Approches mathématiques pour l'aménagement de zones commerciales : modèles linéaires, algorythmes et systèmes multi-agents

Cyril Sahuc

ACADÉMIE D'AIX-MARSEILLE
AVIGNON UNIVERSITÉ

THÈSE

Présentée à Avignon Université pour obtenir le diplôme de DOCTORAT

SPÉCIALITÉ : INFORMATIQUE

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

Approches mathématiques pour l'aménagement de zones commerciales : modèles linéaires, algorythmes et systèmes multi-agents

par
SAHUC Cyril

Soutenue publiquement le 15 février 2020 devant un jury composé de :

M.	Claude YUGMA	PR, Ecole Nationale Supérieure des Mines de St-Etienne - Laboratoire: LIMOS - UMR CNRS 6158	Rapporteur
M.	Adnan YASSINE	PR, Institut Supérieur d'Etudes Logistiques (ISEL) - Laboratoire: LMAH (Laboratoire de Mathématiques Appliquées du Havre)	Rapporteur

Remerciements

Je remercie...

Sommaire

Chapitre 1

Introduction

Sommaire

1.1 Contexte et enjeux

Les zones commerciales, le plus souvent situées à la périphérie des noyaux denses urbains, constituent un élément incontournable des territoires, tant par l'intensité de leur fréquentation (bien souvent elles sont devenues la polarité principale des agglomérations avant même les centres historiques), que par la place qu'elles occupent dans le paysage notamment des entrées de villes. En effet, depuis plus de 50 ans elles n'ont cessé de se développer, en lien notamment avec l'augmentation de la motorisation des populations, sur laquelle se base leur accessibilité. Ainsi , le nombre de ZAE (zones d'activités économiques qui correspondent au regroupement d'activités économiques de un ou plusieurs types : industriel, artisanal, service, commerce) oscillerait entre 24 000 et 32 000 selon les estimations faites en 2007 à l'occasion du Grenelle de l'environnement, soit 450 000 ha.

Aujourd'hui encore, les ouvertures et extensions de ZAE, ou de zones spécifiquement commerciales, demeurent très nombreuses. Ainsi, en France, entre 2012 et 2018, d'après les données fournies par l'inventaire biophysique de l'occupation des sols **Corine Land Cover** (SOeS, Union Européenne 1990-2018), l'étalement des zones commerciales et industrielles a consommé environ 30.000 hectares (environ 30.000 aussi entre 2000 et 2006 puis environ 40.000 entre 2006 et 2012), pour une augmen-

tation de plus +7% en 6 ans. Cette progression des ZAE se fait principalement sur des terrains qui n'étaient pas encore artificialisés en 2012 (plus de 15.000 hectares, selon les *Données des Répartitions des superficies de changements CLC 2012-2018*). Cette consommation foncière des ZAE s'avère supérieure à celle liée à l'extension du tissu urbain pour l'habitat qui n'était pour cette même période 2012-2018 que d'environ 20.000 hectares (la superficie totale des tissus urbains étant de 2.500.000 ha en 2012). Mieux, il a été montré que les surfaces spécifiquement commerciales augmentent depuis 15 ans de 3% par an. Depuis 50 ans en France ce sont ainsi 5 à 6 millions de m2 commerciaux qui sont autorisés chaque année" (rapport CGEDD mars 2017).

Si les définitions et délimitations exactes des ZAE et zones commerciales sont parfois discutées, et par là les chiffres de leur consommation foncière, il n'en reste pas moins évident qu'elle a été et reste très importante, comme chacun peut l'expérimenter dans presque toutes les entrées de villes.

Jusqu'à présent ce développement des zones commerciales s'est fait le plus souvent sans plan d'ensemble prédéfini, notamment parce qu'il est rare qu'un seul opérateur maîtrise la zone dans sa globalité. Le développement se fait au contraire au coup par coup, au gré des opportunités foncières et des velléités d'installation de telle ou telle enseigne. Cette logique du coup par coup s'est le plus souvent accompagnée d'une logique de sur-dimensionnement des parkings, ce qui ne posait pas de problèmes réels dans un contexte où l'espace (agricole le plus souvent) était abondant et peu cher. Le parking est en effet un élément central de la zone commerciale, même si lorsque l'on parle de zone commerciale, l'image qui vient d'abord en tête est celle de vastes parallélépipèdes bardés de tôles, bien avant celle des espaces de stationnement qui leur sont liés, alors que dans les faits ceux-ci occupent très souvent des surfaces supérieures aux surfaces de vente à proprement parler.

Or, depuis peu, cet état de faits, dressé ici à grands traits, change résolument, dans un contexte où les injonctions liées à la question du développement durable se font de plus en plus présentes chez les acteurs du territoire (citoyens, techniciens et élus) et finissent par se traduire dans le dispositif législatif. La consommation foncière au détriment d'espaces agricoles inquiète (elle correspond à un département tous les 10 ans tous usages confondus) . L'artificialisation des sols, liée notamment aux vastes

surfaces de parking, facilite le ruissellement et par voie de conséquence les risques en termes d'inondation. Par ailleurs, certaines zones, victimes de leur succès, étouffent sous le trafic automobile et les encombrements compromettant ainsi leur accessibilité et donc leur attractivité, au moins auprès des individus pour lesquels la valeur du temps est importante et qui se trouvent être en majorité les populations au plus fort pouvoir d'achat. Ce qui explique sans doute, avec la révolution numérique, la stagnation et même la régression récente des chiffres d'affaire de nombreux grands complexes commerciaux (« Les crises de la grande distribution » 2016).

Cette saturation des zones s'avère n'être ni bonne pour le consommateur, qui perd du temps, ni pour les commerçants, qui voient leur attractivité diminuer, ni pour l'environnement, puisque les encombrements s'accompagnent de sur-consommation de carburant et de sur-émissions de polluants.

