parcourt avant de passer au secteur suivant. Ici, clusters et secteurs ne sont pas distingués. Dans la proposition de résolution associée, chaque secteur est représenté par un centre virtuel. Ces centres sont utilisés, avec une estimation de la charge de travail du secteur associé, pour former un VRP de haut niveau. Chaque tournée de haut niveau est associée à un véhicule.

Une telle formulation d'un problème de tournée de véhicules peut être rapprochée de la constitution d'un plan tactique pour l'affectation de créneaux horaires de livraisons telle que proposé par Hernandez *et al.* (2017).

L'objectif du CluVRP est alors de proposer une solution en deux parties : 1) donner un ordre de parcours des clusters par les véhicules et 2) donner un ordre de visite des

nœuds de chaque secteur.

Le parcours d'un cluster est formulé à l'aide d'un problème de plus court chemin Hamiltonien (shortest hamiltonian path - SHP) et modélisé par un TSP classique pour sa résolution. La résolution est réalisée à l'aide d'un modèle MILP pour sa résolution exacte. Une ouverture vers les méta-heuristique est proposée et concrétisée par la suite par Barthélemy *et al.* (2010). Vidal *et al.* (2015) ont contribué à l'exploration de cette variante avec une méthode hybride génétique. Battarra *et al.* (2014) ont quant à eux proposé des méthodes de type *Branch and Cut* et *Branch and Cut and Price*.

Des approches de résolution à deux niveaux ont été apportées par Expósito-Izquierdo *et al.* (2016) et Defryn et Sörensen (2017). Ces derniers ont également introduit la variante *CluVRP with weak cluster constraints* où il n'est pas impératif de satisfaire intégralement les demandes d'un cluster avant d'en parcourir un autre. Il est possible de servir les nœuds d'un cluster en plusieurs fois tant que le cluster est bien servi par un unique véhicule (Soft CluVRP).

Le CluVRP et le soft CluVRP sont actuellement des variantes du CVRP, dans lesquelles les nœuds à servir sont groupés par secteurs prédéfinis. Les fenêtres de temps par exemple sont seulement mentionnées par Hintsch (2021) en soulignant qu'elles pourraient avoir un rôle important dans le cas d'applications pratiques. En effet, le fait d'avoir des nœuds, d'un même cluster, répartis sur des fenêtres de temps disjointes soulève plusieurs questions. Dans le cas du CluVRP, il est nécessaire d'attendre entre ces ensembles plutôt que de parcourir un autre cluster.

Dans le cas du Soft CluVRP cela oblige à revenir dans le secteur, ce qui est autorisé, mais génère donc une augmentation significative de la fonction objectif dans la mesure où les différents groupes formés par les fenêtres de temps disjointes auraient été plus pertinents à servir par des véhicules différents. La situation est similaire dans le cas de retours de marchandises (Backhauls). En effet, ces retours sont à charger une fois les livraisons terminées. Également, il est possible qu'au sein d'un secteur, des clients soient incompatibles avec certains véhicules en raison de compétences ou d'impossibilité d'accès.

5.2 Proposition de définition

Traiter ensemble des nœuds qui sont proches est une demande légitime des chauffeurs. En effet, si deux nœuds sont sur le même segment de rue et qu'ils sont traitables par le même couple véhicule-chauffeur, alors ils doivent être traités par ce même couple.

Lorsque la tournée s'effectue dans une région peu dense, si un point se trouve à une distance réduite par rapport au reste de l'instance, il sera peu pertinent de faire venir un second véhicule dans cette région, puisque les temps et distances d'approches importants justifient de traiter ces nœuds avec le même véhicule. Cependant, même s'ils sont dans le même secteurs, ils peuvent ne pas être compatibles au moins pour une partie des véhicules. De même, l'association de ces nœuds peut ne pas être intéressante. En particulier si les créneaux horaires sont disjoints où s'il s'agit de Backhauls. Un secteur peut alors être découpé en plusieurs couches (*layers*) qui représentent autant de groupes d'incompatibilités potentielles ou de disjonctions sur certains attributs.

Nous souhaitons également pouvoir traiter un ensemble d'attributs en plus de la contrainte de capacité (Caceres-Cruz *et al.*, 2014). Certains attributs sont requis pour fournir une « solution technique minimale » afin que la variante soit intégrée au projet Optimizer-API ; ils sont décrits dans la section 5.2.1. Certains attributs sont intéressants, et sont directement disponibles du fait de leur présence dans le modèle utilisé pour résoudre le TSP en tant que sous-problème. Ils sont décrits dans la section 5.2.2.

5.2.1 Contraintes imposées

Les contraintes imposées correspondent à la fois à la définition des variantes CluVRP et Soft CluVRP, et aux besoins spécifiques de la logistique du dernier kilomètre.

Vehicle Capacity la capacité des véhicules est limitée,

Mutliproducts les véhicules peuvent transporter plusieurs types de produits,

Heterogeneous Fleet la flotte de véhicule est composée de différents types de véhicules avec des coûts et des capacités différentes,

Duration Constraints/Length la durée des routes est limitée et ne peut pas dépasser sa valeur maximale, en incluant le temps de service des nœuds,

Asymmetric Cost Matrix les matrices sont de nature asymétrique.

Vehicle Road Dependance certains véhicules ne peuvent pas traverser certains arcs du réseau routier. Cela a pour conséquence de rendre inaccessibles certains nœuds ou de modifier les temps et distances liées aux trajets.

Il est à noter également que l'association des attributs Heterogeneous Fleet et Vehicle Road Dependance implique de devoir considérer des matrices différentes pour les différents types de véhicules.

5.2.2 Contraintes additionnelles

Puisque nous traitons les problèmes au sein du projet Optimizer-API, nous avons accès aux solveurs interfacés. Nous pouvons alors résoudre des TSP et VRP. Nous avons identifié des attributs qui peuvent être nativement utilisés puisqu'ils demandent peu d'ajustement de la méthode de résolution que nous proposons dans ce chapitre. Bien qu'ils ne soient pas traités directement, la structure proposée pour la méthode de résolution anticipe leur intégration.

Time Windows les nœuds ont des créneaux horaires en dehors desquels il n'est pas possible de les servir,

Multiple Time Windows les nœuds ont de multiples créneaux horaires,

Vehicle Site Dependance certains véhicules ne peuvent pas accéder ou servir certains nœuds en raison d'incompatibilités,

Backhauls dans les tournées de livraison et de collecte, au sein d'une tournée les livraisons doivent être toutes réalisées avant de pouvoir débuter les collectes,

Multidepot il existe plusieurs dépôts depuis lesquels les véhicules partent et reviennent de leurs tournées,

Different End Locations/Open Routes les tournées débutent au dépôt et se terminent au dernier nœud visité.,

Different Start and End Locations les dépôts de départ et de fin de tournées peuvent être différents,

Departure from Different Locations les tournées peuvent commencer depuis différentes positions.

5.2.3 Définition des secteurs

Les propositions précédentes liées au CluVRP considèrent les secteurs comme des données du problème. Or, nous avons pu constater au travers de nos échanges avec les acteurs du dernier kilomètre que les réorganisations faisant appel à des méthodes d'optimisation pour les tournées de véhicules, s'accompagnent également d'une réorganisation complète des secteurs de ces mêmes tournées.

Une fois les outils d'aide à la décision en place, les sectorisations ne sont pas permanentes et demandent des ajustements réguliers. La définition de ces secteurs est problématique pour des activités pour lesquelles l'activité change quotidiennement.

La définition des secteurs et donc des clusters de nœuds doit faire partie intégrante de la proposition. La proposition doit tolérer des secteurs en tant que donnée d'entrée.

5.3 Construction des clusters

Le projet Optimizer-API accepte en entrée des polygones au format geojson. Les points, qui fournissent les coordonnées des nœuds, sont associés au polygone dont l'enveloppe les recouvrent. Cela correspond au fonctionnement énoncé pour le CluVRP et Soft CluVRP, où les secteurs sont des données du problème.

Cependant, les utilisateurs qui effectuent des requêtes au *web service* Optimizer-API n'ont pas nécessairement la possibilité de fournir des polygones qui correspondent à leur activité. Nous devons alors proposer un mode de construction des clusters qui corresponde aux besoins de la logistique du dernier kilomètre.

5.3.1 Secteurs topologiques

En matière de transport, passer d'un secteur à un autre peut être coûteux, non seulement parce qu'il pourrait exister des péages, mais parce que le temps et la distance pour changer de secteur sont importants. En effet, traverser un fleuve, une autoroute ou une voie ferroviaire ne se font que par un nombre limité d'axe routiers. Le passage d'un secteur à autre demande alors un détour important. Il est de fait peu probable, qu'à l'optimalité, 2 nœuds de part et d'autre de ces frontières topologiques [1] soient dans la même tournée et réalisés l'un après l'autre. En effet, il existe deux graphes, de part et d'autre d'une de ces frontières, qui ont une faible connexité, représentés par des ponts, des tunnels ou des embranchements.

Nous pouvons donc déjà établir un premier niveau de découpe pour distinguer des secteurs topologiques. À cet égard, nous avons à disposition un ensemble de jeux de données et d'outils nous permettant d'obtenir ces grands ensembles. L'API Overpass (Olbricht, 2015) permet d'interroger les données OpenStreetMap et d'extraire les informations dont nous avons besoin. À l'heure actuelle, les données suivantes sont demandées sous forme de polylines dans la zone géographique couverte par l'instance traitée :

way[**'waterway'='river'**] les fleuves, rivières et ruisseaux,

1. La topologie, ici, fait référence à l'application de la théorie des graphes à la géographie et en particulier de l'analyse des réseaux routiers

way[‘waterway’=‘canal’] les canaux,

way[‘highway’=‘motorway’] les autoroutes,

way[‘highway’=‘motorway_link’] les bretelles d’autoroutes,

way[‘highway’=‘trunk’] les voies rapides,

way[‘highway’=‘trunk_link’] les sorties de voies rapides,

way[‘natural’=‘coastline’] la ligne de côte en bord de mer.

Les polylines sont représentées par un ensemble de points reliés par des lignes, dont le tracé est ouvert. Cela nous permet d’extraire les axes fluviaux et canaux principaux, les autoroutes, les périphériques, les rocades et les échangeurs liés ainsi que les lignes de côtes. Lorsqu’une instance est reçue, nous déterminons un rectangle à partir des latitudes et longitudes extrêmes des coordonnées des points (*bounding box*) et nous interrogeons l’API Overpass pour obtenir les polylines des entités demandées. Ces polylines sont traitées de manière à générer des polygones à l’aide de l’extension PostGIS (Obe et Hsu, 2011) pour les bases de données PostgreSQL. Nous associons chaque point au polygone qui le contient. PostGIS propose un ensemble d’outils pour manipuler ces données géographiques.

Le résultat de cette séparation des secteurs topologiques est présenté dans la figure 5.1. Les frontières sont représentées par un trait violet. Chaque secteur formé est affiché dans une couleur différente.

5.3.2 Clusters par densité

Maintenant que les grands ensembles de points sont séparés en groupes cohérents du point de vue topologique, nous pouvons supposer que deux points suffisamment proches peuvent effectivement être traitées ensemble, même si nous n’avons pas encore accès aux distances via le réseau routier. En effet, puisque nous avons écarté les principales frontières topologiques, les points sont facilement accessibles les uns aux autres, en particulier en milieu urbain où le réseau routier est dense. Bien que les sens uniques puissent rendre certaines combinaisons peu pertinentes, si les points sont proches au regard de l’instance globale, alors il sera tout de même pertinent de les traiter avec le même véhicule.

Les points doivent, certes, être groupés par proximité, mais cette proximité est relative à la densité des points alentours. Puisque le CluVRP nécessite qu’un groupe puisse être traité par un seul véhicule, nous devons alors considérer les nœuds et plus uniquement les points, afin d’avoir accès aux attributs tels que les demandes ou les fenêtres de temps. De plus, nous ne savons pas à l’avance quel nombre de groupes nous souhaitons avoir.

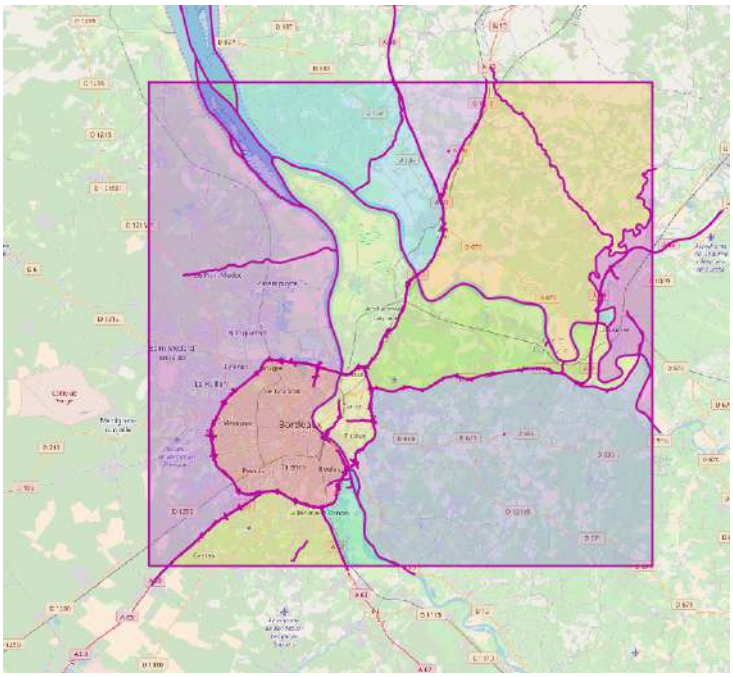

FIGURE 5.1 – Séparation des secteurs topologiques

Pour cet usage, DBSCAN semble être un bon candidat, à la fois parce qu'il permet de propager des clusters par densité, mais également parce que le nombre clusters n'est pas un paramètre d'entrée. Par ailleurs, les éléments isolés sont classés comme « bruit » et pourront donc être traités seuls, ce qui en fait un argument supplémentaire en faveur de DBSCAN.

Cependant, DBSCAN n'a pas de limite sur la taille des clusters qu'il peut agglomérer. Or, nous devons nous assurer que l'ensemble des nœuds d'un cluster aient leur demande satisfaite par un seul véhicule.

Pour cela, nous appelons DBSCAN de manière récursive. L'initialisation s'effectue avec un paramétrage large ; la distance de propagation ϵ peut représenter plusieurs kilomètres puisque nous voulons grouper les points en milieu rural même s'ils sont éloignés. Une fois l'algorithme DBSCAN terminé, nous calculons le cumul des attributs d'un cluster, en termes de demandes et de temps de service. Pour chaque cluster qui dépasse une capacité du véhicule type, nous réappliquons DBSCAN, mais avec une valeur de ϵ réduite. Nous avons choisi ici, le véhicule type comme un véhicule fictif qui représente les plus petites capacités de la flotte. La récursion est poursuivie jusqu'à ce que tous les clusters formés respectent la capacité du véhicule type, ou qu'une distance minimale soit atteinte. Si la distance minimale est atteinte et qu'il n'est pas possible de former un cluster inférieur à

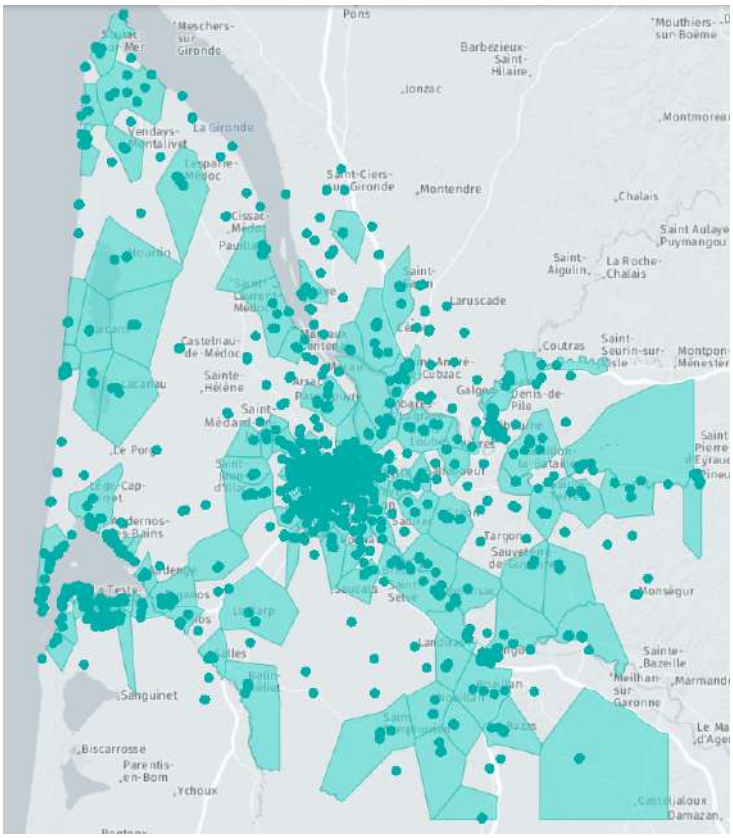

FIGURE 5.2 – Clusters par densité - Gironde (33)

la capacité du véhicule type, l'ensemble des éléments du cluster sont considérés comme « bruit ».

La figure 5.2 représente les clusters par densité obtenus avec la méthode DBSCAN récursive appliquée à l'instance présentée dans la section 5.9.1. La figure 5.3 centre la visualisation sur le centre-ville de Bordeaux. La visualisation des clusters se fait en générant dans un premier temps un diagramme de Voronoï, puis les différentes cellules d'un même cluster sont fusionnées. Nous pouvons constater qu'en zone rurale, les zones sont vastes. En effet, la densité de points est faible, nous pouvons alors propager librement les clusters. En zone urbaine, les clusters sont plus réduits, la récursivité a permis d'obtenir des clusters qui peuvent être servis par un unique véhicule.

5.3.3 Séparation par affinité

Il reste tout de même à assurer qu'un cluster puisse être réellement traité par un unique véhicule. C'est à la résolution du VRP de déterminer s'il est pertinent de grouper

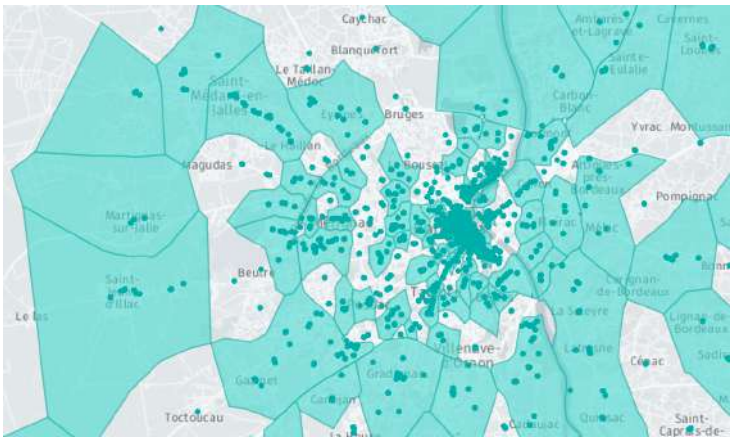

FIGURE 5.3 – Clusters par densité - Bordeaux

des nœuds ayant des propriétés hétéroclites : les nœuds peuvent avoir des fenêtres de temps disjointes, des quantités qui ne sont présentes simultanément que sur quelques véhicules, ou des *skills* qui ne sont pas forcements compatibles. En effet, la décision d'affecter des nœuds avec des propriétés différentes ne peut se faire de manière locale. Elle demande une réflexion au niveau de la totalité de l'instance. Nous pouvons cependant, bénéficier de la sectorisation topologique et du clustering effectués précédemment pour donner des indications qui seront utiles pour la résolution qui en découlera. Nous devons alors séparer les clusters formés par densité en plusieurs couches indépendantes.

En effectuant cette succession d'opérations de sectorisation géographique, de clustering par densité et de séparation par affinité, nous pouvons à la fois grouper des nœuds qui sont relativement éloignés dans des régions peu denses et avoir un niveau de finesse important dans des régions très denses. Les nœuds ayant des propriétés similaires forment un groupe de nœuds par affinité. Un cluster sera alors composé de plusieurs groupes distincts, mais dont la couverture géographique sera similaire.

5.4 Préparation des groupes

Nous envisageons de traiter le problème avec deux niveaux. À haut niveau, on ne souhaite pas manipuler le problème dans son ensemble et les groupes doivent être représentés que par un seul nœud. À bas niveau, nous devons pouvoir utiliser les nœuds tels que définis dans le problème initial, mais dans un volume restreint.

5.4.1 Point médian des clusters

À haut niveau, nous devons définir comment sont représentés les groupes, en particulier pour le calcul des matrices.

Puisque nous manipulons des données qui sont projetées sur le graphe routier, nous ne pouvons pas utiliser le barycentre (au sens géométrique). En effet, utiliser le barycentre pourrait mener à le projeter sur une voie peu praticable, ou sur un graphe disjoint (i.e : cour intérieure, parc), le rendant inaccessible. Pour assurer la conservation de l'accessibilité, nous devons utiliser un point existant.