1.2 Objectifs

Conscient de ces enjeux, le législateur s'efforce de réagir en édictant de nouvelles règles, à l'image de la récente loi Alur (loi pour l'accès au logement et un urbanisme rénové) qui durcit les limites de taille des parkings au regard des surfaces de vente : « l'emprise au sol des surfaces bâties ou non, affectées aux aires de stationnement, annexes d'un commerce soumis à l'autorisation d'exploitation commerciale, ne peut être supérieure aux trois quarts de la surface de plancher des bâtiments affectés au commerce » (l'ancienne réglementation imposait un plafond de 1.5 au lieu de 75%). Cette loi a été précédée par d'autres changements de réglementation en termes d'urbanisme commercial, dont le plus important est sans doute la mise en place des DAC (document d'aménagement commercial) qui précise le volet commercial des Schémas de Cohérence Territoriaux (Scot) dans une logique systémique du fonctionnement territorial.

Si ces nouvelles règles sont très importantes pour l'évolution à moyen ou long terme des structures commerciales, elles ne permettent pas à court terme de mieux gérer l'existant qui est en outre marqué par une forte inertie. La question qui se pose est donc de savoir à court terme, avant une éventuelle « révolution commerciale » (Moati 2011) qui transformera la nature du commerce et ses formes paysagères, en lien notamment avec le commerce électronique, révolution qui ne pourra bien

entendu se faire que dans un temps relativement long, s'il est possible d'optimiser l'existant de façon à satisfaire à la fois les consommateurs, les commerçants et l'environnement dans une logique de durabilité. Dit autrement, est-il possible d'optimiser la forme et le fonctionnement des zones commerciales actuelles (et en construction) au prisme du développement durable en jouant sur les leviers d'actions que sont :

- la localisation des générateurs de trafic que sont les magasins (essentiellement ceux à venir étant donné que l'existant est peu malléable) ;
- la localisation et la taille des parkings actuels ou à venir, ainsi que leur nombre ;
- le réseau routier qui dessert ces zones, en agissant sur les tracés et sur ses propriétés topologiques ;
- la signalisation, l'information, les tarifs des stationnements ;

En jouant sur ces différents leviers, l'objectif serait donc de parvenir à agir à la fois sur le temps d'accès pour les consommateurs aux commerces qu'ils souhaitent fréquenter, et sur la surface des parkings. Autrement dit, d'une part nous chercherons à fluidifier le trafic dans ces zones de forte affluence mais sans augmentation de la consommation d'espace, comme il était d'usage jusqu'ici ; d'autre part nous chercherons aussi à limiter au possible la surface consommée.

Précisons que cette problématique d'optimisation à l'échelle des zones commerciales et/ou d'activités peut se décliner à l'échelle de la ville tout entière. Les attracteurs ne sont alors plus limités aux commerces mais aussi aux concentrations d'emplois, aux lieux de services etc... Les évolutions actuelles qui voient de plus en plus les grands projets commerciaux s'articuler d'emblée avec la construction de logements, de bureaux, de lieux services et ainsi tendre vers une plus grande mixité des fonctions, militent d'ailleurs dans le sens de cette volonté de ne pas séparer la gestion du commerce des autres dimensions de la vie urbaine.

On pourrait penser, eu égard à sa place dans le fonctionnement des territoires, que de nombreux modèles existent pour optimiser le stationnement. En réalité, sans doute parce que la question de la rationalisation de l'usage de l'espace en milieu péri-urbain s'est posée tardivement, l'état de l'art effectué dans le chapitre suivant montre que le problème de l'optimisation du stationnement reste très largement posé.

Chapitre 2

Etat de l'art

Sommaire

2.1 Introduction

Rappelons en premier lieu que les commerces s'implantent dans une zone commerciale le plus souvent au coup par coup au gré des opportunités foncières. Les bureaux d'études qui travaillent sur le stationnement utilisent presque exclusivement des méthodes qualitatives et empiriques. Pour les constructions neuves types ZAE ils ont tendance, dans un premier temps, à sur-dimensionner les parkings, afin de conserver une certaine marge. En cas de saturation éventuelle du stationnement observée dans un second temps, un agrandissement proposé.

Les travaux présentés ici s'orientent principalement autour de l'amélioration des coûts de transport dans une zone commerciale, pour les consommateurs. Il s'agit d'une première étape essentielle avant de pouvoir agir sur la surface utilisée. En effet, la raison principale du surdimensionnement des surfaces destinées au stationnement reste de conserver une certaine fluidité de trafic. Nous ne chercherons pas dans cette étude à réduire ces espaces disponibles ou même à agir sur d'éventuelles tarification dans le but de décourager les usagers de cette zone afin de faire évoluer leur mode de transport (Darbéra 1999). Au contraire, il s'agit là dans un premier temps d'améliorer le trafic, pour éviter le recours

au sur-dimensionnement, suite à quoi, dans un deuxième temps, il faudra estimer la localisation et le nombre de place de parkings nécessaires.

Une **zone commerciale** telle que nous l'étudions ici est une zone d'activités regroupant plusieurs commerces qui en sont les principales sources d'attractivité. Elle est composée aussi d'un grand nombre d'axes routiers permettant le déplacement entre ces commerces. Dans la plupart des cas, un nombre assez réduit d'entrées/sorties permettent d'y entrer. Les nombreux usagers qui la parcourt sont majoritairement des consommateurs. Chacun visite un nombre assez restreint de commerces ou de centres commerciaux. La zone commerciale est aussi dotée d'une importante surface de stationnement, répartie en plusieurs parkings distincts autour des différents commerces.

On désigne par "cible" ou "objectif" d'un consommateur un commerce, parmi l'ensemble possible, que le consommateur à comme objectif de visiter. Nous supposerons qu'un sous-ensemble de cibles est connu pour chaque usager. Cette hypothèse de travail n'est pas restrictive. Elle se justifie par l'existence de plusieurs initiatives ou solutions de recueil de données de fréquentation des centres commerciaux parmi lesquels :

— la création récente d'un observatoire de fréquentation des commerces par la fédération du commerce spécialisé Procos (en partenariat avec Stackr []),
— ou encore la disponibilité d'applications de recueil de données.

Le coût de transport, associé aux cibles d'un agent, correspond au temps total, ou à la distance parcourue dans la zone, pour se rendre aux différents objectifs. Pour des localisations fixées des cibles, le problème consistante à déterminer le trajet correspondant au coût minimal, est une variante du problème bien connu du voyageur de commerce (TSP : Travelling Salesman Problem).