De plus, puisqu'il est possible d'avoir plusieurs couches de nœuds dans un cluster est-il pertinent de les représenter par des points différents ? Les représenter avec des points différents implique d'augmenter la taille des matrices utilisées à haut niveau. De plus, envoyer un même véhicule sur un seul secteur ne doit pas générer de détour supplémentaire. Nous pouvons donc supposer que les multiples groupes qui composent un cluster peuvent être représentés par un même point.

Puisqu'il doit s'agir d'un point existant, nous avons décidé d'utiliser le médoïde du cluster, soit le point qui minimise sa distance à tous les autres points du cluster. Par abus de langage, nous le désignons comme point « médian ».

5.4.2 Approximation de temps de parcours des groupes

Afin d'estimer le temps nécessaire pour satisfaire l'ensemble des demandes des nœuds formant un groupe ainsi que la faisabilité de ce sous-tour, nous définissons le point médian du cluster comme étant un dépôt. Nous choisissons un véhicule adapté et effectuons une résolution d'un TSP avec toutefois l'ensemble des attributs attendus (capacités, fenêtres de temps, etc). De cette manière, nous pouvons filtrer les nœuds qui seraient malgré les étapes précédentes toujours incompatibles avec les autres nœuds, formant la couche, obtenus après séparation par affinité. Les nœuds présents dans ce tour forment nos groupes finaux. Le résultat de ce TSP nous donne une estimation haute de la durée nécessaire à la réalisation de ce groupe pour la résolution de haut niveau. Dans le cas d'une flotte hétérogène, dont les restrictions sur le graphe routier sont différentes, nous avons choisi d'appliquer au sous-tour les temps de trajet propres à chaque type de véhicule. Le résultat de cette étape est représenté par la figure 5.4.

Il pourrait être envisagé d'appliquer un VRP pour le cluster entier plutôt qu'un TSP au niveau des couches. De cette manière les couches compatibles seraient regroupées.

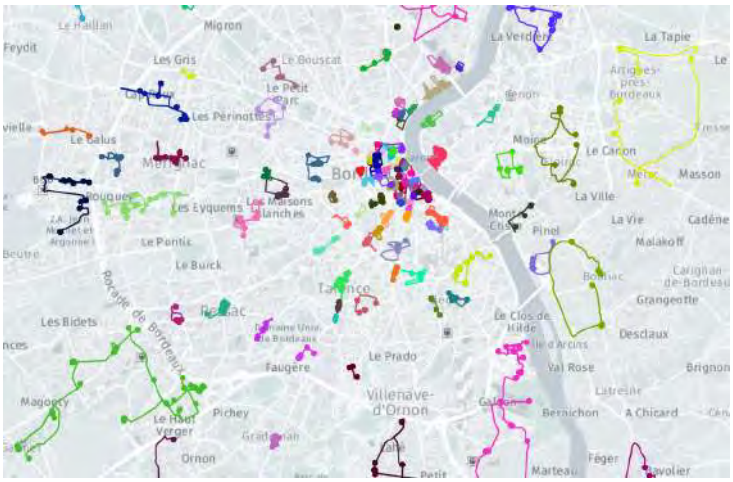

FIGURE 5.4 – Approximation de temps de parcours des groupes - Bordeaux

Cependant, il faut s'assurer que cette compatibilité soit réalisable au niveau du problème dans sa globalité. À cet égard, l'application d'un TSP pour chaque profil de véhicule semble plus appropriée. Les nœuds qui ne peuvent être affectés dans la résolution du sous-tour ne sont pas inclus dans ces estimations et sont considérés comme des nœuds indépendants.

5.5 Initialisation de la résolution

Avec les étapes précédentes, nous avons maintenant des groupes de nœuds ainsi qu'une estimation du temps nécessaire pour satisfaire leurs demandes. Nous avons également un ensemble de nœuds indépendants. Pour initialiser la résolution, nous avons besoin de structurer les tournées. Le nombre de nœuds indépendants peut être important si l'instance a une répartition uniforme. En effet, il est difficile de séparer les éléments à l'aide de DBSCAN qui sur ces configurations aura soit du mal à propager les clusters, soit à constituer des clusters de taille inférieure à la capacité demandée. La séparation du problème en deux niveaux permet justement de réduire le nombre de nœuds à gérer à haut niveau et de traiter les tournées séparément à bas niveau.

Nous ne considérons pour l'initialisation que les groupes. Il pourrait être également intéressant d'intégrer les nœuds indépendants les plus contraints ou qui pourraient être structurants pour les tournées. Les capacités des véhicules sont ajustées de manière à avoir un remplissage maximal équivalent au rapport des demandes des nœuds considérés sur la demande totale.

Ce VRP initial peut alors être résolu, soit directement s'il est de taille réduite, soit à l'aide des approches *Découpe récursive* ou *Dichotomious* présentées dans les sections 4.5.1 et 4.5.2 s'il est long ou difficile de traiter ce VRP obtenu directement. L'idée est d'avoir une solution initiale de bonne qualité où les groupes sont répartis de manière à constituer des tournées cohérentes. Les tournées ainsi obtenues sont présentées dans la figure 5.5.

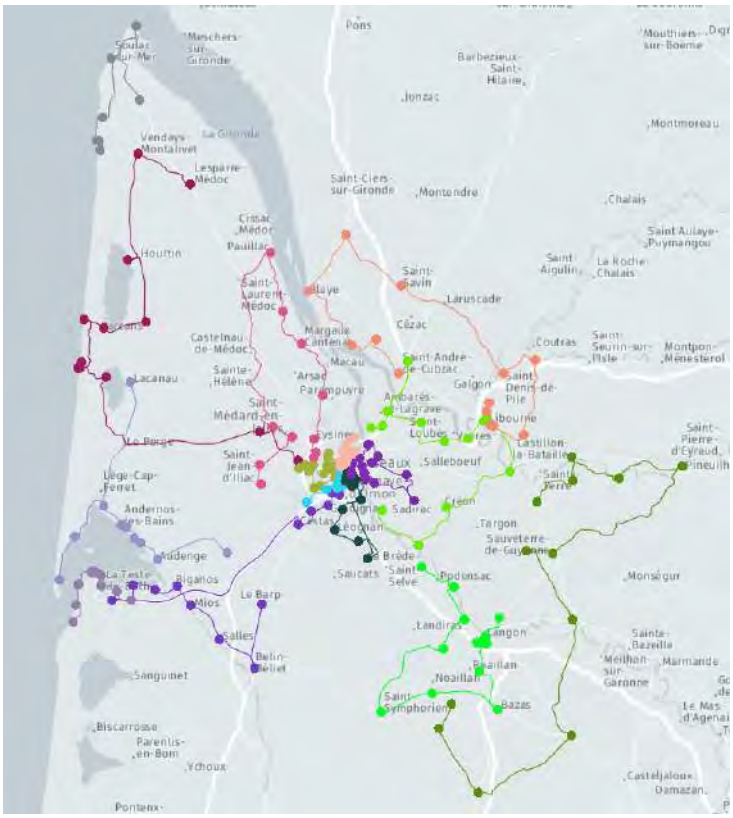

FIGURE 5.5 – Initialisation de la résolution à partir des nœuds médians

Les nœuds indépendants sont, une fois cette solution obtenue, insérés de manière gloutonne dans les tournées construites à l'aide de l'opérateur de réparation présenté dans la section 5.7.3. La solution initiale est à la fois composée des groupes de nœuds et des nœuds indépendants. Il est possible qu'au terme de cette phase il reste des groupes ou des nœuds indépendants qui ne soient pas affectés à des tournées. Un nœud indépendant, est maintenant considéré comme un groupe constitué uniquement d'un nœud. Les nœuds indépendants sont alors placés dans une liste de groupes non affectés. Ils seront insérés ultérieurement à l'aide durant la phase réparation de la résolution (Section 5.7.3).

5.6 Fonction objectif

Maintenant que le problème est prêt à être résolu, nous pouvons déterminer la fonction objectif pour guider la recherche de solution.

5.6.1 Pénalité de non-affectation

Avoir des groupes non affectés peut être nécessaire en début de résolution, si le problème est extrêmement contraint. Il peut également ne pas exister de solution réalisable avec l'ensemble des nœuds. Puisque les groupes ont déjà subi une première phase pour déterminer la faisabilité des sous-tours, on peut considérer légitime de considérer que si un nœud d'un groupe est affecté, alors les autres doivent l'être aussi. Nous devons alors pénaliser fortement la fonction objectif si un groupe est partiellement non affecté dans un problème de bas niveau.

Il est d'ailleurs préférable que les groupes ne soient affectés à aucune tournée, plutôt que partiellement affecté dans une tournée. En effet, si un groupe n'est affecté à aucune tournée, nous pouvons appliquer un unique opérateur pour l'insérer. Alors que si des nœuds sont déjà affectés à une tournée, il faut placer un ou plusieurs groupes dans la liste des groupes non affectés et les réaffecter ensuite, sans garantie que la tournée de laquelle ils ont été extraits ou celle de leur réaffectation soit au final réalisable.

La pénalité de non-affectation d'un groupe doit donc être minime face à la pénalité de non-affectation d'un nœud dans une tournée, mais suffisamment importante pour forcer leur intégration progressive à la solution.

5.6.2 Coût des tournées

Le coût d'une tournée intègre à la fois la pénalité de non-affectation des nœuds ainsi que le coût de la tournée à proprement parler. Puisque chaque tournée est traitée indépendamment, il n'est pas nécessaire de considérer le coût fixe d'utilisation du véhicule à ce niveau. L'objectif ici est de réduire la durée ou le kilométrage des tournées en fonction des pondérations fournies à chacune de ces dimensions dans le problème pour le véhicule associé à la tournée.

5.6.3 Coût de la solution

Le coût de la solution guide la résolution. En effet, nous devons à chaque itération déterminer si les opérations appliquées sont pertinentes et permettent d'améliorer la solution courante. Le coût fixe lié à l'utilisation des véhicules est intégré à ce niveau et couplé au coût des tournées et à la pénalité de non-affectation des groupes. Le coût des tournées intègre déjà la pénalité de non-affectation des nœuds.

5.7 Résolution

Une fois la solution initiale obtenue et la fonction objectif déterminée, nous pouvons mettre en place les mécanismes pour rechercher de nouvelles solutions et guider cette recherche.

L'évaluation des tournées individuelles est réalisée à l'aide d'un TSP sur chacune des tournées. Lors de l'évaluation de ces tournées de bas niveau, l'ordre dans lequel les nœuds médians de chaque groupe sont parcourus est enregistré. Puisque le coût en temps d'évaluation est important, les opérations de haut niveau, doivent manipuler les nœuds médians et les approximations calculées à l'aide des mécanismes décrits dans les sections 5.4.1 et 5.4.2.

Un ensemble d'opérateurs est présenté dans la section 5.7.2 pour perturber la solution courante. Ces opérateurs peuvent échanger des groupes ou en supprimer des tournées. L'opérateur de réparation présenté dans la section 5.7.3 permet de réaffecter des groupes à des tournées. L'ensemble de ces opérateurs vérifient un ensemble de contraintes avant d'affecter des groupes à des tournées. Le cumul des quantités est calculé afin d'assurer qu'il n'y aura pas de dépassement des capacités. Lors de l'insertion de groupe à des tournées, la durée de celles-ci est augmentée en fonction de l'estimation du coût du détour que cela engendre. L'ordre des nœuds médians est utilisé pour déterminer la position estimée du groupe dans la tournée lors de son insertion. À chaque permutation, suppression ou insertion, l'ordre des groupes est ajusté et la durée des tournées est mise à jour.

5.7.1 Gestion des matrices

Nous avons vu dans la section 3.9.1 que le calcul des matrices peut être coûteux en termes de temps de calcul. La séparation du problème en deux niveaux nous permet de réduire la taille de la matrice à haut niveau, puisque nous ne manipulons qu'un ensemble

réduit de nœuds : les nœuds médians. Cette matrice est nécessaire à la fois pour la phase d'initialisation (Section 5.5) et pour l'application des opérateurs. En effet, pour déterminer la pertinence des opérateurs nous avons besoin d'estimer l'impact sur la fonction objectif des changements appliqués.

À bas niveau, nous ne manipulons que des tournées individuelles. Nous n'avons pas besoin de la matrice complète. Pour répondre à cette problématique, des matrices vides sont initialisées ; elles sont complétées à mesure que des combinaisons de groupes sont associés à des tournées. De cette manière, nous ne demandons à Router que les matrices utiles à la résolution des sous-problèmes effectivement rencontrés.

5.7.2 Opérateurs de perturbation

Les opérateurs de perturbation doivent permettre d'effectuer des échanges de groupes entre tournées. En effet, l'évaluation des tournées est déléguée à des méthodes de résolution élaborées qui déterminent l'ordre dans lequel sont parcourus les nœuds. Les opérateurs n'implémentent alors que des échanges inter-routes.

Les opérateurs *Relocate* et *Swap* sont inspirés des opérateurs inter-routes classiques pour les métaheuristiques utilisant des méthodes de recherches locales (ALNS, Tabou, GRASP).

Relocate Un groupe est retiré de la tournée à laquelle il est associé et il est replacé dans une autre tournée de manière à réduire son coût d'insertion.

Swap Deux groupes, de routes différentes, sont échangés à leurs positions respectives.

Nous avons également exploré les opérateurs de la littérature relative au VRP à deux échelons (2E-VRP) et en avons adaptés quelques-uns à notre cas d'usage.

Random removal Cet opérateur, introduit par Ropke et Pisinger (2006), sélectionne aléatoirement q nœuds et les place dans la liste des non affectés. Dans notre cas d'utilisation, nous retirons des groupes entiers plutôt que de simples nœuds.

Related removal Un groupe est sélectionné aléatoirement et les $q-1$ groupes les plus proches sont identifiés. Cet ensemble de groupe est retiré de la solution courante et placés dans la liste des groupes non affectés (Hemmelmayr *et al.*, 2012).

Route removal Une route aléatoire est supprimée de la solution, les groupes associés sont ajoutés à la liste des groupes non affectés (Hemmelmayr *et al.*, 2012).

Worst removal Cet opérateur est également adapté des travaux de Ropke et Pisinger (2006). Les q nœuds dont le retrait génèrent le plus grand gain sont enlevés de la solution. Dans notre cas, nous appliquons cet opérateur aux groupes, en calculant leur coût de suppression de la tournée de haut niveau. Le gain est calculé en comparant le coût avec et sans ce groupe dans la tournée. Ce gain est pondéré à l'aide de la distance aux k plus proches voisins. Ceci afin d'éviter que les groupes situés loin des autres ne soient systématiquement sélectionnés.

Biased removal Pour chaque groupe, le gain associé à son retrait est calculé. Puis la sélection aléatoire est biaisée à l'aide de ce gain. Plus le gain est important, plus les chances de sélectionner ce groupe est important (Breunig *et al.*, 2016).

Single cluster removal Breunig *et al.* (2016) ont proposé un opérateur pour supprimer les routes composées d'un unique nœud. Nous avons adapté cet opérateur afin de supprimer les routes composées d'un unique groupe.

De plus, nous avons mis en place des opérateurs supplémentaires afin de bénéficier des particularités de la structure proposée.

Related route removal Une logique similaire à Related removal est appliquée. Cependant, nous sélectionnons aléatoirement une route. Puis, nous identifions les $q - 1$ routes les plus proches et retirons les groupes de celles de la solution courante pour les placer dans la liste des groupes non affectés.

Loop removal Nous avons constaté lors de nos expérimentations qu'une partie des routes avait tendance à se croiser plusieurs fois. Nous avons alors mis en place un opérateur qui détecte les tournées qui parcourent une partie d'un groupe, s'en éloignent puis y reviennent après avoir parcouru plusieurs autres groupes. Les groupes concernés sont alors retirés des routes auxquels ils sont affectés et placés dans la liste des groupes non affectés.

Combine high level routes Puisque nous pouvons représenter chaque groupe par un nœud qui possède toutes les propriétés du groupe et que nous avons les véhicules qui y

sont associées. Il est alors possible de résoudre un VRP de haut niveau afin de recombiner les tournées de haut niveau. Cet opérateur sélectionne entre 2 et 4 tournées et résout le VRP simplifié ainsi créé. Une fois la résolution du VRP simplifié terminée, les groupes sont affectés à la tournée dans laquelle leur nœud médian est affecté. Si le nœud médian n'est pas affecté, le groupe est placé dans la liste des groupes non affectés.

5.7.3 Réparation de solution

La méthode de résolution pour le 2E-VRP de Hemmelmayr *et al.* (2012) propose, suite à la phase de suppression, un ensemble d'opérateurs pour la phase de réparation des tournées. Ici, nous n'avons retenu qu'un opérateur glouton *Greedy Insertion* ainsi qu'une variante incluant un coefficient perturbateur afin d'insérer les groupes à la position la moins coûteuse. En effet, les autres opérateurs de réparation se basent sur les mécanismes d'ouvertures ou de fermetures de satellites pour le 2E-VRP que nous ne retrouvons pas dans le CluVRP.

5.8 Phase d'évaluation

Une fois les groupes associés aux tournées, nous pouvons les étendre afin de construire les sous-problèmes complets. Actuellement, nous faisons appel soit appel à VROOM, soit à OR-Tools et au modèle PPC présenté. La résolution des TSP de chaque tournée n'est déclenchée que pour les tournées ayant subi une modification par les opérateurs de haut-niveau.

À chaque fin de phase d'évaluation, la fonction objectif est calculée et comparée à la meilleure solution déjà trouvée. Si le coût est amélioré, la solution courante devient celle-ci et elle est enregistrée comme nouvelle meilleure solution. Si le coût est légèrement dégradé [2], la résolution est poursuivie à partir de la solution courante. Si la solution obtenue a un coût dégradé plus important, elle est supprimée et nous repartons de la meilleure solution connue.

Si la résolution est poursuivie, l'ordre de parcours des nœuds médians est mis à jour.

La résolution se poursuit jusqu'au nombre maximal d'itérations ou jusqu'à la durée maximale autorisée.

2. Actuellement la résolution est poursuivie si la dégradation de la solution est inférieure à 2%.

Type	Léger	Moyen	Lourd
Nombre	15	10	10
Profil	VL	VL	VL
Coef de vitesse	1.0	0.9	0.6
Coût horaire	10	13	17
Coût fixe	500	1200	2400
Capacité	500	1250	3000
Amplitude horaire	8-16h	8-16h	8-16h

TABLE 5.1 – Instance Bordeaux 2875 : profils des véhicules

5.9 Expérimentation numérique

5.9.1 Instance d'évaluation

Cette méthode de résolution a principalement été évaluée sur une instance construite à l'aide du générateur présenté dans le chapitre 6. Cette instance est construite en utilisant comme zone géographique la Gironde (33). L'ensemble des 2875 nœuds du problème sont des boutiques qui étaient référencées au moment de la création de l'instance dans OpenStreetMap (Figure 5.6). Les nœuds ont une forte densité dans le centre-ville de Bordeaux, cette densité diminue en s'éloignant de ce centre. Il existe des zones géographiques où la densité est également importante autour du bassin d'Arcachon, de Langon et de Libourne. La durée de visite associée à chaque nœud est fixée à une minute ; la distribution des quantités est présentée dans la figure 5.7. Cette distribution correspond à la taille du nom de chaque enseigne, ou 1 si le nom n'était pas renseigné. La flotte est composée de 35 véhicules répartis sur 3 profils présentés dans la table 5.1. Le dépôt est fixé aux bureaux de R&D de Mapotempo dans le quartier Belcier de Bordeaux.

5.9.2 Évaluation des opérateurs

Les tables 5.2, 5.3, 5.4 représentent les performances de opérateurs sur plusieurs répétitions, sur des intervalles d'itérations respectivement, 1 à 100, 101 à 200 et 201 à 500. Ce découpage se justifie par l'initialisation de la méthode de résolution. En effet, sur les premières itérations, il existe un ensemble de groupes qui n'ont pas encore été affectés. Il faut alors un certain nombre d'itérations pour qu'ils soient tous insérés dans une tournée et que l'ensemble des nœuds soient actifs.