Le TSP est un problème académique général qui, dans sa forme standard, considère un réseau de villes dans lequel on peut se rendre directement d'une ville à une autre (peu importe le mode de transport) à un certain coût (direct), modélisé par des pondérations mises sur les arêtes du graphe associé (Gutin et Punnen 2006)). Si on assimile chaque commerce à une ville, la tournée des commerces peut être vue comme celle des villes du TSP. Mais, à la différence d'un TSP standard, la visite du réseau de villes (commerces) ne se fait pas "directement" mais via des parkings définissant un second réseau (de parkings). Les noeuds parkings permettent de passer du mode de transport "routier" au mode "piéton"

car les commerces ne sont atteignables qu'à pieds. Et leur présence et nature entraînent une contrainte particulière sur les types de trajets que peut faire un agent. Le dépôt de la voiture d'un agent sur un parking, pour passer au mode "piéton" implique nécessairement le retour de cet agent au même point de dépôt (pour y récupérer sa voiture).

S'il y a autant de commerces que de parkings comme c'est le cas actuellement en pratique (1 commerce par parking), et que l'on néglige la distance à pieds pour atteindre un commerce depuis son parking, alors le parcours optimal d'un agent est équivalent à celui obtenu en résolvant un problème TSP. Nous avons vu en introduction que l'un des enjeux majeurs de gestion des zones commerciales, justifiant l'étude proposée dans cette thèse, et justement d'éviter ce cas extrême, gourmand en ressources foncières, pour le remplacer par une solution de partage "optimale" des zones de stationnement. Il s'en suit aussi que le cas des "drives" permettant l'accès aux commerces directement en voiture pour récupérer une commande passée en ligne n'est pas considéré dans notre étude. A l'autre extrême, s'il n'y a qu'un parking pour tous les commerces, le trajet d'un agent relève également d'un TSP dont le noeud de départ et d'arrivée est ce parking. Pour tous les autres cas intermédiaires en nombre de parkings, nous nous trouvons face une variante non standard du TSP qui, à notre connaissance, n'a pas fait l'objet d'études.

Les travaux de la littérature les plus proches du problème décrit ci-dessus sont de deux types :

- Les études orientées "agents" : l'élément central dans ces études est l'agent. Dans certaines études basées sur les systèmes multi-agents, leur comportement est simulé de façon à reproduire le plus fidèlement possible leurs déplacements et interactions. Les simulation multi-agents permettent également d'évaluer l'impact sur l'usage de parkings, et les temps de recherche, des prix des places de stationnement et des informations données sur les places disponibles.
- Les études orientées "aménagement" : ici il s'agit plutôt d'étudier les aménagements de la zone avec un nombre plus restreint d'informations sur les agents. Ces études essaient (comme nous le ferons aussi) d'améliorer les conditions de trafic en agissant sur des leviers opérationnels comme les localisations (parkings, commerces,...). Les trajets des agents ne sont pas toutefois pas pris

en compte explicitement comme nous le proposerons.

2.2 Études orientées "Agent"

Beaucoup d'outils informatiques ont été développés au fil des dernières décennies dans le but de simuler le déplacement de personnes, que cela soit dans une ville ou un centre commercial. Dans ces simulations, des agents indépendants les uns des autres, vont avoir des tâches à effectuer en se déplaçant sur un graphe, et vont essayer de les réaliser en fonction d'un comportement qu'on leur a fixé. On appelle ces outils des **SMA : Systèmes Multi-Agents** (Ferber 1997). Pour les Systèmes Multi-Agents qui nous intéressent, les agents doivent atteindre un ou plusieurs objectifs (cibles) en se déplaçant sur un graphe. Ils peuvent échouer dans leur mission sous certaines conditions. Une pénalité est alors considérée quand un agent n'a pas pu atteindre son objectif. Par exemple, certains SMA estimeront que l'agent abandonne sa tentative de stationnement au bout d'un certain temps de recherche. Dans ce genre d'étude, on cherche à évaluer une situation existante. Il s'agit de systèmes qui évaluent un ensemble de paramètres fixes (en particulier les éléments du graphe et leurs caractéristiques dans le SMA) pour en déceler les défauts (par exemple un manque de places de stationnement à certains endroits) ou à comparer deux jeux de données (par exemple, l'impact de la suppression d'un parking, en comparant les résultats des simulations avec et sans ce parking). Ces systèmes ne sont nativement pas faits pour optimiser les leviers vus précédemment, à savoir les localisations des générateurs de trafic et / ou celles des parkings et leur dimension. Par simulation, la seule possibilité de connaître les meilleurs placements de plusieurs commerces et parkings est de simuler toutes les combinaisons possibles. Ce qui n'est guère envisageable du fait de la combinatoire très importante des possibilités de placement. S'ajoute à cette combinatoire, le fait que les simulations Multi-Agents sont le plus souvent coûteuses en temps de résolution (c'est en particulier le cas pour le SMA MatSim que nous utilisons (Horni, Nagel et K. W. Axhausen 2016)).

Dans la catégorie SMA répondant à une problématique proche de la nôtre on trouve le système **ParkAgent** (Benenson, Martens et Birfir 2008). Ce système a pour particularité de s'appuyer sur la recherche de places de parking, qu'elles soient "on-street" (c'est à dire des places de stationnement le long d'une rue) ou "off-street" (toute aire de sta-

tionnement qui n'est pas le long d'une rue). Le trafic est pris en compte ainsi que la "patience" des agents pour trouver une place. La recherche de place on-street s'effectue d'après une règle de probabilité établie en fonction de l'estimation de places de stationnement restantes avant destination (cette estimation provient de la proportion en nombre de places de stationnement libres constatée sur le passage de l'agent et de la longueur du trajet restant à parcourir avant destination). Un des points particulièrement intéressant de ce Système Multi-Agents est de représenter le comportement réel de recherche de place de stationnement des agents. Avec ce système, il est possible d'évaluer assez fidèlement les saturations en terme de stationnement ainsi que les espaces peu utilisés. Le modèle fournit aussi le temps de "cruising" (temps de recherche d'une place de stationnement) résultant de la simulation, ainsi que les distances séparant les places finalement choisies de leurs cibles correspondantes. Ceci étant, la simulation ne se consacre qu'à cette phase de stationnement. La place de stationnement choisie y est bien considérée comme occupée pendant le temps prévu, mais il n'est pris en compte dans ces études ni le trajet de retour des agents, ni le fait que les agents puissent avoir éventuellement plusieurs objectifs. Pour notre cas d'étude l'ensemble du trajet des agents est important, en particulier les trajets d'objectif à objectif.