Pour l'intervalle de 1 à 100 itérations, les opérateurs ont un gain moyen relativement

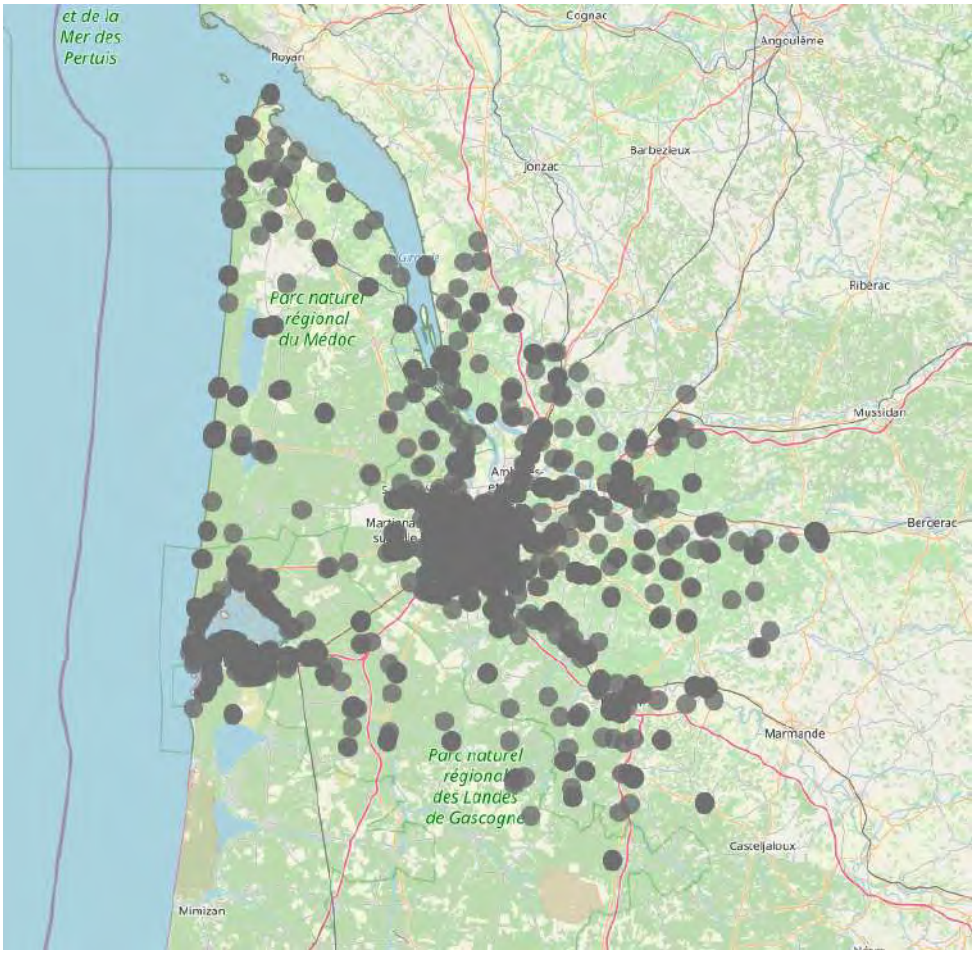

FIGURE 5.6 – Distribution des nœuds. Map data © OpenStreetMap

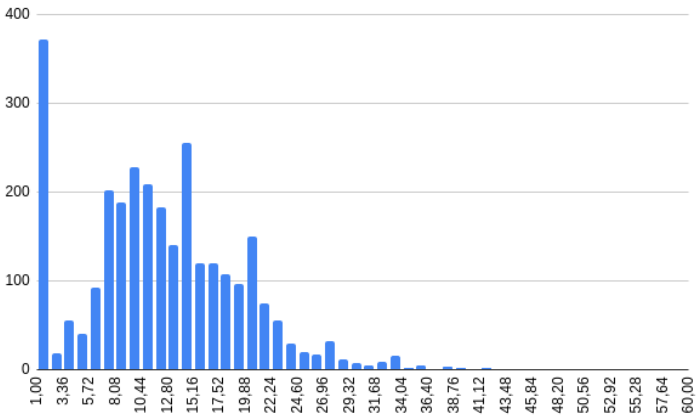

FIGURE 5.7 – Distribution des quantités

important, en effet, il est relativement facile d'insérer des groupes qui étaient inactifs jusqu'alors. La descente de la fonction objectif entre les itérations 1 et 100 est représentée par la figure 5.8. Les courbes de différentes couleurs représentent les répétitions exécutés pour la même instance afin d'affiner l'analyse sur les opérateurs.

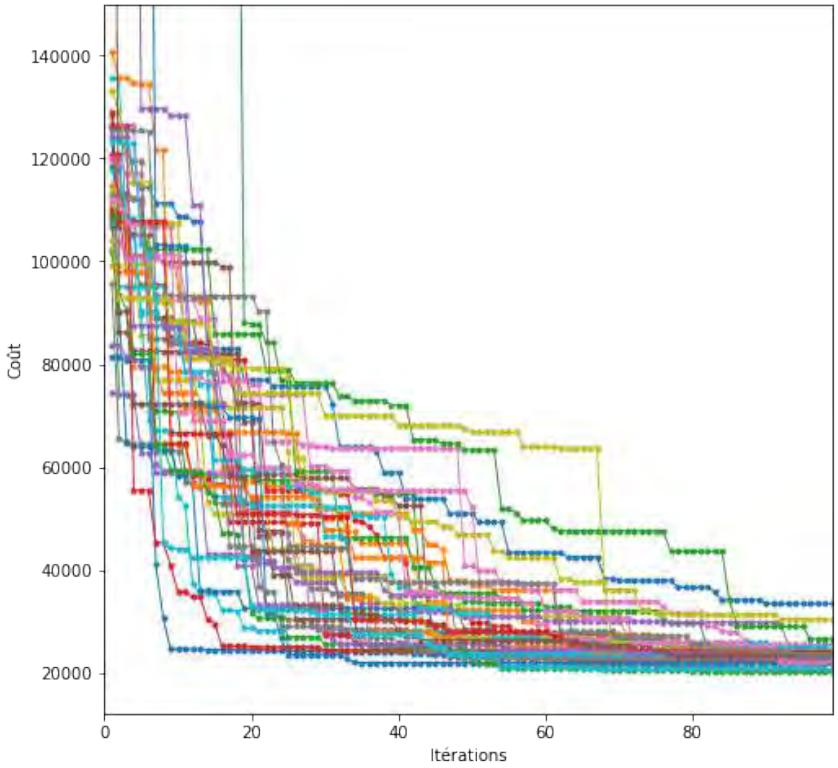

FIGURE 5.8 – Graphe de descente centré sur les itérations 1 à 100

Dans l'intervalle 101 à 200, sur la plupart des répétitions, l'ensemble des groupes est affecté à un véhicule qui peut le servir et la descente est encore relativement rapide. Au-delà des 200 itérations, la descente est plus lente. Nous pouvons alors identifier les opérateurs les plus pertinents pour ce mode de résolution. La colonne « Améliorations » montre le nombre de fois où l'utilisation d'un opérateur a amené à améliorer la solution courante. La colonne « Dégradations » montre le nombre de fois où l'utilisation d'un opérateur a dégradé la solution et amené à recharger la meilleure solution. La colonne « Total » montre le nombre total d'appels d'un opérateur. La colonne « Gain moyen » montre le gain relatif moyen lorsque la solution est améliorée à l'aide d'un opérateur.

Nous pouvons constater que les opérateurs *unassigned cluster removal*, *single cluster removal* et *route removal* améliorent très rarement la solution, cependant le gain apporté

Opérateur	Améliorations	Dégradations	Total	Gain moyen
biased removal	131	187	365	-2.51 %
combine high level routes	373	279	754	-10.68 %
loop removal	87	91	358	-0.90 %
related removal	111	129	347	-1.33 %
relocate	100	114	322	-0.70 %
route removal	1	29	35	0.00 %
single cluster removal	1	45	53	-2.50 %
swap	163	106	350	-0.41 %
unassigned cluster removal	396	108	604	-8.51 %
worst removal	77	61	352	-0.66 %

TABLE 5.2 – Itérations inférieures à 100

Opérateur	Améliorations	Dégradations	Total	Gain moyen
biased removal	124	202	404	-0.39 %
combine high level routes	250	396	808	-0.81 %
loop removal	62	72	384	-0.05 %
related removal	141	106	434	-0.07 %
relocate	148	97	402	-0.07 %
route removal	2	32	36	-3.66 %
single cluster removal	4	38	47	-2.39 %
swap	158	100	389	-0.03 %
unassigned cluster removal	50	89	240	-4.66 %
worst removal	44	46	417	-0.04 %

TABLE 5.3 – Itérations entre 100 et 200

est assez élevé. Cela peut s'expliquer par le manque de d'occurrence. En effet, les taux sont alors peu représentatifs. Nous avons tout de même décidé de conserver l'opérateur *unassigned cluster removal* puisqu'il est particulièrement efficace sur les premières itérations. Les opérateurs *worst removal*, *loop removal* ont un gain moyen très faible ; de plus, ils sont appelés relativement souvent, mais les taux d'amélioration et de dégradation sont bas. On peut alors considérer qu'ils ont peu d'impact sur la solution.

Une fois ces opérateurs filtrés, nous avons pu relancer un ensemble de résolutions afin d'obtenir la table 5.5 et le graphe de descente de la fonction objectif représenté dans la figure 5.9. Nous pouvons constater que l'opérateur *combine high level routes* a un gain moyen beaucoup plus importants que les autres opérateurs (hormis *unassigned cluster removal* qui n'est ici utilisé que pour retirer les éventuelles clusters non affectés). De même, nous pouvons nous interroger sur le fait que moins d'un appel sur quatre à ces opérateurs améliore la solution courante. Nous supposons que ceci est la conséquence de l'estimation haute de la durée de parcours des groupes associés à l'utilisation des points médians pour représenter les groupes. En effet, il est peu probable qu'une tournée doive

Opérateur	Améliorations	Dégradations	Total	Gain moyen
biased removal	282	432	1094	-0.39 %
combine high level routes	633	1176	2490	-0.81 %
loop removal	159	119	1177	-0.05 %
related removal	318	244	1257	-0.07 %
relocate	397	235	1284	-0.07 %
route removal	3	75	94	-3.66 %
single cluster removal	6	89	102	-2.39 %
swap	483	206	1189	-0.03 %
unassigned cluster removal	17	173	213	-4.66 %
worst removal	105	104	1216	-0.04 %

TABLE 5.4 – Itérations supérieures à 200

atteindre le point médian avant de parcourir l'ensemble des nœuds d'un groupe.

Opérateur	Améliorations	Dégradations	Total	Gain moyen
biased removal	1175	2329	5700	-0.11 %
combine high level routes	1342	3549	6569	-0.70 %
related removal	1383	1102	6368	-0.04 %
relocate	1635	846	6346	-0.01 %
swap	2209	684	6145	-0.01 %
unassigned cluster removal	17	1	23	-3.29 %

TABLE 5.5 – Itérations supérieures à 200 avec opérateurs filtrés

5.10 Conclusion

Nous avons pu voir dans ce chapitre une nouvelle méthode pour traiter les variantes riches du Soft CluVRP. Un outil pour l'appliquer à des instances réelles sans pré-requis de zones est proposée sous la forme d'un appel aux données OSM puis de l'application d'une méthode de clustering. Une fois les clusters formés, les nœuds sont séparés afin de constituer des groupes par compatibilité.

Un ensemble d'opérateurs est présenté afin de bénéficier de la structure du problème. Nous avons pu isoler un ensemble restreint d'opérateurs qui sont pertinents. Cependant, l'approximation faite pour estimer la durée nécessaire à un groupe et les temps de parcours inter-groupes pénalisent l'application de ces opérateurs qui ont alors des difficultés à améliorer la solution courante. Malgré ces difficultés, cette étude a permis de sélectionner les opérateurs les plus pertinents pour notre problème.

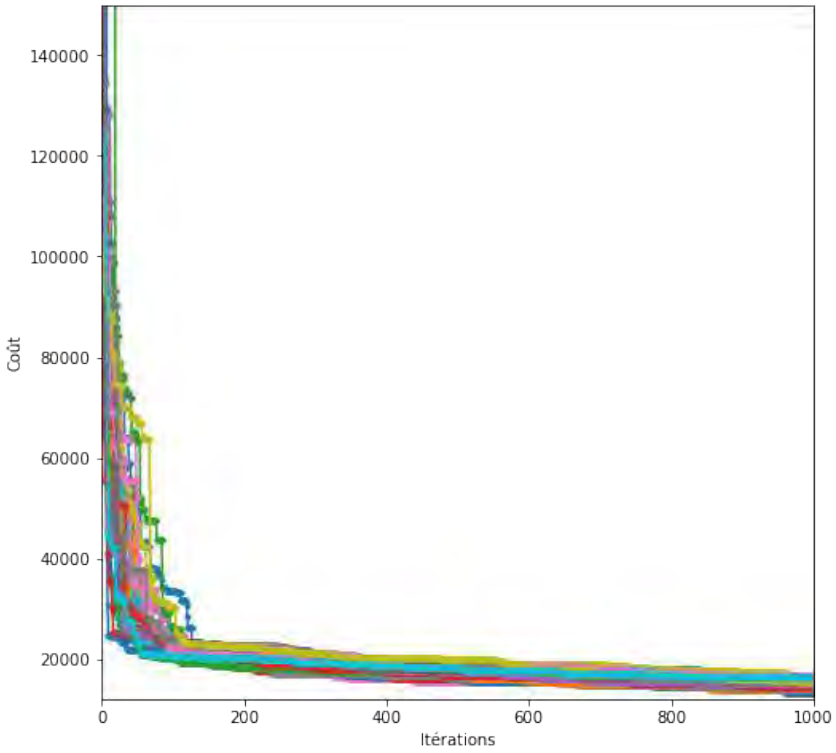

FIGURE 5.9 – Graphe de descente

Chapitre 6

Générateur d'instances

Ce chapitre présente des travaux conduits en marge de la thèse qui permettent de proposer à la communauté un générateur d'instances.

6.1 Analyse des instances de la littérature

Afin de comparer la pertinence et les performances des méthodes de résolution, il est nécessaire d'avoir des données qui soient facilement diffusables et utilisables. Pour échanger autour du VRP, la communauté à mis en place un ensemble d'instances qui font toujours office de référence.

Solomon (1987) a proposé des instances pour le VRPTW chacune composée de 100 nœuds. Fisher (1994) a proposé des instances pour le CVRP ayant de 44 à 134 nœuds. Rochat et Taillard (1995) ont proposé des instances (de 75 à 385 nœuds) pour lesquelles les nœuds sont répartis soit de manière aléatoire (Figure 6.3) soit autours de points, formant ainsi des clusters (Figure 6.2). Golden *et al.* (1998) ont proposé un ensemble d'instances (de 240 à 480 nœuds) pour lesquelles les nœuds à servir sont répartis sur des cercles concentriques (Figure 6.1).

Ces instances historiques sont limitées aux attributs étudiés au moment de leur création. Elles sont également limitées en nombre de nœuds (jusqu'à 480 nœuds). En effet, les performances des machines et les méthodes ne pouvaient alors pas traiter des problèmes de grande taille. Par la suite, la communauté a exploré de nouveaux attributs et puisque les performances des méthodes de résolution ont augmenté. Alors de nouvelles instances étaient nécessaires pour suivre ces nouveaux besoins.

Ainsi, les instances de Solomon (1987) ou Fisher (1994) ont servi de base pour construire des instances avec davantage de nœuds ou avec de nouveaux attributs. C'est en particulier le cas des instances proposées par Brandão (2011) ou Belmecheri *et al.* (2013). Les instances de Golden *et al.* (1998) ont également été modifiées par Li *et al.* (2007).

Ces instances sont peu représentatives des cas rencontrés dans la logistique du dernier

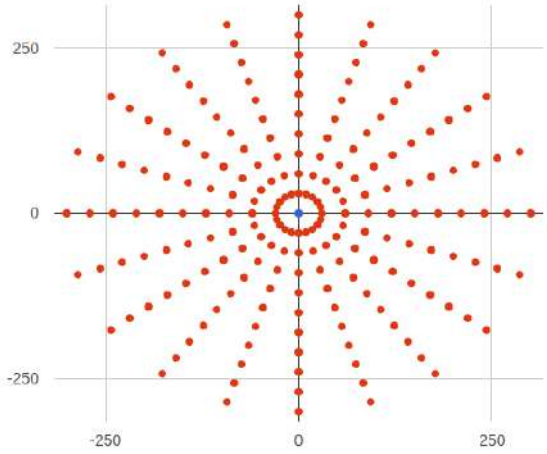

FIGURE 6.1 – Instance Golden 5

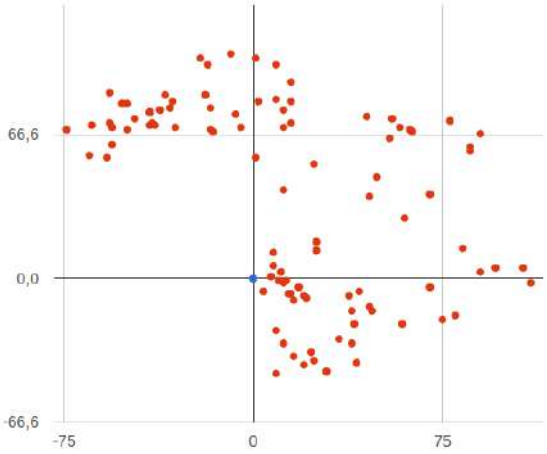

FIGURE 6.2 – Instance Taillard 100a

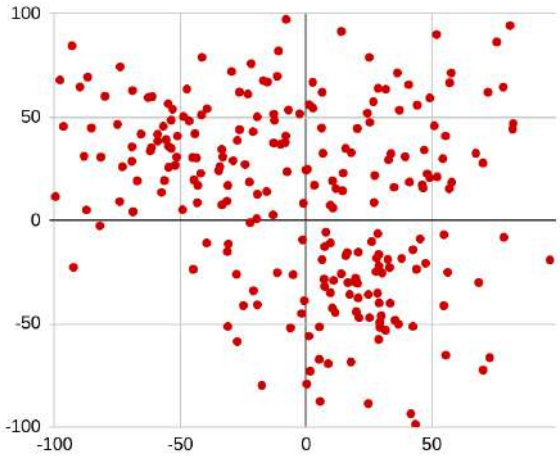

FIGURE 6.3 – Instance Taillard c100

kilomètre puisque les nœuds sont disposé dans une espace en deux dimensions dans lequel ils sont distribués, soit de manière géométrique, soit de manière aléatoire. Ces configurations représentent mal les distributions de points qui peuvent être rencontrées en réalité. Les distances utilisées sont pour la majorité des distances euclidiennes.

Sur ce point, Duhamel *et al.* (2011) ont proposé des instances construites à partir des villes de chaque département français et ont généré une matrice symétrique à partir des API proposées par Google pour le calcul de matrices au travers du réseau routier (Figure 6.4). L'une des particularités de ces instances est que l'inégalité triangulaire n'est pas respectée. En effet, les itinéraires sont calculés « au plus rapide », ce qui correspond au plus court chemin en temps de transport entre deux nœuds.

Cependant, la distance renseignée dans la matrice est kilométrique. Ainsi, il n'est pas assuré que la distance entre les nœuds a et b soit plus courte que d'effectuer les deux itinéraires $a - c$ et $c - b$. En réalité cela peut s'illustrer par le fait que réaliser un trajet par une voie rapide se fait à une vitesse plus élevée : un véhicule parcourt alors une plus grande distance sur une durée plus courte (Figure 6.5) ; deux petits trajets sur des voies de ville seront plus lent : un véhicule parcourt des petites distances avec une vitesse réduite et donc un temps de trajet allongé (Figure 6.6).

Cette contribution se fait le témoin de l'évolution des outils de cartographie et de leur implication pour le VRP. L'objectif de gagner en réalisme sur les instances passe également par une augmentation de la taille des instances, ce qui correspond à la fragmentation de la demande dans la logistique du dernier kilomètre. Ainsi Arnold *et al.* (2019) proposent

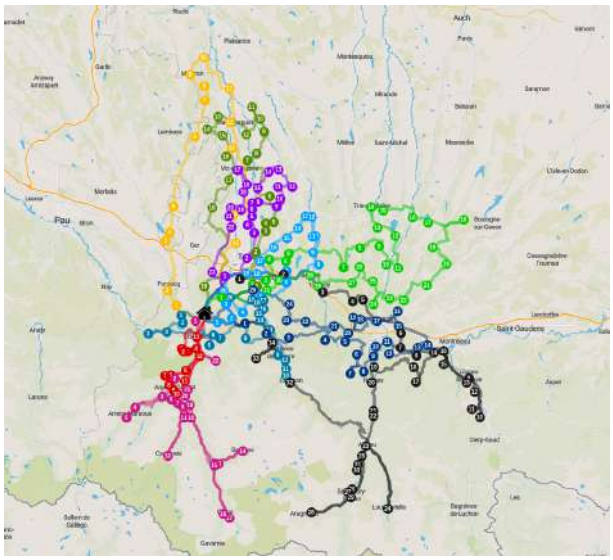

FIGURE 6.4 – représentation de l'instance DLP pour les Hautes-Pyrénées (65). Map data © OpenStreetMap

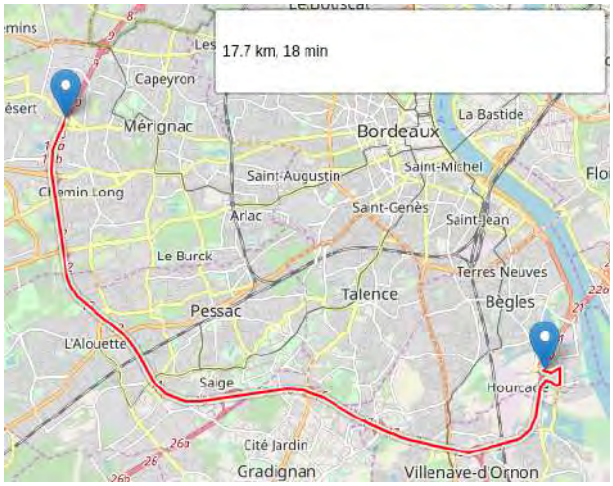

FIGURE 6.5 – Calcul d'itinéraire au plus rapide. Map data © OpenStreetMap

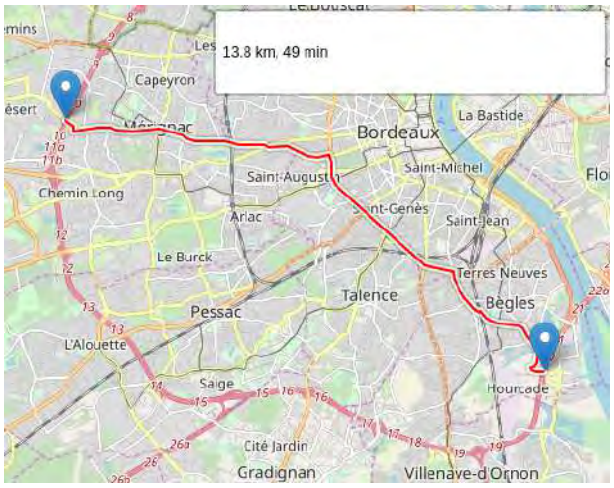

FIGURE 6.6 – Calcul d'itinéraire au plus court. Map data © OpenStreetMap

des instances allant jusqu'à 30 000 nœuds avec une distribution réaliste puisque les nœuds correspondent à la distribution de colis en Belgique. Cependant, les distances utilisées sont des distances euclidiennes. En effet, la taille d'une matrice pour un problème de 30 000 nœuds est importante. Cette taille rend alors le partage de ces données complexes, ce qui ralentit alors leur réutilisation.

Afin de faciliter des échanges de données une plateforme est proposée par Mendoza *et al.* (2014) pour centraliser la diffusion des instances. Un format XML et un outil de représentation des solutions sont également proposés.

Plus récemment, Peng et Murray (2020) ont mis en place un outil [1] permettant au travers une interface web de représenter des instances sur un fond cartographique. Cet outil propose également de récupérer des matrices à l'aide de calculateurs d'itinéraires. La constitution des instances passe soit par des données fournies par l'utilisateur, soit en ajoutant des nœuds en cliquant sur la carte.

Il n'existe pas à notre connaissance d'outil *open-source* qui permette de générer des instances de VRP ou de ses variantes à partir de bases de données cartographiques.

6.2 Proposition

Nous avons pu voir au travers des sections 2.3.2 et 5.3 qu'il est possible d'utiliser les données fournies par le projet OpenStreetMap (OSM), de les intégrer dans d'autres projets

1. Disponible sous forme d'un paquet python

et de les réutiliser. Afin d'accompagner les efforts de la communauté vers des instances à la fois plus réalistes et de plus grande en taille, nous pouvons réutiliser les concepts explorés et présentés précédemment dans cette thèse.

Ainsi, nous proposons un outil, nommé Epona [2], pour générer des instances du VRP à partir de données cartographiques en réutilisant les éléments de systèmes d'aide à la décision présentés dans le chapitre 2.

Cet outil est également à visée pédagogique puisque nous souhaitons mettre à disposition une interface graphique. L'outil doit être utilisable par des experts de la communauté de Recherche Opérationnelle, qui souhaitent construire des instances pour un scénario particulier avec des données réalistes. Cette interface doit également être adaptée à un public plus large en représentant le VRP au travers d'un outil permettant de le manipuler simplement sur un fond cartographique (Figure 6.7). L'interface graphique fait actuellement l'objet d'un stage.

Nous avons choisi de créer un premier *web service* sous forme d'une API pour générer des instances à partir des données OSM. Le choix d'une API permet d'héberger le service sur un serveur distant. Ainsi les appels aux autres services ou sources de données est pris en charge par le serveur et ne dépend pas de la stabilité de la connexion du client et n'utilise pas les ressources de la machine sur lequel est utilisé le client.

Nous souhaitons également utiliser l'API Router pour obtenir des matrices de temps et de distance issues d'itinéraires au travers du réseau routier. Puisque le projet Optimizer permet de prendre en charge une grande variété d'attributs, nous souhaitons l'appeler et obtenir quand c'est possible une solution afin de vérifier si l'instance générée permet d'obtenir des tournées réalisables. La solution obtenue peut également servir de base de comparaison. De plus, les tournées peuvent être utilisées pour leur représentation dans l'interface associée.

Par ailleurs, puisque les étapes de la génération peuvent être longues, la génération est réalisée de manière asynchrone. La longueur de la génération peut s'expliquer par l'utilisation des services suivants :

Overpass-API nous utilisons actuellement les instances mises à disposition par la communauté. Il n'existe pas de mécanisme de traitement asynchrones, mais puis-

2. Epona est une déesse de la mythologie celtique gauloise. Les symboles qui y sont associés sont le cheval et la corne d'abondance. L'une des symboliques du cheval étant le « véhicule » permettant à l'Homme d'être porté rapidement d'un point à un autre. Elle nous parait parfaitement adaptée pour représenter les problèmes de tournées de véhicules. La corne d'abondance est quant à elle une source inépuisable de bienfaits, ce qui nous parait également adapté pour représenter la génération des données.

FIGURE 6.7 – Prototype d'interface pour le générateur. Map data © OpenStreetMap

qu'il existe un mécanisme de quota, les requêtes peuvent être refusées. Ces refus obligent à répéter les requêtes après un délai d'attente qui peut aller de plusieurs secondes à plusieurs minutes.

Router-API le calcul des matrices peut être long si les matrices demandées sont larges.

Optimizer-API la résolution d'un VRP se fait majoritairement de manière asynchrone.

La collaboration avec Flavien Lucas a permis de mettre en lumière que les zones relatives à la densité de population sont intéressantes. En effet, nous les utilisons pour pondérer les profils de vitesse, ces zones délimitent des secteurs où l'intérêt et l'accessibilité des véhicules peut changer. Ces informations peuvent participer à caractériser une instance ou une solution. Ceci pourra alors être utilisé pour des méthodes d'apprentissage (Lucas, 2020)

6.3 Epona API

6.3.1 Définition des secteurs

Le périmètre de la génération doit être limité. En effet, nous souhaitons définir des instances sur des zones géographiques restreintes, ou du moins, nous voulons contrôler ce qui est généré dans chacune de ces zones.

Pour cela, nous avons deux moyens de les contrôler. Premièrement, nous pouvons définir un cadre de délimitation (*bounding box*). Ce cadre de délimitation est facilement

utilisable depuis une représentation cartographique, puisqu'il est alors possible d'utiliser les bords de cette représentation pour déterminer les coordonnées des quatre points extrêmes de ce cadre. La définition de ce cadre est simplifiée par la transmission de latitude et longitude minimales et maximales.

L'exemple 6.1 montre la requête à utiliser dans le démonstrateur Overpass-turbo pour obtenir l'ensemble des nœuds de type d'accès à l'eau potable (*drinking water*) dans la zone affichée (*bbox*). Le résultat obtenu est présenté dans la figure 6.8.

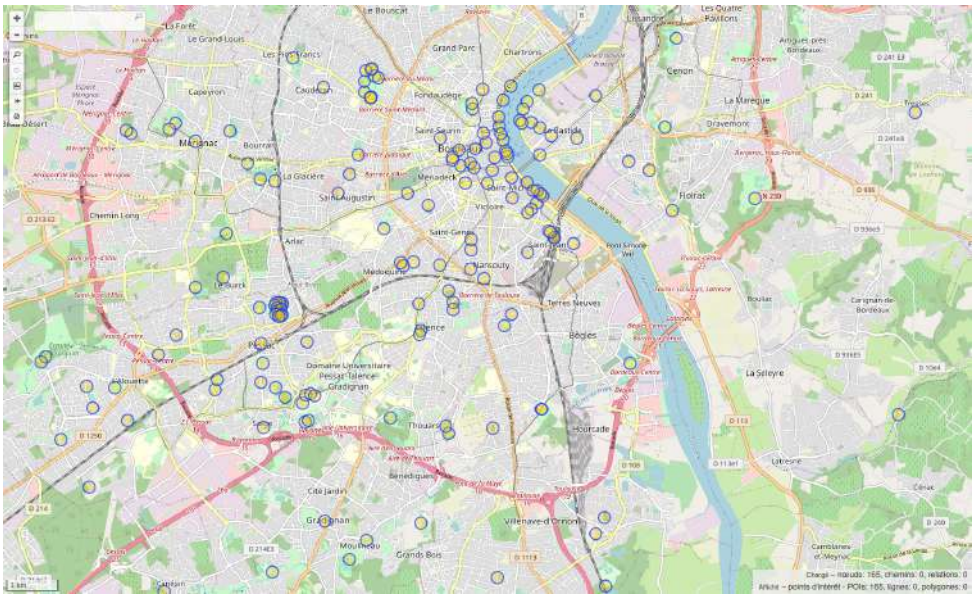

FIGURE 6.8 – Overpass : réponse à la requête de nœuds d'eau potable. Map data © OpenStreetMap

Cependant, les zones définies par un cadre de délimitation ne correspondent pas aux critères d'organisation des aires géographiques propres aux activités humaines. En effet, les aires administratives ont, particulièrement en Europe, des formes complexes.

Nous pouvons alors utiliser le concept de *relation* proposé par OSM. Certaines relations forment un polygone et sont indiquées à l'aide du libellé *area*. Un certain nombre de ces zones correspond aux délimitations de zones administratives. Il est alors possible d'utiliser ces zones, comme représenté dans l'exemple 6.2. Ici les *nodes* et *way* portant le libellé *shop*

```
1 node[amenity=drinking_water]({{bbox}});
2 out;
```

Exemple 6.1 – Overpass : Eau potable dans le cadre de délimitation

```
area[name="Bordeaux"]->.searchArea;
(
  node[shop](area.searchArea);
  way[shop](area.searchArea);
);
out;
```

Exemple 6.2 – Overpass : Boutiques dans la zone nommée Bordeaux

```
area(3600105270)->.searchArea;
(
  node[shop](area.searchArea);
  way[shop](area.searchArea);
);
out;
```

Exemple 6.3 – Overpass : Boutiques en utilisant la référence de la zone Bordeaux

sont extraites de la zone appelée Bordeaux. Il est également possible d'utiliser directement les références des zones, plutôt que d'utiliser les libellés qui la définissent. Cette possibilité est représentée dans l'exemple 6.3.

6.3.2 Définition des points

Nous voulons construire des instances avec des points qui aient un sens pour la logistique du dernier kilomètre. OSM référence un ensemble d'objets qui sont intéressants et ne sont pas restreints à un domaine particulier. Par exemple, les boutiques de tous genres y sont référencées, avec un certain nombre d'informations telles que le type de produits ou de prestations vendus, les horaires d'ouverture, etc.

Les points que nous voulons représenter peuvent être des points géographiques précis, qui sont alors représentés dans OSM par la notion de *node*. Mais ces points peuvent également être représentées par des bâtiments. Auquel cas, c'est ce bâtiment qui portera les propriétés souhaitées ; nous pouvons alors interroger les *way*. Les *way* ont dans OSM la particularité de pouvoir être représentés par un *point*. Nous ne pouvons pas savoir à priori, comment une entité est renseignée dans OSM, nous interrogeons alors simultanément *node* et *way*.

Pour demander un ensemble de points, par le biais des *node* ou des *way*, nous avons besoin d'un ensemble de libellés (*tags*) ainsi que de la zone dans laquelle ils se situent. Il n'est pas possible pour un point de fournir à la fois des coordonnées et une définition

pour le générer.

Par ailleurs, la zone dans laquelle le point doit se situer ne peut être définie que par une définition à la fois. il n'est pas possible de fournir une zone rectangulaire (*bbox*), un identifiant relation ou une description de relation (*area_tags*) simultanément.

6.3.3 Dépôts et nœuds

Les dépôts et les nœuds sont représentés par un point géographique. Leur définition passe donc soit par l'utilisation directe de coordonnées, de manière similaire à Optimizer API, soit nous devons en extraire à partir des données OSM.

La génération de points géographiques passe par la définition d'une zone, telle que présentée dans la section 6.3.1. De cette manière, nous pouvons contrôler les aires de répartition des différents points qui seront utilisés dans l'instance générée. Il est possible par exemple de choisir pour des nœuds des zones totalement disjointes.

Une fois la zone déterminée, nous devons définir les propriétés des objets que nous recherchons à l'aide des libellés (*tags*) tels que définis dans la section 6.3.2.

Il est possible de limiter le nombre de points à générer à partir d'une définition. S'il existe davantage de points dans la zone que le nombre demandé, les points sont sélectionnés aléatoirement dans l'ensemble obtenu.

Une fois les points obtenus, nous pouvons générer les demandes qui y sont associés afin de former les nœuds du problème. Similairement au concept d'unité du projet Optimizer, nous associons à chaque dimension un identifiant. Cet identifiant permet alors de lier la quantité d'une demande à la capacité associée dans les véhicules.

Il est possible pour chaque unité de donner directement la quantité associée. Cependant dans le cas où plusieurs points sont générés il est possible de définir une distribution statistique pour déterminer comment seront répartis les quantités associées aux demandes des nœuds ainsi générés. Il est à noter que les quantités fournies directement et les distributions statistiques sont mutuellement exclusives.

Par ailleurs les dépôts n'ont pas à l'heure actuelle de lien avec les demandes. En effet, il n'est pas encore possible, dans ce projet, de générer d'instances liées au Inventory Routing Problem (IRP) (Coelho *et al.*, 2014).

6.3.4 Véhicules

Un véhicule possède un certain nombre de propriétés qu'il est nécessaire d'avoir pour le lier correctement à l'instance en cours de génération.

Premièrement, nous avons besoin du type de véhicule dont il s'agit : camion, vélo, voiture, etc. Ensuite les restrictions particulières qui s'y appliquent : quelles sont ses dimensions ? Peut-il emprunter les sections à péage. Enfin, ce véhicule réalise-t-il ses itinéraires au plus rapide ou au plus court ?

Par ailleurs, ce véhicule a-t-il des dépôts ? Auquel cas, il est possible de le lier aux dépôts générés et présentés dans la section 6.3.3.

Ces informations couplées aux nœuds générés nous permettent de calculer les matrices liées à l'instance en cours de construction.

Les capacités des véhicules sont définies de manière strictement identique au projet Optimizer API, elles sont composées d'une référence à une unité et une valeur limite.

Il est également possible de définir des *skills* (Section 1.5.2) qui s'appliqueront aux véhicules définis.

Puisque nous définissons des matrices pour les distances en termes de temps et de distance, il est possible d'appliquer des coûts horaires et kilométriques ainsi qu'un coût fixe pour l'utilisation d'un véhicule. Chaque définition de véhicule s'accompagne d'un nombre de véhicule maximal de ce type à générer.

6.3.5 Configuration

La génération d'une instance est réalisée en suivant un certain nombre d'étapes :

Limite de nœuds le nombre total de nœuds à générer peut être limité au niveau global de l'instance,

Zones de densité chacun des nœuds se voit attribué un *skill* relatif à la zone de densité à laquelle il appartient, ces zones correspondent aux différentes zones de pondérations de vitesse appliqués par le Router,

Calcul des matrices les matrices sont calculées à l'aide du Router,

Résolution le projet Optimizer API est appelé afin de fournir une solution,

Filtrage si une solution est fournie par Optimizer API, alors il est possible de vérifier qu'une solution avec l'ensemble des nœuds est réalisable. Si ce n'est pas le cas, les nœuds non assignés sont filtrés et ne font plus partie de l'instance.

Ces différentes étapes sont appliquées de manière séquentielle, il est possible à l'aide de la configuration de définir quelles sont les étapes qui ne sont pas nécessaires. Il est toutefois à noter qu'une étape non réalisée empêche la réalisation des suivantes. En effet, il n'est pas possible de filtrer une instance, si elle n'a pas été évaluée au travers d'une résolution du VRP. L'étape d'application des zones de densité est toutefois optionnelle, puisqu'elle n'implique pas de contrainte forte, mais retourne principalement une indication.

Un exemple de JSON complet est présenté en Annexe avec l'exemple A.4 pour effectuer une requête sur l'API Epona.

6.4 Conclusion

Les instances de la littérature ont évolué progressivement vers davantage de réalisme. Dans ce chapitre nous proposons une contribution à cet effort de la communauté. Pour cela, un générateur d'instances utilisant des données cartographiques est proposé. Cet outil est imaginé pour proposer des instances proches des besoins de la logistique du dernier kilomètre. Les instances générées peuvent donc être accompagnées de matrices d'itinéraires calculés au travers du réseau routier. De plus, nous proposons, quand c'est possible de fournir une première solution qui permet au besoin de d'identifier et filtrer les nœuds qui ne peuvent pas être servis, ainsi que de fournir un premier point de comparaison.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Synthèse des travaux

Cette thèse, réalisée au sein de la société Mapotempo, propose des solutions pratiques pour la résolution de problèmes de tournées de véhicules dans le cadre de la logistique du dernier kilomètre. Nous avons pu introduire dans le premier chapitre les problématiques rencontrées par les clients de la société Mapotempo qui ont justifié cette thèse. Par ailleurs, nous avons détaillé les besoins particuliers des différents intervenants de la logistique du dernier kilomètre en termes de qualité de service. En effet, proposer des solutions pour la logistique du dernier kilomètre nécessite de prendre en compte les attentes du client final, du chauffeur et du planificateur. Nous avons également passé en revue un ensemble de variantes et de méthodes de résolutions disponibles dans la littérature.

Au cours du chapitre 2 nous avons détaillé le système d'aide à la décision mis en place par la société Mapotempo et qui se présente sous la forme d'un ensemble de *web services* accessibles depuis internet. Ce positionnement demande l'utilisation de concepts adaptés. Chaque service est construit autour d'une fonctionnalité. Les services se composent de Geocoder qui permet la conversion d'adresse en coordonnées (et réciproquement), Router qui permet de calculer des itinéraires ; Fleet permet l'envoi de tournées sur les smartphones des chauffeurs, ainsi que de remonter au planificateur un ensemble d'informations en cours de tournée ; Optimizer permet la résolution des variantes du VRP. Mapotempo Web est un client léger présenté sous forme d'une application web qui fait appel à l'ensemble des *web services*. Ce client léger a été construit à la fois comme un démonstrateur pour les *web services* et comme un logiciel métier clé en main.

Le chapitre 3 nous a permis de détailler le *web service* en charge de la résolution des problèmes de tournées de véhicules (VRP). Nous avons ainsi pu rentrer plus en détail dans la définition d'un *web service* sous la forme d'une API. Nous avons montré une architecture envisageable pour un tel service. De même, nous avons passé en revue un ensemble de points clés dans la conception d'un *web service* : l'échange des données, l'authentification,

la mise en place de quotas, la répartition de charge, les traitements asynchrones et la mise en cache. En réponse à cela, nous avons présenté les choix faits pour Optimizer, dont le *front end* est une API REST, de type requête/réponse. L'échange des données se fait au travers du protocole HTTPS. L'authentification se fait actuellement à l'aide de clés d'API auxquelles sont associées des quotas sous forme de fenêtres glissantes. La répartition n'est actuellement pas un sujet traité directement.

La méthode de résolution principale utilisée au sein d'Optimizer est la programmation par contraintes. Nous avons détaillé le modèle mis en place pour en bénéficier. Ceci nous a permis d'introduire OR-Tools, solveur sur lequel est appliqué le modèle. Nous avons également vu les autres solveurs (VROOM et RADOS). Cependant l'utilisation directe de solveurs ne permet pas toujours de traiter les requêtes dans un temps raisonnable. Afin de réduire la durée de traitement de ces requêtes, les méthodes *Découpe récursive* et *Dichotomious* peuvent être appelées. Optimizer intègre également une heuristique permettant de traiter les problèmes de tournées de véhicules périodiques.

Dans le chapitre 4 nous avons étudié les différentes approches de clustering : partitionnement, hiérarchique et par densité. Ces approches sont complétées par un ensemble d'indicateurs qui permettent de les évaluer. Cependant, ces indicateurs sont principalement adaptés aux approches de clustering par partitionnement. Or ces méthodes ne conviennent que partiellement aux problématiques liées aux tournées de véhicules, elles permettent de classer des données. Les clusters n'ont pas de capacité qui limitent le nombre d'éléments qui les composent.

Nous avons alors abordé les approches pour associer le clustering et le VRP. Il existe deux principales approches : les approches de type *cluster-first route-second*, où les nœuds sont groupés dans un premier temps pour ensuite appliquer la notion de tournée ; et les approches de type *route-first cluster-second*, où la première étape consiste à ordonner les points dans une tournée puis de grouper les points pour répondre au problème initial. Nous avons alors pu proposer de nouvelles approches mises en place pour les méthodes *Découpe récursive* et *Dichotomious*.

Le chapitre 5 est inspiré de la variante Clustered VRP dans laquelle les tournées sont constituées de clusters. Dans ces tournées, les véhicules doivent satisfaire l'ensemble des demandes des nœuds d'un cluster avant de passer au cluster suivant. Le besoin étudié au sein de cette thèse se rapproche davantage du Soft Clustered VRP, où il est permis de commencer à servir les nœuds d'un cluster avant de terminer les nœuds des clusters précédents. La contrainte est alors que l'ensemble des nœuds d'un cluster doivent être

servis par un même véhicule.