D'autres projets ne simulent pas le comportement des usagers, mais proposent d'agir directement sur l'environnement, en essayant d'influencer le comportement des usagers dans le but d'améliorer la fluidité du trafic. Parmi ces projets nous pouvons citer le ***LA Express Park™*** (*LA Express Park* 2012) et ***SF Park*** (*SF Park* 2010) respectivement en application dans les villes de Los Angeles et San Francisco (initialement, mais LA Express Park a aussi été exporté pour les villes de Venise et d'Hollywood). Il s'agit à la fois d'agir sur les prix du stationnement en fonction de la demande (Pierce et Shoup 2013) ainsi que sur l'information, par des applications en ligne permettant d'informer les usagers du prix et localisation des places disponibles. Des capteurs de présence sur l'ensemble des places de stationnement concernées permettent d'obtenir ces informations. Les variations de prix sont en effet souvent étudiées, comme un moyen efficace de libérer de la place disponible sur des secteurs trop encombrés. Pour inciter les usagers à utiliser d'autres stationnements plus disponibles (sur lesquels les prix auront été réduits). L'objectif principal de ces programmes est *«d'accroître la disponibilité des places de stationnement et de réduire les embouteillages et*

la pollution.»[traduction, à propos de *LA Express Park*™] (Glasnapp et al. 2014). Notons que dans la mesure où nous nous sommes intéressés aux zones commerciales dont le stationnement est en majorité gratuit, la prise en compte du prix de stationnement sort du cadre de notre travail.

MatSim

MatSim Horni, Nagel et K. W. Axhausen 2016 est un Système Multi-Agent open-source implémenté en Java qui permet de simuler le déplacement d'agents (trafic) sur un réseau. Le trafic y est limité par des capacités de flux propre à chaque axe routier avec lesquelles des embouteillages (ou ralentissements) sont simulés si le flux avoisinent ces capacités. La particularité de MatSim par rapport à d'autres SMA, est d'obtenir des résultats généralement très proches de la réalité au niveau des temps de transport, notamment grâce à cette prise en compte du trafic. Ces résultats sont obtenus après plusieurs itérations de simulation, où les agents ont la possibilité ou non (suivant un facteur aléatoire) d'améliorer leurs itinéraires en fonction des temps de transport constatés. C'est justement cet aspect de calcul de temps de transport que nous allons exploiter dans ce SMA. Le système permet, comme tout SMA, de réaliser des évaluations, ou éventuellement des comparaisons de jeux de données. Dans une simulation MatSim, les agents essaient d'améliorer leurs trajets, au fil des itérations, mais il n'est pas possible d'optimiser directement les aménagements (localisations des commerces ou parkings, réseau,...).

Un module de **MatSim**, appelé **Location Choice** (Horni, Scott et al. 2009), permet à un agent de pouvoir choisir entre plusieurs lieux possibles à atteindre pour une même tâche. Néanmoins, cela n'est pas applicable à notre cas. Ce module optimise, pour chaque agent, le choix d'un aménagement à atteindre parmi plusieurs aménagements disponibles pour réaliser une même activité. Ce que nous souhaitons est directement agir sur la localisation d'un même et unique aménagement parmi plusieurs localisations possibles qui initialement ne sont pas affectées. Ce qui n'est pas réalisable sous MatSim et n'est d'ailleurs pas vraiment dans l'esprit du SMA où le choix de chaque agent est indépendant. En effet, même si l'on considérait toutes ces localisations disponibles comme pouvant réaliser ladite activité dans le but de déduire la meilleure localisation, le choix d'un agent se portant sur une localisation devrait forcer tous les autres agents à atteindre cet objectif sur exactement la même

localisation, car notre aménagement à localiser est unique et ne peut être localisé qu'en un seul des lieux disponibles. Dans le cas du module **Location Choice** chaque agent pourra librement accéder à l'activité dans chacune des localisations disponibles pour celle-ci sans se soucier pour cela du choix des autres agents.

De la même manière un autre module permet aux agents de choisir un stationnement idéal pour son parcours : **Parking Choice** (Waraich et K. Axhausen 2012). Ce dernier correspond au choix de parkings qu'auraient à réaliser nos agents entre tous les parkings de notre problème, qui là, ne sont pas des objectifs ciblés mais simplement des outils. Chaque parking sera des aménagements où se déroule exactement la même activité de stationnement, permettant d'accéder par la suite au réseau piéton. Cependant, ce module ne sera pas utilisé, pour des raisons de cohérence entre les optimisations qui seront vues dans les chapitres qui suivent et l'utilisation qui sera faite de MatSim (recalculer des temps de transport en fonction de chemin déjà choisis par une optimisation). Il pourrait cependant être intéressant de réaliser des expériences similaires avec ce module.