Nous proposons dans ce chapitre une définition de ce problème appliquée au VRP riche qui intègre une démarche de création des clusters adaptée à la logistique du dernier kilomètre. En effet, la présentation du projet OpenStreetMap dans le chapitre 2 nous permet d'envisager l'utilisation de données cartographiques pour assurer que la construction des clusters en assurant que les graphes faiblement connexes soient correctement séparés. Une fois cette première séparation appliquée, une méthode de clustering est appliquée. Nous avons choisi la méthode de clustering par densité DBSCAN afin d'assurer une propagation par densité des clusters. DBSCAN est appelé de manière récursive en réduisant progressivement le critère de distance de propagation afin d'assurer que chaque cluster puisse être servi par un seul véhicule. Ensuite nous devons assurer que tous les nœuds contenus dans un cluster puissent être servis par le même véhicule. Nous constituons alors des groupes de nœuds à la fois en les séparant par caractéristiques mais également en appliquant la résolution d'un TSP. Ce TSP permet d'extraire les nœuds qui ne peuvent pas être servis et permet d'évaluer la durée nécessaire pour servir les nœuds du groupe. La solution du problème est initialisée en résolvant un VRP composé des nœuds médians et des approximation de la durée nécessaire pour les groupes associés. Les nœuds sans groupe sont ensuite ajoutés de manière gloutonne aux tournées. Une fois la solution initiale obtenue, nous pouvons appliquer une méthode de résolution à deux niveaux en appliquant des opérateurs, sur le problème de haut niveau, adaptés à la résolution de ce problème. Puis nous pouvons résoudre les problèmes de bas niveau, où chaque tournée est un problème à résoudre.

Le chapitre 6 propose un générateur d'instances qui :

— utilise des données cartographiques pour générer les nœuds à servir,
— propose des matrices issues de calculs d'itinéraires sur le réseau routier,
— propose une solution en faisant appel à Optimizer,
— filtre les données afin d'assurer qu'il existe une solution réalisable.

Ce générateur reprend les concepts présentés dans les chapitres 2 et 3.

Perspectives

Le modèle PPC mis en place apporte une souplesse d'utilisation confortable pour la résolution de problèmes de tournées de véhicules riches, en particulier dans le cadre de la logistique du dernier kilomètre. Cependant, l'utilisation d'une métaheuristique générique

appliquée au modèle PPC ne nous permet pas de bénéficier de l'ensemble des avantages de la programmation par contraintes. Nous envisageons donc de revoir ce modèle et son implémentation afin de basculer sur une résolution exacte sur les problèmes de petite taille.

Nous avons montré que les méthodes de recherche exactes ne permettaient pas de répondre aux impératifs temporels nécessaires pour répondre aux besoins de la logistique du dernier kilomètre. Cependant, la création de sous-problèmes à l'aide des méthodes *Découpe récursive*, *Dichotomious* ou la méthode de séparation et de résolution appliquée au *Rich Soft Layered-Clustered Problem* nous permet de constituer des sous problèmes de taille réduite qui pourraient bénéficier d'une recherche exacte.

La méthode de clustering *VRP-Kmeans* constitue des clusters qui se chevauchent. De plus, dans les zones peu denses, il est possible que deux nœuds proches se retrouvent dans des clusters différents, alors qu'il serait plus pertinent de les servir avec le même véhicule. L'application de la découpe topographique, puis de DBSCAN avec capacité en amont de l'application de *VRP-Kmeans* permettrait de réduire ces phénomènes.

Pour la méthode de résolution mise en place pour le *Rich Soft Layered-Clustered Problem*, les approximations de temps de parcours des groupes sont à affiner. En effet, celles-ci supposent qu'un groupe ne commence à être servi que lorsque le nœud médian est atteint. Par ailleurs, les opérateurs pour les échanges de groupes entre tournées sont impactés par cette approximation. Ils devront donc être revus afin de bénéficier des améliorations apportées et accroître le taux auquel les opérateurs permettent de diminuer le coût de la solution.

Concernant le générateur Epona, les variantes actuellement couvertes n'intègrent pas la génération de fenêtres de temps par exemple. Or, ces informations sont disponibles et elles ont, d'ailleurs, fait l'objet d'un projet communautaire OpenStreetMap durant le confinement de mars 2020 [3]. Nous voulons étendre à la fois le nombre de variantes possibles au sein d'Epona, mais également le nombre de sources de données qui peuvent être utilisées pour constituer les instances.

3. https ://wiki.openstreetmap.org/wiki/France/Covid-19

BIBLIOGRAPHIE

Nuri Almarimi, Ali Ouni, Salah Bouktif, Mohamed Wiem Mkaouer, Raula Gaikovina Kula et Mohamed Aymen Saied : Web service API recommendation for automated mashup creation using multi-objective evolutionary search. *Applied Soft Computing Journal*, 82:1–13, 2019.

Zyad M. Alzaydi, Ali Al-Hajla, Bang Nguyen et Chanaka Jayawardhena : A review of service quality and service delivery : Towards a customer co-production and customer-integration approach. *Business Process Management Journal*, 24(1):295–328, 2018. ISSN 14637154.

David Applegate, William Cook et André Rohe : Chained Lin-Kernighan for Large Traveling Salesman Problems. *INFORMS Journal on Computing*, 15(1):82–92, 2003. URL `http://www.math.princeton.edu/tsp/`.

Claudia Archetti, Elena Fernández et Diana L. Huerta-Muñoz : The Flexible Periodic Vehicle Routing Problem. *Computers and Operations Research*, 85:58–70, 2017. ISSN 03050548.

Florian Arnold, Michel Gendreau et Kenneth Sörensen : Efficiently Solving Very Large Scale Routing Problems. *Computers and Operations Research*, 107:32–42, 2019.

Florian Arnold et Kenneth Sörensen : What makes a solution good ? The generation of problem-specific knowledge for heuristics. *Computers and Operations Research*, 106: 280–288, 2019.

Farhad Azadivar, Tu Truong et Yue Jiao : A decision support system for fisheries management using operations research and systems science approach. *Expert Systems with Applications*, 36(2):2971–2978, 2009. ISSN 09574174.

Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin W P Savelsbergh et Pamela H Vance : Branch-and-price : Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

Thibaut Barthélemy, André Rossi, Marc Sevaux et Kenneth Sörensen : Metaheuristic approach for the clustered VRP. *In EU/MEeting : 10th Anniversary of the Metaheuristics Community-Université de Bretagne Sud, France*, 2010.

Maria Battarra, Güneş Erdoğan et Daniele Vigo : Exact algorithms for the clustered vehicle routing problem. *Operations Research*, 62(1):58–71, 2014. ISSN 0030364X.

Nicolas Beldiceanu, Mats Carlsson et Jean-Xavier Rampon : Global Constraint Catalog, 2010. ISSN 1100-3154.

Farah Belmecheri, Christian Prins, Farouk Yalaoui et Lionel Amodeo : Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. *Journal of Intelligent Manufacturing*, 24(4):775–789, 2013. ISSN 09565515.

Naima Benkebir, Marc Le Pouliquen, Jean-François Trévien, Ahcène Bounceur, Reinhardt Euler, Emmanuelle Pardiac et Marc Sevaux : On a multi-trip vehicle routing problem with time windows integrating European and French driver regulations. *Journal on Vehicle Routing Algorithms*, 2:55–74, 2019.

Tony Bourke : *Server load balancing*. O'Reilly, 2001. ISBN 0596000502.

Eric Bourreau, Matthieu Gondran, Philippe Lacomme et Marina Vinot : *Programmation Par Contraintes : démarches de modélisation pour des problèmes d'optimisation*. Ellipses, 2020. ISBN 2340035856.

Burak Boyaci, Thu Huong Dang et Adam N. Letchford : Vehicle routing on road networks : How good is Euclidean approximation ? *Computers and Operations Research*, 129:105197–, 2021. ISSN 03050548.

José Brandão : A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers and Operations Research*, 38(1):140–151, 2011. ISSN 03050548.

David Bredström et Mikael Rönnqvist : Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191:19–31, 2008. ISSN 03772217.

Ulrich Breunig, Verena Schmid, Richard F. Hartl et Thibaut Vidal : A large neighbourhood based heuristic for two-echelon routing problems. *Computers and Operations Research*, 76:208–225, 12 2016. ISSN 03050548.

Andreas Brieden, Peter Gritzmann et Fabian Klemm : Constrained clustering via diagrams : A unified theory and its application to electoral district design. *European Journal of Operational Research*, 263(1):18–34, 2017. ISSN 03772217.

Kamil Bujel, Feiko Lai, Michal Szczecinski, Winnie So et Miguel Fernandez : Solving High Volume Capacitated Vehicle Routing Problem with Time Windows using Recursive-DBSCAN clustering algorithm. *arXiv preprint arXiv :1812.02300*, 2018. URL `http://arxiv.org/abs/1812.02300`.

George Büttner, Jan Feranec, Gabriel Jaffrain, László Mari, Gergely Maucha et Tomas Soukup : The CORINE land cover 2000 project. *EARSeL eProceedings*, 3 (3):331–346, 2004.

Jose Caceres-Cruz, Pol Arias, Daniel Guimarans, Daniel Riera et Angel A. Juan : Rich vehicle routing problem : Survey. *ACM Computing Surveys (CSUR)*, 47(2):1–28, 2014.

Paola Cappanera, Luís Gouveia et Maria Grazia Scutellà : The skill vehicle routing problem. *In International Conference on Network Optimization*, pages 354–364, 2011.

Ivan Cardenas, Yari Borbon-Galvez, Thomas Verlinden, Eddy Van de Voorde, Thierry Vanelslander et Wouter Dewulf : City logistics, urban goods distribution and last mile delivery and collection. *Competition and Regulation in Network Industries*, 18(1-2):22–43, 2017.

Maxime Chassaing, Marc Sevaux, Pierre Bomel et Ivan Crepeau : Un Web Service sur les problèmes de transport avec des distances réelles, pour propager les algorithmes de recherche opérationnelle. *In ROADEF : Recherche Opérationelle et d'Aide à la Décision*, 2017. URL `http://labsticc.univ-ubs.fr/WS4RP/`.

Nicos Christofides : Worst-case analysis of a new heuristic for the travelling salesman problem. Rapport technique, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.

Geoff Clarke et John W. Wright : Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 54(2):568–581, 1964.

Jens Clausen : Branch and Bound Algorithms-Principles and Examples. *Department of Computer Science, University of Copenhagen*, pages 1–30, 1999.

Leandro C. Coelho, Jean François Cordeau et Gilbert Laporte : Thirty years of inventory routing. *Transportation Science*, 48(1):1–19, 2014. ISSN 15265447.

Marc David Cohen, Charles B. Kelly et Andrés L. Medaglia : Decision support with Web-enabled software. *Interfaces*, 31(2):109–129, 2001. ISSN 00922102.

Jean-François Cordeau, Michel Gendreau et Gilbert Laporte : A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems. *Networks*, 30(2):105–119, 1997.

Jean-François Cordeau, Gilbert Laporte et Anne Mercier : A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. *Journal of the Operational Research Society*, 52(8):928–936, 2001. URL `https://www.jstor.org/stable/822953?seq=1&cid=pdf-`.

Julien Coupey : VROOM, 2015. URL `https://github.com/VROOM-Project/vroom`.

Teodor Gabriel Crainic, Michel Gendreau et Jean Yves Potvin : Intelligent freight-transportation systems : Assessment and the contribution of operations research. *Transportation Research Part C : Emerging Technologies*, 17(6):541–557, 2009. ISSN 0968090X.

George B. Dantzig et John H. Ramser : The truck dispatching problem. *Management Science*, 6:80–91, 1959.

George B Dantzig et Philip Wolfe : Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

Christof Defryn et Kenneth Sörensen : A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Computers and Operations Research*, 83:78–94, 2017. ISSN 03050548.

Guy Desaulniers, Jacques Desrosiers, Andreas Erdmann, Marius M Solomon et Francois Soumis : VRP with Pickup and Delivery. *In* Paolo Toth et Daniele Vigo, éditeurs : *The Vehicle Routing Problem*, pages 225–242. SIAM, 2002.

Guy Desaulniers, Oli B.G. Madsen et Stefan Ropke : The vehicle routing problem with time windows. *In Vehicle Routing : Problems, Methods, and Applications, Second Edition*, pages 119–159. SIAM, 2014.

Rodolfo Dondo et Jaime Cerdá : A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, 176(3):1478–1507, 2007. ISSN 03772217.

Michael Drexl : Rich vehicle routing in theory and practice. *Logistics Research*, 5 (1-2):47–63, 2012. ISSN 1865035X.

Richard O. Duda, Peter E. Hart et David G. Stork : *Pattern Classification and Scene Analysis*, volume 3. Wiley New York, 1973.

Christophe Duhamel, Philippe Lacomme et Caroline Prodhon : Efficient frameworks for greedy split and new depth first search split procedures for routing problems. *Computers and Operations Research*, 38(4):723–739, 2011. ISSN 03050548.

Christophe Duhamel, Philippe Lacomme et Caroline Prodhon : A hybrid evolutionary local search with depth first search split procedure for the heterogeneous vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 25(2):345–358, 2012. ISSN 09521976.

A. Expósito, J. Brito, José A. Moreno et Christopher Expósito-Izquierdo : Quality of service objectives for vehicle routing problem with time windows. *Applied Soft Computing Journal*, 84:105707, 2019. ISSN 15684946.

Christopher Expósito-Izquierdo, André Rossi et Marc Sevaux : A Two-Level solution approach to solve the Clustered Capacitated Vehicle Routing Problem. *Computers and Industrial Engineering*, 91:274–289, 1 2016. ISSN 03608352.

Thomas A Feo et Mauricio G C Resende : A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71, 1989.

Thomas A Feo et Mauricio G C Resende : Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

Marshall L Fisher : Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42(4):626–642, 1994.

Marshall L. Fisher et Ramchandran Jaikumar : A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.

Hermann Gehring et Jörg Homberger : A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. *In Proceedings of EUROGEN99*, pages 57–64, 1999.

Michel Gendreau et Christos D. Tarantilis : *Solving Large-Scale Vehicle Routing Problems with Time Windows : The State-of-the-Art*. Cirrelt Montreal, 2010.

Ian P. Gent, Ian Miguel et Peter Nightingale : Generalised arc consistency for the AllDifferent constraint : An empirical survey. *Artificial Intelligence*, 172(18):1973–2000, 2008. ISSN 00043702.

Billy E. Gillett et Leland R. Miller : A Heuristic Algorithm for the Vehicle-Dispatch Problem. *Operations research*, 22(2):340–349, 1974.

Fred Glover : Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

Asvin Goel : Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, 43(1):17–26, 2009. ISSN 15265447.

David E. Goldberg : *Genetic algorithms*. Pearson Education India, 2006.

Bruce L. Golden, Edward A. Wasil, James P. Kelly et I-Ming Chao : The impact of metaheuristics on solving the vehicle routing problem : algorithms, problem sets, and computational results. *In Fleet management and logistics*, pages 33–56. Springer, 1998.

Ralph E. Gomory : Outline of an algorithm for integer solutions to linear programs. *Bulletin of American Mathematical Society*, 64:275–278, 1958.

Élisabeth Habert : De l'État au citoyen, redistribution des cartes : éléments d'une histoire de la cartographie. *Revue d'ethnoécologie*, 11, 2017.

Ruhan He, Weibin Xu, Jiaxia Sun et Bingqiao Zu : Balanced K-means algorithm for partitioning areas in large-scale vehicle routing problem. *In 3rd International Symposium on Intelligent Information Technology Application, IITA 2009*, pages 87–90, 2009. ISBN 9780769538594.

Vera C. Hemmelmayr, Jean-François Cordeau et Teodor Gabriel Crainic : An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. *Computers and Operations Research*, 39(12):3215–3228, 2012.

Florent Hernandez, Michel Gendreau et Jean Yves Potvin : Heuristics for tactical time slot management : a periodic vehicle routing problem view. *International Transactions in Operational Research1233*, 24(6):1233–1252, 2017. ISSN 14753995.

Timo Hintsch : Large multiple neighborhood search for the soft-clustered vehicle-routing problem. *Computers and Operations Research*, 129:105132, 2021. ISSN 03050548.

Irène Hirt : Cartographies autochtones. Éléments pour une analyse critique. *L'Espace géographique*, 38(2):171–186, 2009.

John H Holland : Adaptation in natural and artificial systems. *Ann Arbor : University of Michigan press*, 1(97):5, 1975.

Daniel Jacobson, Greg Brail et Dan Woods : *APIs : A Strategy Guide*. O'Reilly, 2012. ISBN 978-1449308926.

Brenda Jin, Saurabh Sahni et Amir Shevat : *Designing Web APIs - Building APIs that developers love*. O'Reilly Media Inc, USA, 2018. ISBN 978-1449308926.

Erman Karakoc et Pinar Senkul : Composing semantic Web services under constraints. *Expert Systems with Applications*, 36(8):11021–11029, 2009. ISSN 09574174.

Leonard Kaufman et Peter J. Rousseeuw : *Finding groups in data : an introduction to cluster analysis*. John Wiley & Sons, 1990. ISBN 9780470316801.

Peter G W Keen : Decision support systems : the next decade. *Decision Support Systems*, 3(3):253–265, 1987.

Peter Bernard Keenan et Piotr Jankowski : Spatial Decision Support Systems : Three decades on. *Decision Support Systems*, 116:64–76, 2019. ISSN 01679236.

Michael Khachay et Helen Zaytseva : Polynomial Time Approximation Scheme for Single-Depot Euclidean Capacitated Vehicle Routing Problem. *In Combinatorial Optimization and Applications*, Lecture Notes in Computer Science, pages 178–190. Springer, 2015. ISBN 978-3-319-26625-1.

Hans Peter Kriegel, Peer Kröger, Jörg Sander et Arthur Zimek : Density-based clustering. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 1 (3):231–240, 2011. ISSN 19424795.

Philippe Lacomme, Chrisian Prins et Wahiba Ramdane-Chérif : A genetic algorithm for the capacitated arc routing problem and its extensions. *In Workshops on Applications of Evolutionary Computation*, pages 473–483, 2001.

Ailsa H. Land et Alison G. Doig : An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28(3):497–520, 1960.

Csaba Legány, Sándor Juhász et Attila Babos : Cluster Validity Measurement Techniques. *In Proceedings of the 5th WSEAS international conference on artificial intelligence, knowledge engineering and data bases*, pages 388–393, 2006.

Feiyue Li, Bruce Golden et Edward Wasil : A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, 34(9):2734–2742, 2007. ISSN 03050548.

Canhong Lin, King L. Choy, George T.S. Ho, H. Y. Lam, Grantham K.H. Pang et Kwai-Sang Chin : A decision support system for optimizing dynamic courier routing operations. *Expert Systems with Applications*, 41(15):6917–6933, 2014. ISSN 09574174.

John D. C. Little, Katta G. Murty, Dura W. Sweeny et Caroline Karel : An algorithm for the traveling salesman problem. *Operations Research*, 11(6):972–989, 1963.

Fuh-Hwa Liu et Sheng-Yuan Shen : The fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society*, 50(7):721–732, 1999. ISSN 14769360.

Flavien Lucas : *Résolution de problèmes réalistes de tournées à flotte hétérogène en milieu urbain : vers un solveur adaptatif mêlant recherche opérationnelle et apprentissage automatique*. Thèse de doctorat, Université Bretagne Sud, IMT Atlantique, 2020.

Cathy Macharis et Sandra Melo : *City distribution and urban freight transport : multiple perspectives*. Edward Elgar (Ed), 2011. ISBN 9780857932747.

Eleanor A. Maguire, Katherine Woollett et Hugo J. Spiers : London taxi drivers and bus drivers : A structural MRI and neuropsychological analysis. *Hippocampus*, 16 (12):1091–1101, 2006. ISSN 10509631.

Mark Massé : *REST API Design Rulebook*. O'Reilly, 2011. ISBN 978-1449310509.

Ujjwal Maulik et Sanghamitra Bandyopadhyay : Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 2002. ISSN 01628828.

Peter Membrey, David Hows et Eelco Plugge : *Practical load balancing : ride the performance tiger*. Apress, 2012. ISBN 9781430236801.

Jorge E Mendoza, M Hoskins, Christelle Guéret, Victor Pillac et D Vigo : VRP-REP : a vehicle routing community repository. *In VeRoLog14*, Oslo, 2014. URL `http://okina.univ-angers.fr/publications/ua3268`.

Jorge E. Mendoza, Andrés L. Medaglia et Nubia Velasco : An evolutionary-based decision support system for vehicle routing : The case of a public utility. *Decision Support Systems*, 46(3):730–742, 2009. ISSN 01679236.

John E Mitchell : Branch-and-Cut Algorithms for Combinatorial Optimization Problems. *Handbook of Applied Optimization*, 1:65–77, 2002.

Arsh Modak, Sanjay D. Chaudhary, Priyanka S. Paygude et S .R Ldate : Techniques to Secure Data on Cloud : Docker Swarm or Kubernetes. *In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 7–12, 2018. ISBN 9781538619742.

Fionn Murtagh et Pedro Contreras : Algorithms for hierarchical clustering : An overview. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 2 (1):86–97, 2012. ISSN 19424795.