2.3 Études orientées "Aménagements"

L'outil **ParkFit** (Levy et Benenson 2015) utilise un algorithme glouton pour simuler l'utilisation des parkings par les agents afin d'estimer la saturation de ces derniers. L'agent n'y est pas vraiment représenté, seule leur destination respective aide à prédire le parking utilisé (le plus proche parking qui n'est pas déjà utilisé). Les agents (ou plutôt, leur destination) sont traités un par un dans un ordre aléatoire. Une fois tous les agents placés, en résulte une estimation de la saturation des parkings par zone ainsi que des moyennes de distance entre les destinations souhaitées et le parking utilisé pour les atteindre. Ce type d'outils s'utilisera plutôt dans le contexte d'une zone d'habitation, où presque tous le monde va souhaiter être en stationnement pendant une longue période (la nuit). Il n'est pas adapté à une zone où les temps de stationnement sont courts et où se succéderont différents agents sur une même place de stationnement. De la même manière que les SMA, ce type d'outil a vocation d'évaluation et non d'optimisation. Au cour du chapitre 6 nous utiliserons nous aussi une méthode similaire basée sur l'aléatoire, pour représenter un aspect de la réalité qui lui aussi est aléatoire. Par aléatoire, il faut entendre

qu'il s'agit de quelque chose que nous ne pouvons pas prédire à notre échelle. Ce facteur aléatoire pour nous ne reposera pas sur l'ordre d'arrivée, à l'image de **ParkFit**, mais du temps d'arrivée. Ceci pour estimer le nombre de place de stationnements nécessaires en connaissant les utilisations qui vont avoir lieu au cour d'une période prédéfinie. Il faudra cependant réitérer un très grand nombre de fois le même processus pour obtenir des résultats représentatifs.

La plupart des travaux qui ont pour but d'optimiser les localisations, que ce soit de commerces ou de parkings, vont "graviter" autour des problèmes p-médian (Mladenović et al. 2007). Il sera le plus souvent proposé différentes heuristiques pour les résoudre. Par exemple, dans sa thèse en 2002, BARAY réalise une application reposant sur la résolution de p-median successifs pour optimiser le développement d'un réseau de magasin (Baray, Ding et Abdellaoui 2013). Il va, un peu comme nous souhaitons le faire, optimiser les localisations de plusieurs commerces, à des échelles plus grandes (villes, régions, France entière). Néanmoins, il s'agira là, non pas de permettre à des agents d'accéder plus rapidement à un ou plusieurs objectifs qu'ils souhaitent atteindre, mais d'améliorer l'accessibilité d'une même enseigne pour le plus grand nombres possibles d'agents. Cependant, il est au-préalable nécessaire pour nous de trouver des méthodes de résolution exactes, sous peine de n'avoir aucun point de comparaison pour établir si les potentielles heuristiques que l'on souhaitera effectuer sont efficaces ou non.

Il est également important de citer d'autres travaux qui, même si ils ne considèrent pas les leviers d'optimisation que nous avons définis, seraient complémentaires à notre étude par des améliorations au niveau des aménagements d'une zone commerciale. Par exemple, quand la quantité d'espace physique disponible pour un parking est plus contraignante que ce que la législation ne permet, les études réalisées par Porter (Porter et al. 2013) permettent d'agencer au mieux un espace disponible afin d'optimiser le nombre de places de stationnement effectives qui pourraient y être présentes. Dans une autre optique, à l'aide de ces mêmes travaux, et ce même si l'espace disponible n'est pas contraignant, un meilleur agencement des places de stationnement pourra aussi limiter la surface qui sera consommée , et donc réduire l'impact néfaste sur l'environnement. Il est aussi d'une grande importance, pour limiter l'impact causé sur l'environnement, de se référer aux bonnes pratiques en aménagement (qui relèvent souvent plus de l'expérience que d'un aspect pu-

rement théorique). On peut trouver par exemple le recueil réalisé par la Communauté métropolitaine de Montréal (2013).

2.4 Conclusions et choix

Les travaux "orientés agents", en formalisant de façon très fine toutes les composantes du système zone commerciale (réseau de transport, commerces, parkings et agents) parviennent par simulation informatique à reproduire assez fidèlement le fonctionnement des zones. En particulier, la simulation peut faire émerger des phénomènes réels de congestion résultants de la connaissance des trajets des agents. Cependant, chercher la configuration "idéale" de la zone revient à simuler et évaluer toutes les possibilités de positionner les commerces et parkings entre eux, de dimensionner les parkings et de concevoir un réseau de transport. Le fait que la combinatoire qui en résulte rend inappropriée une telle approche pour la résolution de notre problématique, mais nous pourrons nous servir de ces travaux dans l'objectif-clé de recalculer rigoureusement les coûts de déplacement entre les différents points d'intérêt, en aval de nos optimisations. En effet, le trafic routier pourra être fortement altérés par les congestions, suite aux modifications de localisation que l'on effectuera au niveau des aménagements.

A l'inverse, les modèles "orientés aménagements" sont plus simples à mettre en oeuvre, notamment parce qu'il font fi du trajet des agents. Cependant, ils ne peuvent faire état des phénomènes de congestion qu'il est important de ne pas sous-estimer dans une approche d'optimisation du fonctionnement de la zone. De plus, ils ne proposent pas de modèles permettant d'agir directement, à travers des variables, sur les différents leviers d'action identifiés en introduction. A la place, il s'agit d'algorithmes "heuristiques" n'optimisant qu'un des leviers, souvent difficilement adaptable pour un autre ou ne considérant pas le principe de stationnement. Un autre problème quant à l'adaptabilité de ces travaux à notre problématique est l'impossibilité d'évaluation. Il faudrait pour cela avoir une méthodologie permettant de connaître un résultat théoriquement optimal pouvant faire office de mesure qualitative.

La nouvelle approche que nous proposons se veut hybride, en tirant profit des méthodes décrites dans la littérature. Nous adoptons tout d'abord une vue orientée "aménagement" en considérant un réseau comportant des noeuds où peuvent être localisés les commerces ou les par-

kings, et des liens (logiques) valués représentant le déplacement entre ces noeuds et le temps de transport nécessaire. Dans cette représentation, le chemin exact emprunté par les usagers à travers le réseau de routes n'est pas considéré. Seul importe le fait de savoir que l'on peut aller d'un noeud origine vers un noeud destination dans un temps estimé supposé constant et connu. Nous parlerons de réseau "logique" en ce sens que les liens ne sont pas les routes réelles, mais les possibilités de déplacement. Le réseau dit "physique" désignera le réseau de routes réelles. Considérer un réseau logique avec des temps constants de transport sur les liens revient à omettre dans un premier temps les phénomènes de congestion, qui justement peuvent faire fluctuer ces temps. Du fait de la simplicité de cette représentation logique, elle autorise à formuler le problème d'optimisation de la localisation/dimensionnement des parkings, et de la localisation des parkings de façon compacte sous la forme d'un programme mathématique linéaire.