Charles Edward Noon : *The Generalized Traveling Salesman Problem*. Thèse de doctorat, Ann Arbor : University of Michigan, 1988.

Regina Obe et Leo Hsu : PostGIS in action. *GEOInformatics*, 14(8):30–33, 2011.

Roland M. Olbricht : Data retrieval for small spatial regions in OpenStreetMap. *In OpenStreetMap in GIScience*, pages 101–122. Springer, 2015.

Ibrahim Hassan Osman : Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41(4):421–451, 1993.

Manfred Padberg et Giovanni Rinaldi : A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Review*, 33 (1):60–100, 1991. URL http://www.siam.org/journals/ojsa.php.

Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar et Frank Leymann : Service-Oriented Computing : State of the Art and Research Challenges. *Computer*, 40 (11):38–45, 2007.

Julie Paquette, François Bellavance, Jean François Cordeau et Gilbert Laporte : Measuring quality of service in dial-a-ride operations : The case of a Canadian city. *Transportation*, 39(3):539–564, 2012. ISSN 00494488.

Julie Paquette, Jean François Cordeau et Gilbert Laporte : Quality of service in dial-a-ride operations. *Computers and Industrial Engineering*, 56(4):1721–1734, 2009. ISSN 03608352.

José Antonio Parejo, Sergio Segura, Pablo Fernandez et Antonio Ruiz-Cortés : QoS-aware web services composition using GRASP with Path Relinking. *Expert Systems with Applications*, 41(9):4211–4223, 2014. ISSN 09574174.

Lan Peng et Chase C Murray : VeRoViz : A Vehicle Routing Visualization Toolkit. *SSRN*, pages 1–18, 2020. URL https://ssrn.com/abstract=3746037.

Laurent Perron et Vincent Furnon : OR-Tools, 2013. URL https://developers.google.com/optimization/.

Jean-Yves Potvin et Michel Gendreau : *Handbook of Metaheuristics*, volume 2. Springer International Publishing, 2019.

Marie Pouponneau et François Cape : Les zones à faibles émissions (low emission zones) à travers l'Europe : déploiement, retours d'expérience, évaluation d'impact et efficacité du système. *Pollution Atmosphérique*, 235:6325, 2017.

Ali Gul Qureshi, Eiichi Taniguchi et Tadashi Yamada : Exact solution for the Vehicle Routing Problem with Semi Soft Time Windows and its application. *In Procedia - Social and Behavioral Sciences*, 2010. ISBN 18770428.

Irendra Radjawali : Counter-Mapping Land Grabs with Community Drones in Indonesia. *In Land grabbing, conflict and agrarian-environmental transformations : perspectives from East and Southeast Asia*, 2015. ISBN 3120675717.

Alireza Rahimi-Vahed, Teodor Gabriel Crainic, Michel Gendreau et Walter Rei : Fleet-sizing for multi-depot and periodic vehicle routing problems using a modular heuristic algorithm. *Computers and Operation Research*, 53:9–23, 2014.

Aurora Ramírez, José Antonio Parejo, José Raúl Romero, Sergio Segura et Antonio Ruiz-Cortés : Evolutionary composition of QoS-aware web services : A many-objective perspective. *Expert Systems with Applications*, 72:357–370, 2017. ISSN 09574174.

Jacques Renaud, Fayez F. Boctor et Gilbert Laporte : An Improved Petal Heuristic for the Vehicle Routing Problem. *Journal of Operational Research Society*, 47(2):329–336, 1996.

Eréndira Rendón, Itzel Abundez, Alejandra Arizmendi et Elvia M Quiroz : Internal versus External cluster validation indexes. *International Journal of computers and communications*, 5(1):27–34, 2011.

Yves Rochat et Éric D. Taillard : Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167, 1995.

Juan Carlos Rojas-Thomas, Mathilde Santos et Marco Mora : New internal index for clustering validation based on graphs. *Expert Systems with Applications*, 86:334–349, 2017. ISSN 09574174.

Stefan Ropke et David Pisinger : An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40 (4):455–472, 2006. ISSN 15265447.

Francesca Rossi, Peter Van Beek et Toby Walsh : *Handbook of Constraint Programming*. Elsevier, 2006. ISBN 978-0-444-52726-4.

Ruslan Sadykov, Eduardo Uchoa et Artur Pessoa : A Bucket Graph–Based Labeling Algorithm with Application to Vehicle Routing. *Transportation Science*, 2020. ISSN 0041-1655.

David Savourey : Programmation par contraintes : Cours 2 - Arc consistance et autres amusettes.

Amit Saxena, Mukesh Prasad, Akshansh Gupta, Neha Bharill, Om Prakash Patel, Aruna Tiwari, Meng Joo Er, Weiping Ding et Chin Teng Lin : A review of clustering techniques and developments. *Neurocomputing*, 267:664–681, 2017. ISSN 18728286.

Stefan Schröder : Jsprit, 2014. URL `https://github.com/graphhopper/jsprit`.

Marc Sevaux et Kenneth Sörensen : Hamiltonian paths in VRP Hamiltonian paths in large clustered routing problems. *In EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing*, pages 411–417, 2008.

Jung P. Shim, Merrill Warkentin, James F. Courtney, Daniel J. Power, Ramesh Sharda et Christer Carlsson : Past, present, and future of decision support technology. *Decision Support Systems*, 33(2):111–126, 2002.

Va Sivaram Kumar, M.R. Thansekhar, R Saravanan et S Miruna Joe Amali : Solving Multi-objective Vehicle Routing Problem with Time Windows by FAGA. *Procedia Engineering*, 97:2176–2185, 2014.

Marius M Solomon : Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2):254–265, 1987.

María Soto, Marc Sevaux, André Rossi et Andreas Reinholz : Multiple neighborhood search, tabu search and ejection chains for the multi-depot open vehicle routing problem. *Computers and Industrial Engineering*, 107:211–222, 2017. ISSN 03608352.

Eric Taillard : Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673, 1993.

Duygu Taş, Nico Dellaert, Tom Van Woensel et Ton de Kok : The time-dependent vehicle routing problem with soft time windows and stochastic travel times. *Transportation Research Part C : Emerging Technologies*, 2014. ISSN 0968090X.

Paolo Toth et Daniele Vigo : The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *Informs Journal on Computing*, 15(4):333–346, 2003.

Ruey S. Tsay : *Analysis of Financial Time Series*. John Wiley & Sons, Inc, 2005. ISBN 978-0471690740.

Shu Mei Tseng et Pin Hong Wu : The impact of customer knowledge and customer relationship management on service quality. *International Journal of Quality and Service Sciences*, 6(1):77–96, 2014. ISSN 17566703.

Gündüz Ulusoy : The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22(3):329–337, 1985.

David Van Dantzig : General procedures of empirical science. *Synthese*, 5(9):441–455, 1947.

Robert J. Vanderbei : *Linear Programming*, volume 5. Springer, 2020. ISBN 978-3-030-39415-8.

Thibaut Vidal, Maria Battarra, Anand Subramanian et Güneş Erdogan : Hybrid metaheuristics for the Clustered Vehicle Routing Problem. *Computers and Operations Research*, 58:87–99, 2015. ISSN 03050548.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi et Walter Rei : A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*, 60(3):611–624, 2012. ISSN 0030-364X.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau et Christian Prins : A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers and Operations Research*, 40(1):475–489, 2013a. ISSN 03050548.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau et Christian Prins : Heuristics for multi-attribute vehicle routing problems : A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21, 2013b.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau et Christian Prins : A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234:658–673, 2014.

Christos Voudouris et Edward P. K. Tsang : Guided Local Search. *In Handbook of metaheuristics*, pages 185–218. Springer, 2003. ISBN 978-0-306-48056-0.

Wei Wang, Marta Indulska et Shazia Sadiq : Guidelines for Business Rule Modeling Decisions. *Journal of Computer Information Systems*, 58(4):363–373, 2018.

Anthony Wren et Alan Holliday : Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points. *Journal of the Operational Research Society*, 23(3):333–344, 1972.

Jiang Xie, Zhong-Yang Xiong, Qi-Zhu Dai, Xiao-Xia Wang et Yu-Fang Zhang : A new internal index based on density core for clustering validation. *Information Sciences*, 506:346–365, 2020. ISSN 00200255.

Peisheng Zhao, Liping Di et Genong Yu : Building asynchronous geospatial processing workflows with web services. *Computers and Geosciences*, 39:34–41, 2012.

Wei Zhou, Tingxin Song, Fei He et Xi Liu : Multiobjective vehicle routing problem with route balance based on genetic algorithm. *Discrete Dynamics in Nature and Society*, 2013:325686, 2013. ISSN 10260226.

ANNEXES

A.1 Mapotempo Web : captures d'écran

Les figures de cette section reprennent les concepts présentés dans la section 2.6. Les dépôts (Figure A1) sont les points de départ et de fin des tournées.

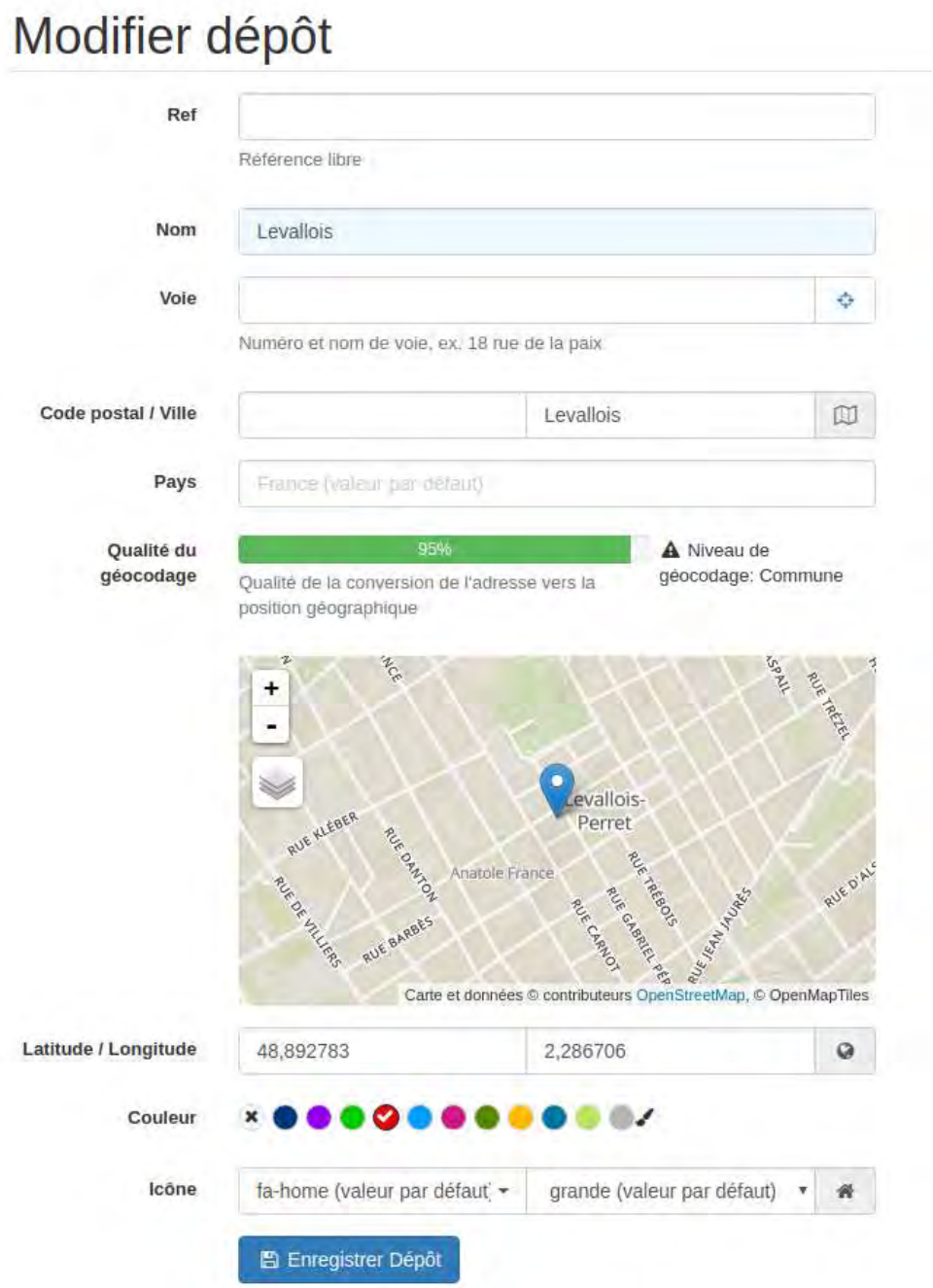

FIGURE A1 – Dépôt

Les unités livrables (Figure A2) permettent de représenter les entités échangées en cours de tournées.

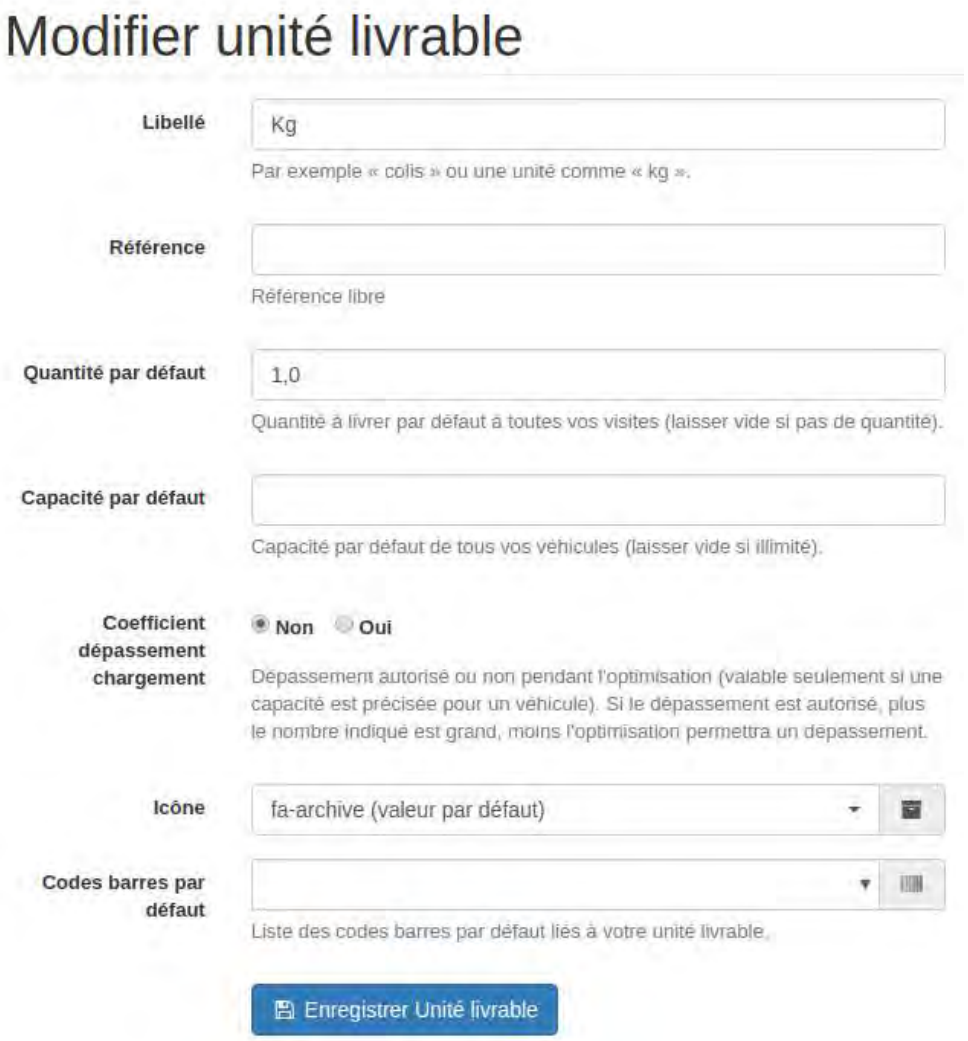

FIGURE A2 – Unités livrables

Les véhicules représentent à la fois le moyen de transport (Figure A3) ainsi que les caractéristiques liées au chauffeur (Figure A4).

FIGURE A3 – Véhicule physique

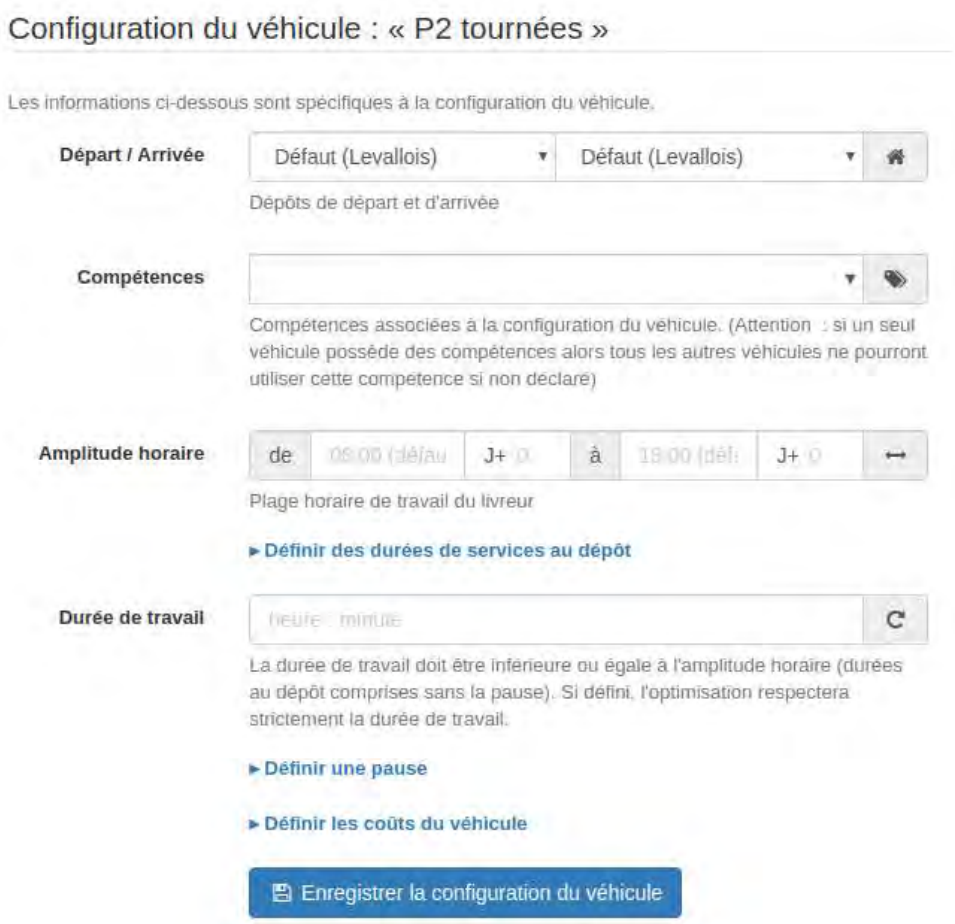

FIGURE A4 – Véhicule et agent

Les libellés (Figure A5) permettent de donner d'appliquer une icône ou une couleur particulière aux nœuds du VRP, appelés visites dans le contexte de Mapotempo Web. Ces libellés permettent également de signifier les *skills* des nœuds et des véhicules.

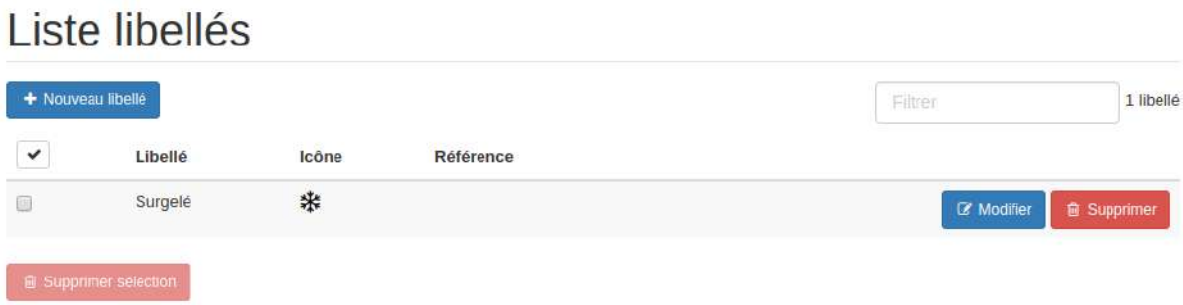

FIGURE A5 – Libellés

Un client (Figure A6) permet de localiser un point sur la carte. Ce concept est utilisé pour localiser les visites qui sont les nœuds du VRP. Ainsi, plusieurs visites peuvent correspondre au même client.

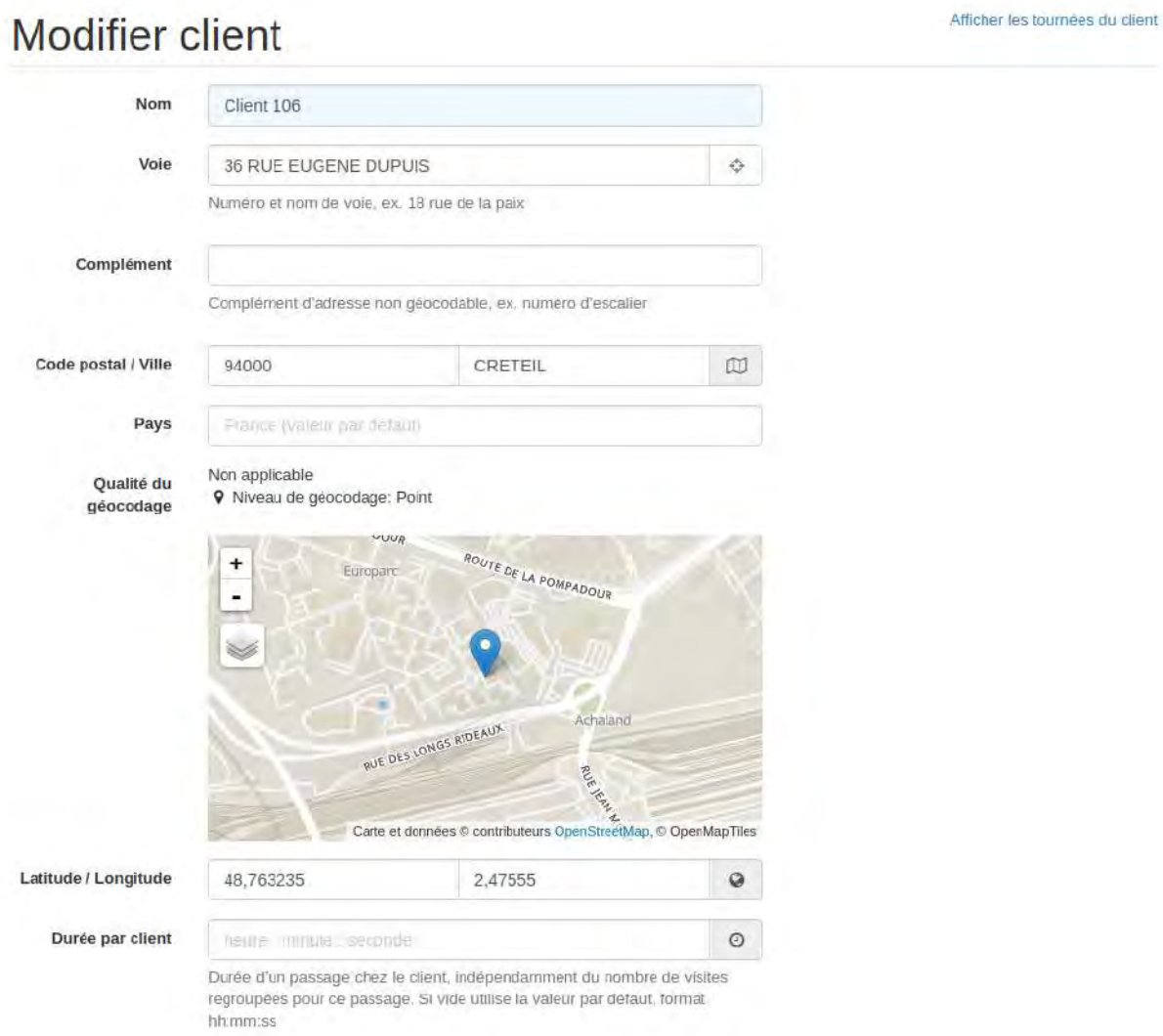

FIGURE A6 – Destination

Les visites (A7) permettent de décrire les demandes à satisfaire pour servir un nœud. Il s'accompagne également de fenêtre de temps ou des libellés (*skills*).

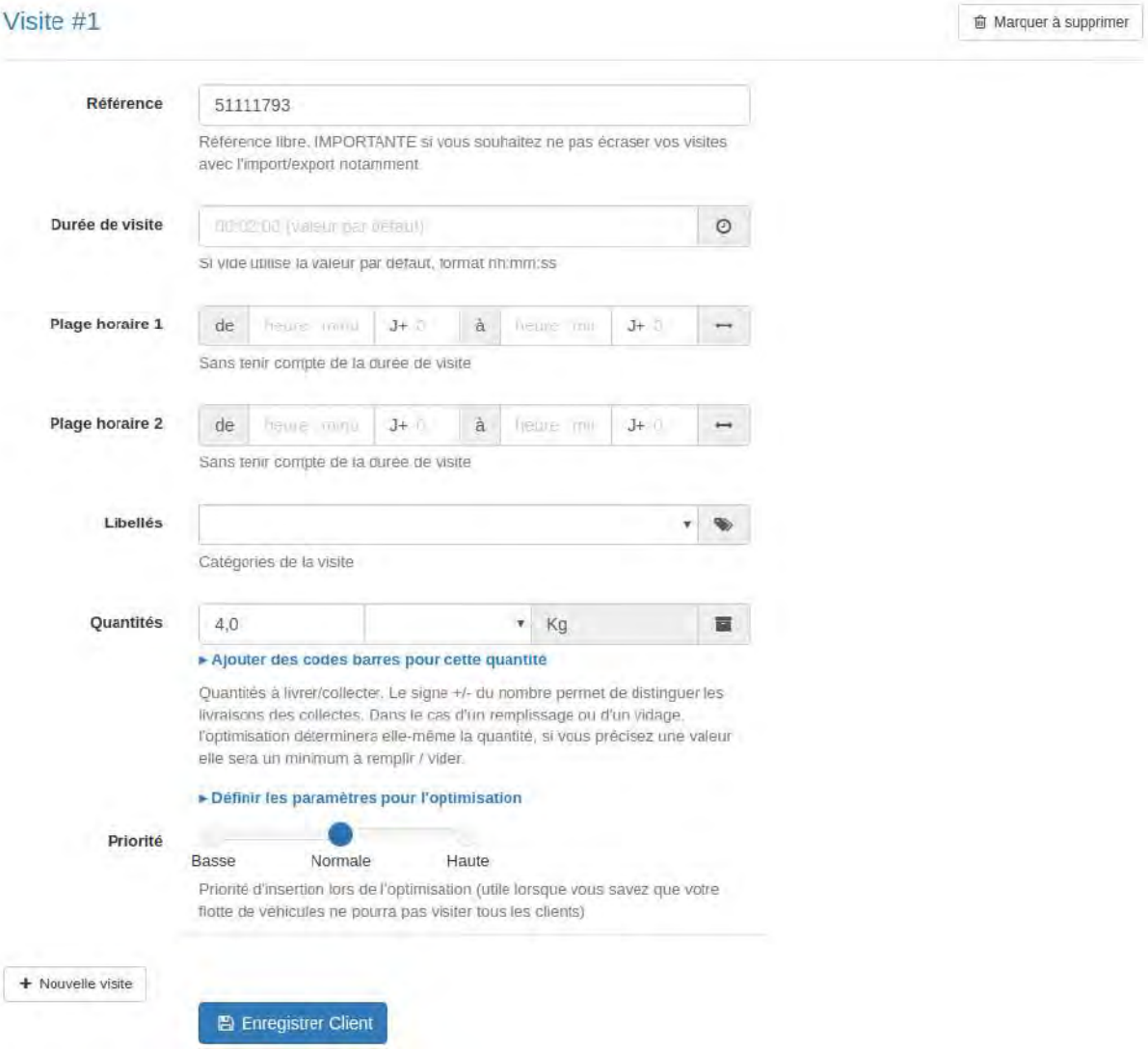

FIGURE A7 – Visite

L'interface de zonage A8) permet de grouper les clients dans des zones construites manuellement ou à l'aide d'algorithme de clustering.

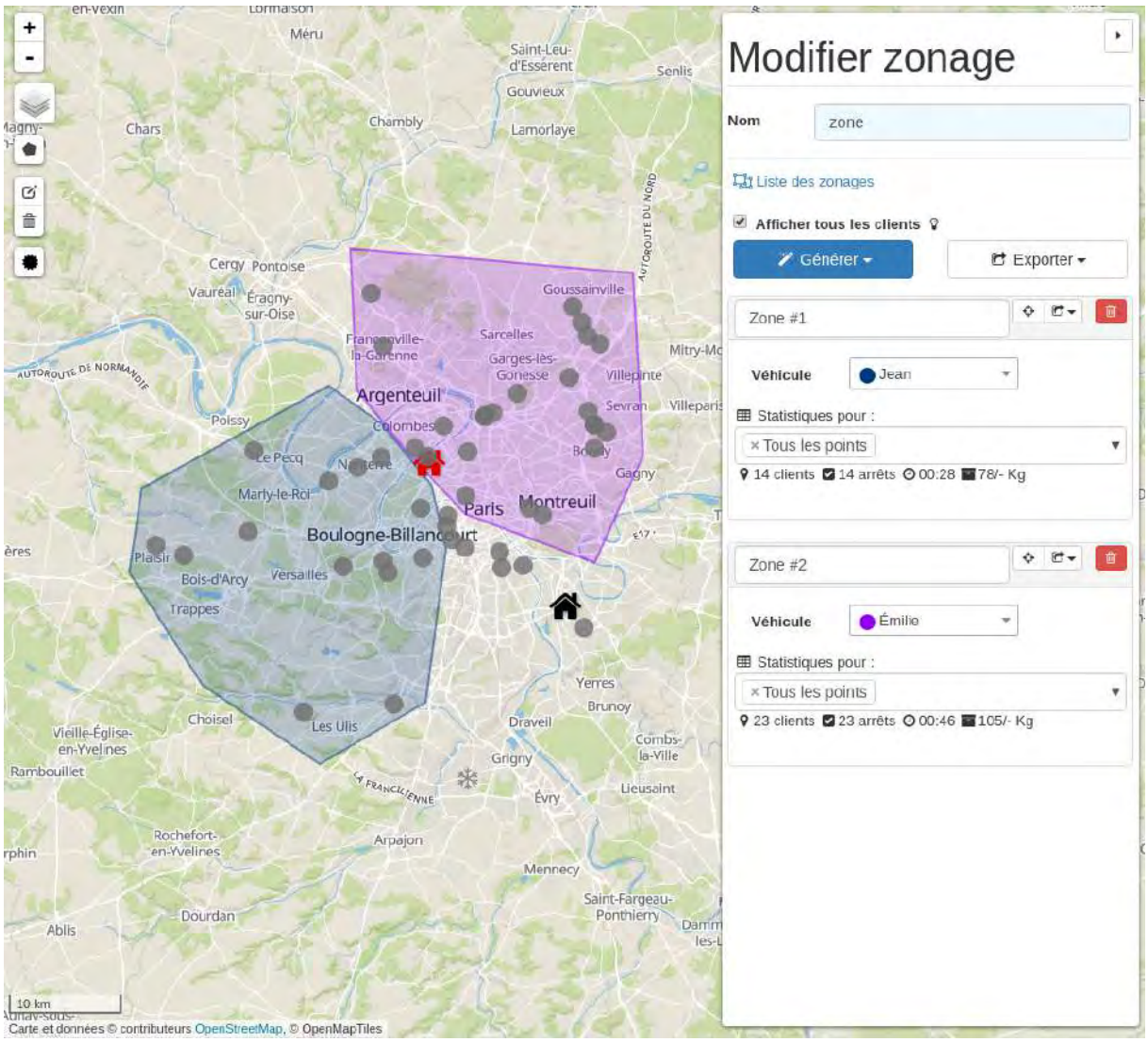

FIGURE A8 – Zonage

Le plan (A9) permet de représenter l'ensemble des concepts utiles à la manipulation ou la résolution des données permettant de résoudre un VRP. Un ensemble de sont également disponibles afin de fournir au planificateur les informations nécessaire à ses prises de décisions.

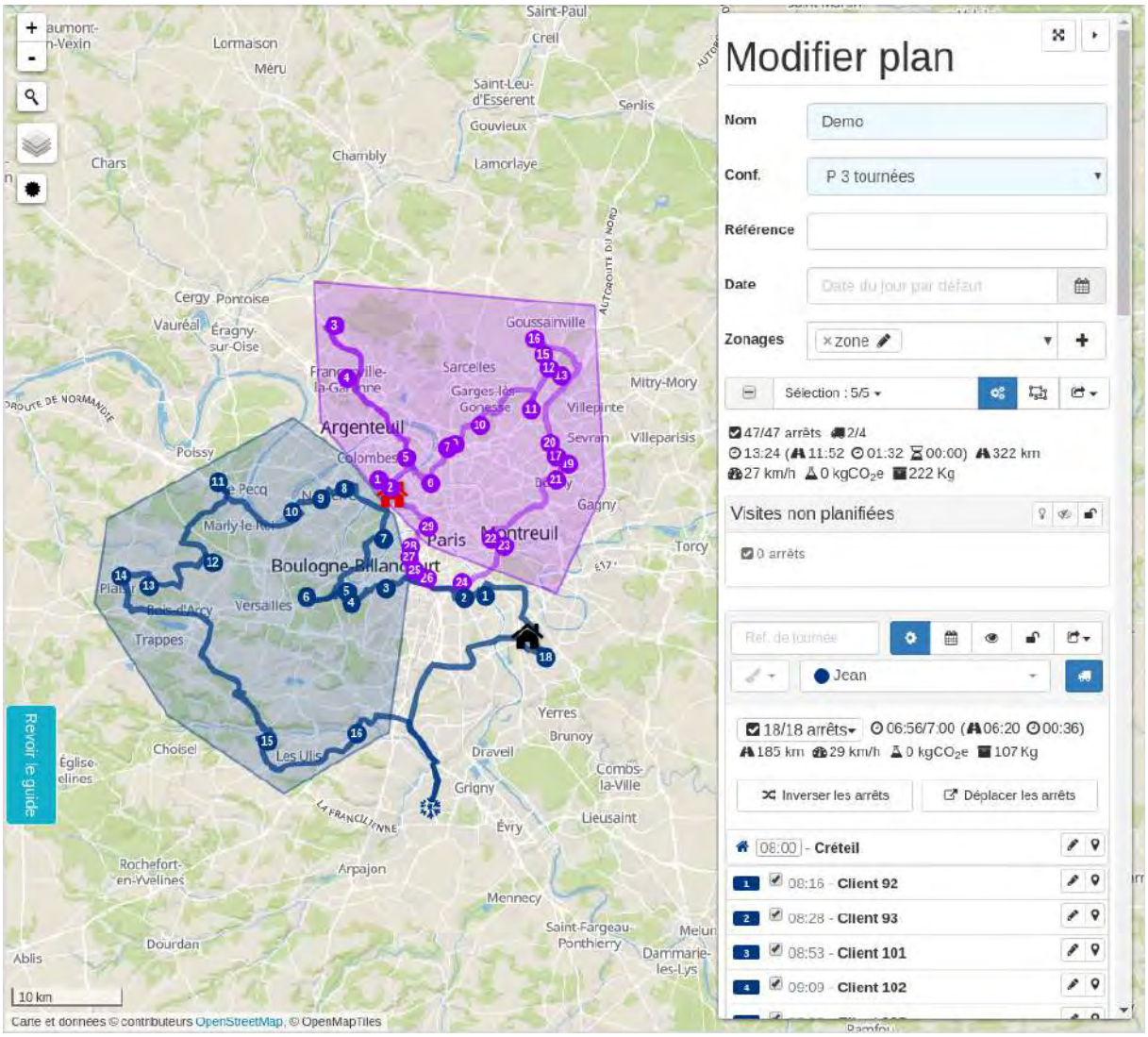

FIGURE A9 – Plan

Pour fournir davantage d'informations au planificateur, il est possible de masquer la carte afin de déployer les tournées et les placer côte à côte (Figure A10).

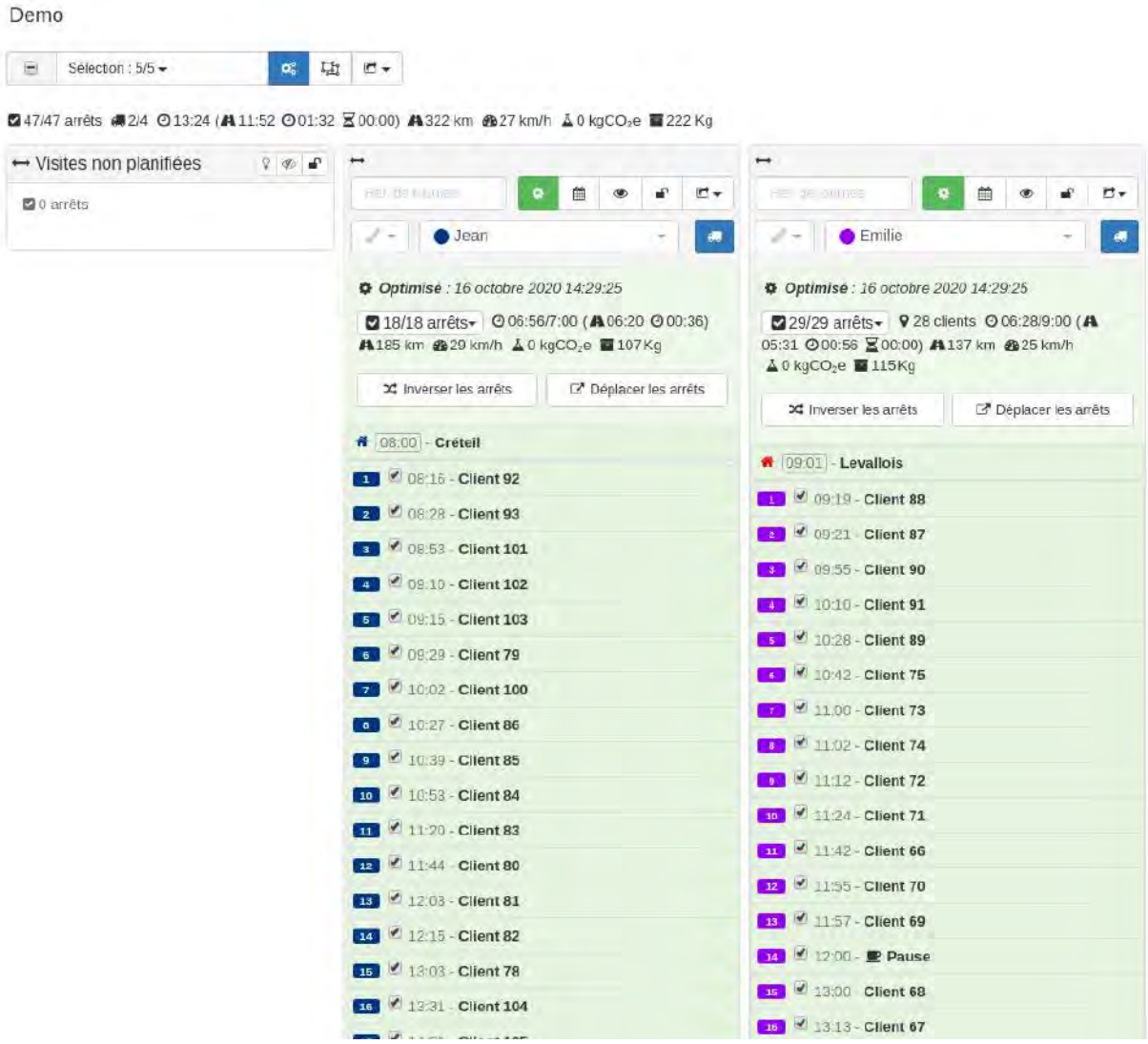

FIGURE A10 – Plan déployé

A.2 Détail des résultats

Les tables présentées dans cette section détaillent les résultats présentés dans la section 3.8. La table A6 montrent les différences de performances entre l'environnement utilisé pour réaliser les benchmarks dans cette thèse et les résultats utilisés comme point de comparaison. Cette table présente également le facteur multiplicatif appliqué pour normaliser les temps de résolutions.