Diminuer le nombre de zones de stationnement (une mutualisation globale des parkings) pourrait entraîner une baisse considérable de l'espace utilisé pour satisfaire la demande. Ce que confirme Garcez et Mangin 2014 : «En mutualisant mieux les parkings, en permettant de passer plus facilement d'un commerce à l'autre à pied, on peut optimiser le nombre de places nécessaires et permettre une certaine densification.». Pour cet aspect, la mutualisation des parkings pourra être imposé, par limitation du nombre de parking sur les modèles linéaires.

Dans un deuxième temps les phénomènes de congestion peuvent être capturé par une représentation orientée "agents". Dans cette approche, le réseau physique est considéré : routes réelles, positions géographiques des commerces et parkings. Les positions "optimales" des commerces et des parkings sont obtenues par résolution du programme mathématique de la première étape ; de même que l'itinéraire des usagers à travers le réseau logique. Ces données permettent de mettre oeuvre des simulations multi-agents du même type que celles décrites dans les approches microscopiques de l'état de l'art. Pour ce faire, le SMA **MatSim** sera retenu, d'une part parce qu'il est open source et d'autre part parce que sa construction rigoureuse, sa déjà longue histoire et ses nombreuses applications tant théoriques que pratiques, en font un modèle très fiable qui produit des résultats très convaincants en termes de simulations de trafic.

mettre des références d'usage de matism

Matsim sera donc mobilisé parce qu'il permet d'évaluer dans quelle mesure la fréquentation simultanée de mêmes routes par plusieurs agents pendant la simulation peut produire de la congestion, ce qui permettra de modifier les temps de transport utilisés sur le réseau logique en fonction de l'observation des temps de déplacement des agents pendant la simulation. Connaissant les temps de transport simulés (supposés plus proche de la "réalité"), le réseau logique peut-être actualisé et le modèle d'optimisation résolu de nouveau. Et ainsi de suite,... Idéalement, ce couplage convergerait vers une solution qui resterais une des plus efficace en prenant en compte le trafic (la congestion amenant à des ralentissements). Seul l'expérimentation peut confirmer ou non cette hypothèse.

L'objet du chapitre 3 est de présenter de façon détaillée le réseau logique adopté et le modèle d'optimisation associé. En réalité plusieurs modèles ont été développés au cours de nos recherches, chacun améliorant par une formulation différente le précédent. Nous verrons que ces modèles sont tous linéaires en nombres entiers et généralisent des problèmes standards de recherche opérationnelle. Leur complexité en termes de nombre de variables et de contraintes, bien que polynomiale, croît toutefois de façon importante avec la taille des données. Si bien, qu'en plus d'une résolution directe, plusieurs schémas heuristiques que nous détaillerons ont été proposés. Ils font l'objet du chapitre 4. Un couplage avec une simulation informatique sera étudier afin de prendre en compte le trafic dans nos solutions. Ce couplage est expliqué au chapitre 5 ainsi que les résultats obtenus. Pour les optimisations réalisées, une notion très importante est le nombre d'utilisation d'un parking. Cette valeur est utilisée pour fixer une capacité maximale sur les parkings ainsi que pour témoigner, après optimisation, du besoin de ce dernier. Le chapitre 6 présentera une méthode qui permet de convertir pour un parking, par estimation, un nombre de places en nombre d'utilisation de ce dernier par les agents (et inversement). En plus de cela, cette méthode permettra d'estimer si une réduction du nombre de parking, par une mutualisation généralisée, peut réduire le nombre de places nécessaires au bon fonctionnement de la zone.

Chapitre 3

Modélisations du Problèmes du Voyageur de Commerce avec prise en compte des parkings : TSP-PR

Sommaire

3.1 Introduction au problème

3.1.1 Le réseau logique

Pour optimiser le fonctionnement des zones commerciales, en particulier au niveau de l'accessibilité aux commerces, une zone commerciale peut être représentée par un graphe dont les noeuds constituent les parkings et les localisations des commerces. À ces noeuds se rajoutent ceux modélisant les points d'entrée et de sortie de la zone commerciale (en général peu nombreux). Les arcs de ce graphe représente les possibilités de déplacement entre les différents noeuds. Chaque arc est caractérisé par un mode de déplacement pris de façon unique parmi deux possibilités (routier, pédestre), et est valué par un coût de transport. Le coût de transport sera défini ici comme étant le temps de transport, néanmoins il pourrait être autre. Nous admettons, dans notre modèle, que le mode associé à un arc est défini de façon unique par ses noeuds origine et destinations. Ainsi, le mode est supposé "routier" si l'arc relie :

- une entrée à un parking,
- deux parkings entre eux,
- un parking à une sortie.

Sont considérés comme se faisant "à pieds", les arcs allant :

- d'un parking à un commerce,
- d'un commerce à un commerce,
- d'un commerce à un parking.

Les noeuds "entrée" (dans la zone) et sortie jouent un rôle particulier. Dans le réseau considéré, il n'y a pas d'arcs liant ces noeuds aux commerces. On suppose en effet que l'entrée dans la zone se fait exclusivement en mode routier. De même, partant du principe qu'une personne entrant dans la zone y visite des commerces, il n'y a pas de liens directs des entrées aux sorties.