				Modèle PPC			Sadykov et al.(2020)		Luo & Chen(2014)	
Instance	n	m	d	Coût	Temps	Écart	Coût	Temps	Coût	Temps
p01	50	4	4	**576,87**	286	0,00 %			**576,87**	2,77
p02	50	4	4	**473,53**	211	0,00 %			**473,53**	4,66
p03	75	2	5	**641,18**	228	0,00 %			**641,18**	7,14
p04	100	2	2	1008,98	560	0,79 %			**1001,04**	8,3
p05	100	2	2	753,86	156	0,51 %			**750,03**	12,52
p06	100	3	3	893,55	458	1,94 %			**876,5**	9,61
p07	100	4	4	904,37	448	2,54 %			**881,97**	7,28
p08	249	2	2	4746,75	187	8,55 %	**4372,78***	18052,63	**4372,78**	31,31
p09	249	3	3	4189,98	292	8,59 %	**3858,66***	7631,58	3860,28	30,87
p10	249	4	4	3931,27	195	8,27 %	**3631,11***	2198,25	3634,76	30,44
p11	249	5	5	3789,80	240	6,87 %	**3546,06***	2619,30	**3546,06**	31,75
p12	80	2	2	**1318,95**	197	0,00 %			**1318,95**	8,88
p13	80	2	2	**1318,95**	200	0,00 %	**1318,95***	0,88	**1318,95**	9,32
p14	80	2	2	**1360,12**	252	0,00 %	**1360,12***	0,88	**1360,12**	9,47
p15	160	4	4	2568,72	169	2,53 %			**2505,42**	27,23
p16	160	4	4	**2572,23**	362	0,00 %	**2572,23***	0,88	**2572,23**	26,8
p17	160	4	4	**2709,09**	197	0,00 %	**2709,09***	48,25	**2709,09**	26,94
p18	240	6	6	3890,24	510	5,06 %			**3702,85**	30,87
p19	240	6	6	**3827,06**	147	0,00 %	**3827,06**	74,56	**3827,06**	31,31
p20	240	6	6	4147,93	160	2,21 %	**4058,07***	131,58	**4058,07**	31,46
p21	360	9	9	6011,64	487	9,80 %			**5474,84**	45,29
p22	360	9	9	5760,71	152	1,03 %	**5702,16***	129,82	**5702,16**	45,72
p23	360	9	9	6224,16	85	2,39 %	**6078,75***	308,77	**6078,75**	44,85
pr01	48	4	4	**861,32**	296	0,00 %	**861,32***	2,63	**861,32**	4,22
pr02	96	4	4	1318,77	166	0,87 %	**1307,34**	226,32	**1307,34**	8,74
pr03	144	4	4	1827,41	370	1,31 %	**1803,8***	545,61	**1803,8**	14,27
pr04	192	4	4	2131,08	273	3,54 %	**2058,31**	1203,51	**2058,31**	20,97
pr05	240	4	4	2590,69	230	11,13 %	**2331,2***	19947,37	**2331,2**	35,1
pr06	288	4	4	3074,22	600	14,87 %	**2676,3***	21000	2677,64	36,7
pr07	72	6	6	1089,56	198	0,00 %	**1089,56***	14,91	**1089,56**	7,14
pr08	144	6	6	1696,36	363	1,89 %	**1664,85***	457,9	**1664,85**	20,1
pr09	216	6	6	2285,21	295	7,13 %	**2133,2**	1215,79	**2133,2**	30,29
pr10	288	6	6	3468,46	227	20,97 %	**2867,26***	76105,26	2868,26	39,03
				Moy.	279	3,72 %		6905,30		22,16

Table A1 – MDVRP

			Modèle PPC			Sadykov et al.(2020)		Penna et al.(2019)	
Instance	n	m	Coût	Temps	Écart	Coût	Temps	Coût	Temps
HVRP_DLP_75	20	3	**452,85**	161	0 %	**452,85***	0	**452,85**	166,66
HVRP_DLP_92	35	3	**564,39**	141	0 %	**564,39***	507,89	**564,39**	454,49
HVRP_DLP_93	39	6	1043,81	222	0,66 %	**1036,99***	86,84	**1036,99**	13,94
HVRP_DLP_94	46	5	1386,11	222	0,57 %	**1378,25***	14,91	**1378,25**	18,51
HVRP_DLP_55	56	3	10351,23	283	1,04%	**10244,34***	33,33	**10244,34**	381,84
HVRP_DLP_52	59	3	4071,67	260	1,1 %	**4027,27***	681,58	**4027,27**	213,24
HVRP_DLP_08	69	4	4685,1	187	2,03 %	**4591,75***	1,75	**4591,75**	206,82
HVRP_DLP_39	77	5	3050,49	230	4,51 %	**2918,87***	1514,91	2921,36	284,8
HVRP_DLP_70	78	4	6878,84	561	2,91 %	**6684,56***	730,7	**6684,56**	138,73
HVRP_DLP_82	79	3	4787,3	404	1,46 %	**4718,27***	3183,33	4766,74	56,56
HVRP_DLP_06	84	3	12495,16	384	6,95%	**11682,98***	137,72	11688,64	330,16
HVRP_DLP_36	85	6	5867,54	473	3,22 %	**5684,62***	78,07	**5684,62**	548,02
HVRP_DLP_43	86	7	9349,4	271	7,01 %	**8737,02***	37,72	**8737,02**	653,27
HVRP_DLP_01	92	4	9500,94	168	3,16 %	**9210,14***	1295,61	**9210,14**	35,33
HVRP_DLP_09	95	4	7970,43	371	4,88 %	**7599,72***	7157,89	7603,38	205,98
HVRP_DLP_90	102	4	2397	449	5,22 %	**2278,05***	127842,11	2346,43	165,97
HVRP_DLP_15	105	3	8534,58	425	3,81 %	**8221**	189473,68	8260,65	244,41
HVRP_DLP_84	105	4	7494,26	207	3,69 %	**7227,88***	1106,14	7233,95	224,64
HVRP_DLP_81	106	4	10927,08	150	3,25%	**10583,6***	17684,21	10693,7	277,28
HVRP_DLP_29	107	6	10034,02	582	9,88%	**9132,03***	7263,16	9142,86	231,8
HVRP_DLP_05	108	4	12133,2	172	11,63 %	**10869,04***	25,44	10876,48	450,75
HVRP_DLP_87	108	4	3862,86	186	2,9 %	**3753,87***	1571,05	**3753,87**	297,31
HVRP_DLP_47	111	5	17147,1	204	6,13 %	**16156,12***	306,14	16206,88	996,66
HVRP_DLP_48	111	5	22285,34	153	4,84%	**21257,38***	1495,61	21320,3	225,57
HVRP_DLP_61	111	3	7433,66	294	1,94 %	7293	189473,68	**7292,03**	300,22
HVRP_DLP_10	112	4	2135,95	159	1,35 %	**2107,55***	980,7	**2107,55**	215,8
HVRP_DLP_30	112	3	6532,79	263	4,05 %	**6278,62***	99421,05	6313,5	82,45
HVRP_DLP_28	113	6	5865,48	316	6,15 %	**5525,9**	189473,68	5530,55	672,87
HVRP_DLP_53	115	3	6669,79	350	3,65 %	**6434,83***	6328,07	**6434,83**	27,01
HVRP_DLP_03	116	5	11443,08	268	6,85%	**10709,66***	64,04	**10709,66**	204,82
HVRP_DLP_11	119	5	3499,96	163	3,94 %	**3367,41***	1100	**3367,41**	16,78
HVRP_DLP_04	121	8	11175,02	218	4,29%	**10714,84***	43842,11	10748,17	346,01
HVRP_DLP_2A	124	4	8126,12	199	4,27 %	**7793,16***	89,47	**7793,16**	466,09
HVRP_DLP_83	124	4	10226,64	232	2,25%	**10001,8***	2734,21	10020,07	97,64
HVRP_DLP_68	125	4	9780,64	279	10,03 %	**8889,03***	9473,68	8970,63	342,33
HVRP_DLP_74	125	5	11987,94	237	3,46%	**11586,58***	226,32	**11586,58**	404,28
			Moy.	273,47	3,97 %		25151,02		277,75
							Ecart Moy.	0,24 %	

TABLE A2 – HFVRP - Instances jusqu'à 125 nœuds

			Modèle PPC			Sadykov et al.(2020)		Penna et al.(2019)	
Instance	n	m	Coût	Temps	Écart	Coût	Temps	Coût	Temps
HVRP_DLP_18	126	3	11538,86	344	19,59 %	**9649,05**	189473,68	9655,91	116,77
HVRP_DLP_24	126	5	9536,07	539	4,77 %	**9102**	189473,68	9126,55	542,09
HVRP_DLP_88	127	5	13521,94	456	9,17 %	**12385,74***	25,44	12388,23	70,34
HVRP_DLP_14	129	6	5988,9	178	6,19 %	**5639,98***	189473,68	5663,91	204,98
HVRP_DLP_51	129	3	7999,62	266	3,6 %	**7721,47***	1541,23	**7721,47**	246,93
HVRP_DLP_31	131	8	4256,59	523	4,03 %	**4091,52***	968,42	**4091,52**	136,07
HVRP_DLP_40	132	5	11944,7	161	8,04 %	**11056,13***	9421,05	11122,32	123,05
HVRP_DLP_89	134	5	7485,47	151	5,62 %	**7087**	189473,68	7091,99	427,18
HVRP_DLP_41	135	7	8436,93	318	11,33 %	7598	189473,68	**7578,53**	415,49
HVRP_DLP_34	136	6	6051,13	156	5,44 %	**5739,02***	3684,21	5751,05	409,72
HVRP_DLP_60	137	4	17811,19	362	4,7 %	**17012,42***	2349,12	17037,23	322,03
HVRP_DLP_73	137	5	10599,05	304	3,96 %	**10195,33***	111,4	**10195,33**	309,65
HVRP_DLP_26	141	5	6794,61	498	6,06 %	**6406,16***	32105,26	6433,23	350,19
HVRP_DLP_23	143	6	8073,52	516	4,28 %	**7742**	189473,68	7760,01	564,78
HVRP_DLP_85	146	4	9130,04	586	4,18 %	**8763,9***	8421,05	8795,31	139,47
HVRP_DLP_79	147	4	7602,15	352	4,74 %	**7257,97***	3842,11	7262,91	297,09
HVRP_DLP_66	150	4	13463,87	428	5,38 %	**12776,24**	189473,68	12783,94	429,49
HVRP_DLP_69	152	4	9736,01	174	6,67 %	**9127,16***	11368,42	9169,42	182,18
HVRP_DLP_76	152	8	12792,41	440	6,65 %	**11994,22**	189473,68	11994,4	0,01
HVRP_DLP_56	153	4	32750,59	159	5,97 %	**30905,95***	185789,47	31030,19	128,89
HVRP_DLP_86	153	5	9269,94	453	2,76 %	**9020,63***	27631,58	9038,03	258,71
HVRP_DLP_37	161	5	7329,06	453	7,17 %	**6838,72**	189473,68	6850,77	70,53
HVRP_DLP_64	161	3	17549,35	171	2,42 %	**17135,16**	189473,68	**17135,16**	171,01
HVRP_DLP_22	163	4	14012,85	574	7,38 %	**13050**	189473,68	13096,26	223,13
HVRP_DLP_57	163	4	47560,66	180	6,21 %	44782	189473,68	**44781,64**	229,11
HVRP_DLP_27	164	4	9376,92	593	11,33 %	**8423**	189473,68	8424,73	236,97
HVRP_DLP_07	167	5	8835,77	271	9,46 %	**8071,97***	637,72	8089,46	248,59
HVRP_DLP_35	168	6	10503,5	285	10,3 %	**9522,45***	172,81	9555,92	274,07
HVRP_DLP_45	170	3	11120,95	544	6,15 %	10477	189473,68	**10476,25**	502,97
			Moy.	348,87	6,67 %		101406,47		263,15
							Ecart Moy.	0,18 %	

TABLE A3 – HFVRP - Instances entre 126 et 170 nœuds

			Modèle PPC			Sadykov et al.(2020)		Penna et al.(2019)	
Instance	n	m	Coût	Temps	Écart	Coût	Temps	Coût	Temps
HVRP_DLP_80	171	3	7136,03	274	4,68 %	**6816,89***	41421,05	6826,38	320,06
HVRP_DLP_44	172	3	13545,65	503	11,1 %	**12192**	189473,68	12208,3	148,59
HVRP_DLP_54	172	4	11047,78	364	6,72 %	10352	189473,68	**10351,97**	282,99
HVRP_DLP_67	172	5	11514,79	547	6,89 %	**10772,81***	72631,58	10884,91	299,25
HVRP_DLP_63	174	5	22103,29	254	11,12 %	**19890,65***	17052,63	20075,44	559,97
HVRP_DLP_12	176	4	3591,14	264	1,33 %	**3543,99**	5210,53	3543,99	178,79
HVRP_DLP_42	178	7	12230,99	199	13,19 %	**10805,94**	189473,68	10817,9	220,14
HVRP_DLP_02	181	4	12571,18	538	7,91 %	**11649,81***	503,51	11675,26	35,33
HVRP_DLP_2B	183	4	8900,84	426	5,18 %	8463	189473,68	**8462,56**	201,97
HVRP_DLP_95	183	2	6472,84	342	4,9 %	**6170,2**	189473,68	6175,62	10,59
HVRP_DLP_71	186	3	11123,87	371	13,53 %	**9798,06***	13052,63	9892,58	81,08
HVRP_DLP_72	186	4	6399,4	507	9,01 %	**5870,43***	99526,32	5917	262,93
HVRP_DLP_50	187	6	12940,14	256	4,6 %	12371	189473,68	**12370,94**	669,15
HVRP_DLP_13	188	7	7012,21	326	4,72 %	**6696,43**	189473,68	**6696,43**	48,28
HVRP_DLP_33	189	7	10081,43	419	8,01 %	**9333,39**	189473,68	9410,99	764,49
HVRP_DLP_77	190	3	7794,23	143	12,68 %	**6917**	189473,68	6933,15	288,18
HVRP_DLP_78	190	4	7411,51	463	5,34 %	**7036**	189473,68	7045,3	188,3
HVRP_DLP_59	193	6	14896,51	447	4,3 %	**14283**	189473,68	14304,46	694,76
HVRP_DLP_91	196	4	6729,05	505	6 %	**6348,36**	189473,68	6390,24	10,38
HVRP_DLP_21	203	4	5309,1	577	3,29 %	**5139,84***	1320,18	**5139,84**	682,34
HVRP_DLP_38	205	5	11798,73	366	5,4 %	11195	189473,68	**11194,68**	386,06
HVRP_DLP_25	220	5	7636,11	505	5,95 %	**7207**	189473,68	7209,29	299,39
HVRP_DLP_58	220	6	25367,06	573	8,54 %	23371	189473,68	**23370,42**	318,88
HVRP_DLP_65	223	3	14096,86	522	8,08 %	13044	189473,68	**13043,54**	47,55
HVRP_DLP_17	224	5	5531,85	205	3,15 %	**5362,83***	184,21	**5362,83**	122,24
HVRP_DLP_62	225	5	26565,24	396	15,57 %	**22987**	189473,68	23010,35	73,11
HVRP_DLP_19	239	2	13320,69	547	13,98 %	11687	189473,68	**11686,39**	821,69
HVRP_DLP_32	244	8	10288,08	591	9,65 %	9383	189473,68	**9382,6**	208,37
HVRP_DLP_49	246	8	17815,52	292	10,1 %	16182	189473,68	**16181,17**	250,88
HVRP_DLP_46	250	5	26217,5	280	6,72 %	24567	189473,68	**24566,23**	280,42
HVRP_DLP_16	256	5	4204,03	184	1,13 %	4157	189473,68	**4156,97**	611,63
			Moy.	393	7,51 %		142558,83		302,19
							Ecart Moy.	0,21 %	

TABLE A4 – HFVRP - Instances avec plus de 170 nœuds

		Modèle PPC		VROOM		Sadykov et al.(2020)
Instances	n	Temps	Écart	Temps	Écart	Temps
C1_2	200	243	3,06 %	3,3	0,41 %	103,25
C1_4	400	205	8,28 %	24,8	0,71 %	1465,79
C2_2	200	272	3,52 %	3,3	1,03 %	22715,53
C2_4	400	182	11,59 %	22,6	4,28 %	81844,65
R1_2	200	340	7,4 %	5,3	3,03 %	10054,12
R1_4	400	160	14,26 %	32,5	4,85 %	170614,74
R2_2	200	414	6,68 %	6	7,16 %	6349,12
R2_4	400	168	16,47 %	29,6	11,84 %	134993,16
RC1_2	200	262	7,84 %	5,8	3,16 %	48671,75
RC1_4	400	112	13,57 %	30,9	5,05 %	189473,68
RC2_2	200	414	5,75 %	5,7	5,01 %	42603,25
RC2_4	400	148	13,14 %	22,7	7,63 %	146252,63
	Moy.	243	9,30 %	16	4,51 %	71261

Table A5 – VRPTW

	Vidal et al.(2013)	Luo & Chen(2014)	Penna et al.(2019)	Sadykov et al.(2020)	Our proposal
CPU	Opteron 2755 2.2GHz	Pentium 4 2.8 GHz	i7 2.93 GHz	E5-2680 v3 2.50 GHz	i7-7700HQ 2.80GHz
OS			Ubuntu 14.04		Linux Mint 20
Language	C++	C++	C++	C++	C++
Rating	445	502	1394	1810	2069
Speed factor	4,65	4,12	1,48	1,14	1

TABLE A6 – Performances relatives - cpubenchmark.net

A.3 Epona : fichier d'exemple

L'exemple A.4 présente un fichier JSON type pour l'utilisation du générateur d'instance Epona, présenté dans le chapitre 6. Ce fichier permet la construction d'une instance ayant au plus 250 nœuds qui correspondent aux restaurants et pubs à Bordeaux. Le dépôt est une enseigne d'ameublement et la flotte est composée de 3 trois types de véhicules, légers, moyens et lourds.

```
{
"request": {
"name": "Bordeaux-250-ikea-restaurant-pub",
"units": [{
"id": "kg"
}],
"depots": [{
"id": "ikea",
"definition": {
"tags": {
"brand": "IKEA"
},
"area_tags": {
"boundary": "local_authority",
"name":  "Bordeaux"
}
}
}],
"nodes": [{
"id": "resto",
"definition": {
"tags": {
"amenity": "restaurant"
},
"area_tags": {
"boundary": "local_authority",
"name":  "Bordeaux"
}
},
```

```
"demands": [{
"unit_id": "kg",
"distribution": {
"type": "normal",
"sign": "plus",
"mean": 0,
"deviation": 2,
"precision": 2
}
}]
}, {
"id": "pub",
"definition": {
"tags": {
"amenity": "pub"
},
"area_tags": {
"boundary": "local_authority",
"name":  "Bordeaux"
}
},
"demands": [{
"unit_id": "kg",
"distribution": {
"type": "normal",
"sign": "plus",
"mean": 0,
"deviation": 2,
"precision": 2
}
}]
}],
"vehicles": [{
"id": "light",
"number": 20,
"cost_fixed": 150,
```

```
"cost_time": 0,
"cost_distance": 0.00023,
"router_mode": "car",
"router_dimension": "time",
"start_depot_id": "ikea",
"end_depot_id": "ikea",
"capacities": [{
"unit_id": "kg",
"limit": 10
}],
"router_options": null
}, {
"id": "medium",
"number": 10,
"cost_fixed": 300,
"cost_time": 0,
"cost_distance": 0.0004,
"router_mode": "truck_medium",
"router_dimension": "time",
"start_depot_id": "ikea",
"end_depot_id": "ikea",
"capacities": [{
"unit_id": "kg",
"limit": 30
}],
"router_options": null
}, {
"id": "heavy",
"number": 4,
"cost_fixed": 500,
"cost_time": 0,
"cost_distance": 0.0007,
"router_mode": "truck_medium",
"router_dimension": "time",
"start_depot_id": "ikea",
"end_depot_id": "ikea",
```

```
"capacities": [{
"unit_id": "kg",
"limit": 50
}],
"router_options": {
"speed_multiplier": 0.75
}
}],
"configuration": {
"location_limit": 250,
"require_corine": true,
"require_matrix": true,
"require_solution": false
}
}
}
```

Exemple A.4 – JSON complet requête à Epona-API

Titre : Système d'aide à la décision pour le dernier kilomètre : application aux problèmes de tournées de véhicules riches

Mot clés : Système d'aide à la décision, problème de tournées de véhicule riche, clustering, API REST, Clustered VRP

Résumé : Le travail de recherche a porté sur la création de méthodes de résolution dédiées au VRP riche et de leur intégration dans un environnement intégré de résolution pour fournir aux utilisateurs des solutions qui correspondent aux besoins de la logistique du dernier kilomètre dans des temps de calcul acceptables. Ce positionnement signifie que les VRP traités interviennent essentiellement en milieu urbain. Ce milieu apporte des contraintes spécifiques, en particulier l'interdictions d'accès à certaines voies par certains véhicules, des temps de transport qui varient au cours de la journée et des horaires de livraisons stricts à respecter.

Ces considérations ont amené à concevoir des méthodes de résolution du VRP riche qui tirent profit des méthodes de clustering. En effet, ces méthodes permettent de réduire la combinatoire et de garantir des tournées localisées réduisant ainsi l'itinérance des chauffeurs sur plusieurs quartiers. Nous proposons alors la résolution d'une nouvelle variante du VRP, le Soft Layered Clustered VRP. Notre contribution porte d'une part sur l'association des caractéristiques du graphe routier et l'utilisation d'une méthode de clustering. D'autre part, nous proposons une approche de résolution du VRP riche constituée de deux étapes.

Title: Decision Support Sytem for the Last Mile : application to the rich vehicle routing problems

Keywords: decision support system, vehicle routing problem, clustering, REST API, Clustered VRP

Abstract: The research work focused on the creation of resolution methods dedicated to rich VRP and their integration in an integrated resolution environment to provide users with solutions that meet the needs of last mile logistics in acceptable computation times. This positioning means that vehicle routing problems occur mainly in urban environments. This environment brings specific constraints and in particular the prohibition of access to certain roads by certain vehicles and transport times that vary greatly during the day and strict delivery schedules to be respected.

These considerations have led to the design of rich VRP resolution methods that take advantage of clustering methods. Indeed, these methods allow to reduce the combinatoriality and to guarantee localized routes reducing the itinerancy of the drivers on several districts. We then propose the resolution of a new variant of the VRP, the Soft Layered Clustered VRP. Our contribution is based on the association of the characteristics of the road graph and the use of a clustering method. On the other hand, we propose a two-step approach to solve the rich VRP.