Définition 1. *Nous appelons* ***réseau logique*** *(de la zone commerciale), ou encore* ***graphe IOPD*** *(dans un contexte plus général), le graphe ainsi construit. Une représentation schématique de celui-ci est donnée à la figure 3.1. À ce réseau peuvent s'associer plusieurs problèmes relatifs à la circulation.*

On pourrait décomposer ce réseau logique en deux réseaux :

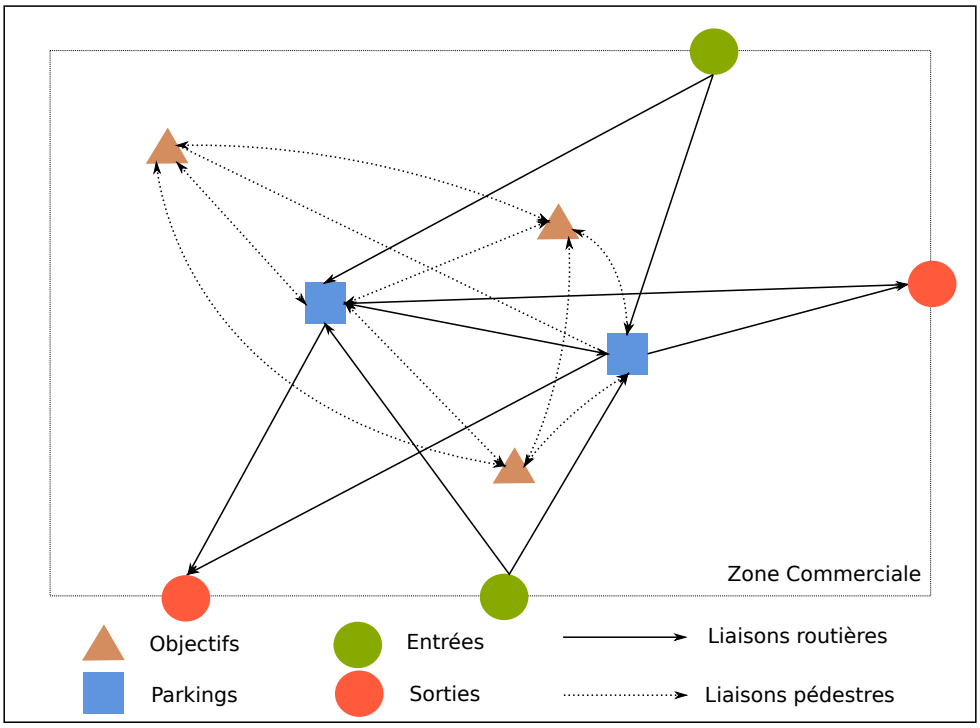


FIGURE 3.1 – Réseau logique de la zone commerciale

- *Le **réseau pédestre** : l'ensemble des parkings et des objectifs, avec pour liaison uniquement celles s'effectuant à pied.*
- *Le **réseau routier** : l'ensemble des parkings, des entrées et des sorties, avec pour liaison uniquement celles s'effectuant en voiture.*

Nous allons procéder en deux étapes pour expliquer concrètement notre problème général. Tout d'abord en traitant le problème d'un agent seul qui cherche à atteindre des commerces. Puis, en faisant évoluer ce premier problème en intégrant les différents leviers qui doivent être optimisés.

3.1.2 Le problème de l'agent

Sur le réseau circulent en pratique des individus que nous nommerons de façon équivalente des **agents**. Ces agents y entrent en voiture par les noeuds d'entrée, à une heure donnée, se rendent à différents commerces en exploitant les parkings disponibles et le mode pédestre pour aller des parkings aux commerces ou se déplacer de commerces à commerces. Enfin, ils sortent de la zone commerciale en voiture par les noeuds de sortie. Pour chaque agent, ses noeuds d'entrée / sortie ainsi que la liste des commerces qu'il souhaite visiter sont supposés connus.

Par ailleurs, afin de simplifier le modèle, les heures d'arrivée des agents seront omis. En pratique, ceci peut être effectué en s'intéressant par exemple à une période représentative où les agents circulent dans la zone. Un aménageur pourrait ainsi s'intéresser à une période de forte affluence (le samedi après-midi) en dimensionnant ce réseau dans ce "pire des cas", ou au contraire à une influence moyenne. Le problème de la période adéquate des données à considérer n'est pas pris en compte par notre contribution. Nous verrons toutefois par la suite comment à partir d'un modèle dont la temporalité est omise celle-ci peut être ré-introduite, par simulation, pour décider de la capacité de stationnement nécessaire (Chapitre 6).

Définition 2. *Nous appellerons* ***objectifs****, les commerces qu'il souhaite visiter.*

Du fait du mode de déplacement unique associé à chaque arc, il n'est possible à un agent d'atteindre un commerce qu'à pied (en ayant utilisé auparavant un parking). A cela s'ajoute la contrainte suivante :

Définition 3. *Si un agent passe du mode routier au mode pédestre, après avoir déposé sa voiture dans un parking, il ne pourra retrouver le mode routier qu'après avoir récupérer sa voiture. Nous appelons cette contrainte* ***la contrainte de retour au parking (PR : Parking Return)****.*

La figure 3.2 étaye cette contrainte par des exemples de parcours possibles ou non. La figure 3.3 quant à elle montre un exemple de parcours correct d'un agent.

Le problème de l'agent, sous contrainte de retour au parking, consiste à déterminer un chemin de son entrée à sa sortie lui permettant de visiter tous ses objectifs à moindre coût (comme l'illustre la figure 3.4). Le coût d'un chemin étant la somme des coûts de déplacement sur tous les arcs empruntés.

Si l'agent souhaite minimiser ses temps de déplacement, chaque objectif ne sera visité qu'une et une seule fois. Si par ailleurs nous nous plaçons dans le cas particulier où il n'existe qu'un seul parking, ce problème se ramène à trouver une tournée optimale, à partir du parking, passant une et une seule fois par chaque commerce objectif. On reconnaît là, dans ce cas particulier, le problème du voyageur de commerce

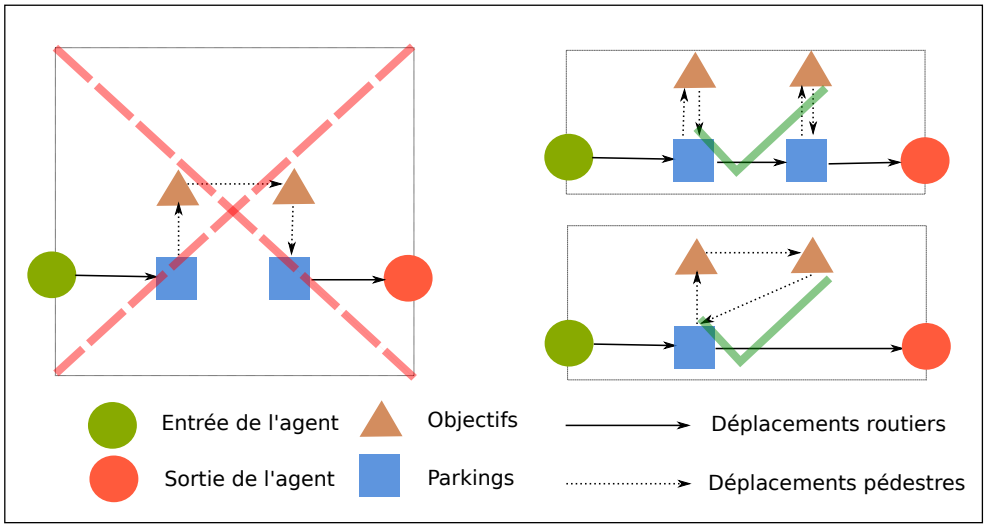


FIGURE 3.2 – La contrainte de retour au parking

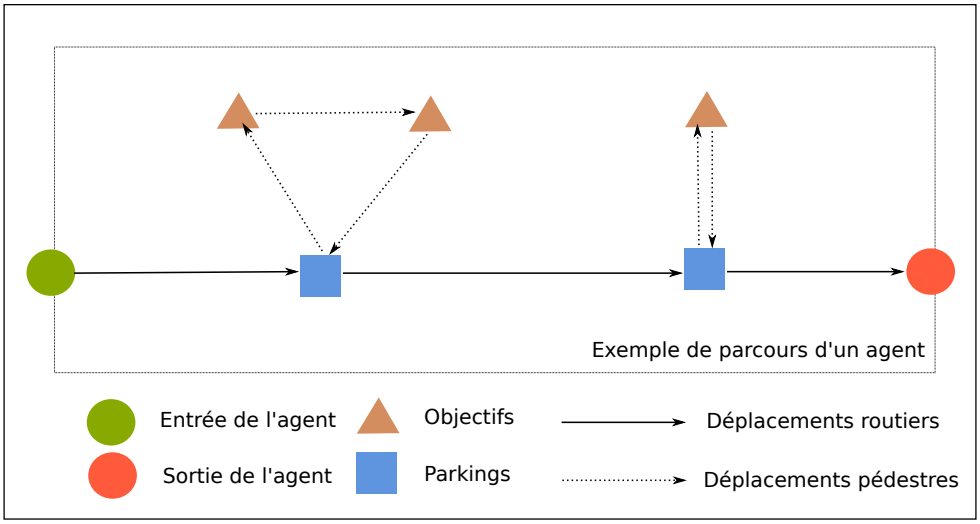


FIGURE 3.3 – Exemple de parcours d'un agent.

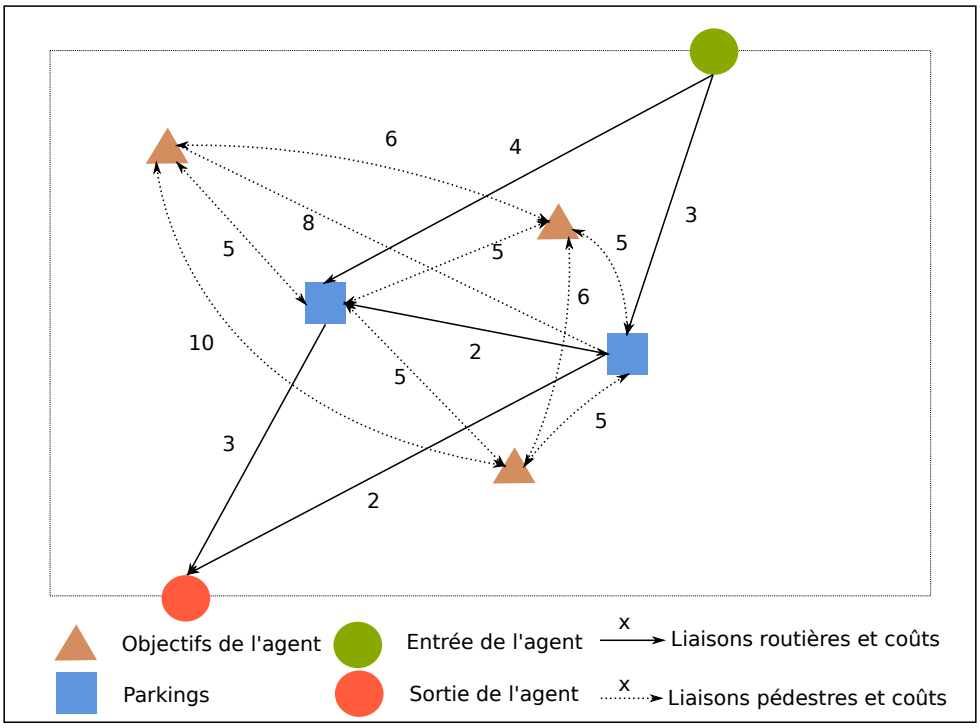


FIGURE 3.4 – Exemple de problème pour l'agent : Quel est le chemin le plus court pour atteindre tous les objectifs sur ce réseau logique, avec la contrainte de retour au parking ?

(Traveling Salesman Problem, TSP) abondamment étudié dans la littérature (GUTIN et PUNNEN 2006). En conséquence, on peut en déduire que le problème de l'agent fait partie de la classe des problèmes NP-complet puisque généralisant le problème TSP qui l'est lui-même. Nous désignerons par l'acronyme **TSP-PR** le problème de l'agent (TSP with Parking Return).

Théorème 1. *TSP-PR est NP-Complet.*

Typiquement, une zone commerciale est caractérisée par l'afflux d'un grand nombre d'agents dont les circulations sont à l'origine de congestions routières ou de congestion au niveau des parkings (manquent de places disponibles). La solution optimale du problème TSP-PR revient à trouver ce que peut faire de mieux un agent en termes de minimisation de temps de déplacement pour atteindre ses objectifs. Mais il est important de noter, qu'en pratique, l'itinéraire optimal calculé n'est pas forcément celui réalisé par les agents car il suppose la connaissance des temps de déplacement sur tous les axes du réseau.

Chaque agent résolvant ce problème, nous mesurons la qualité de la zone par la somme de tous les temps de déplacements ainsi obtenus. Dans

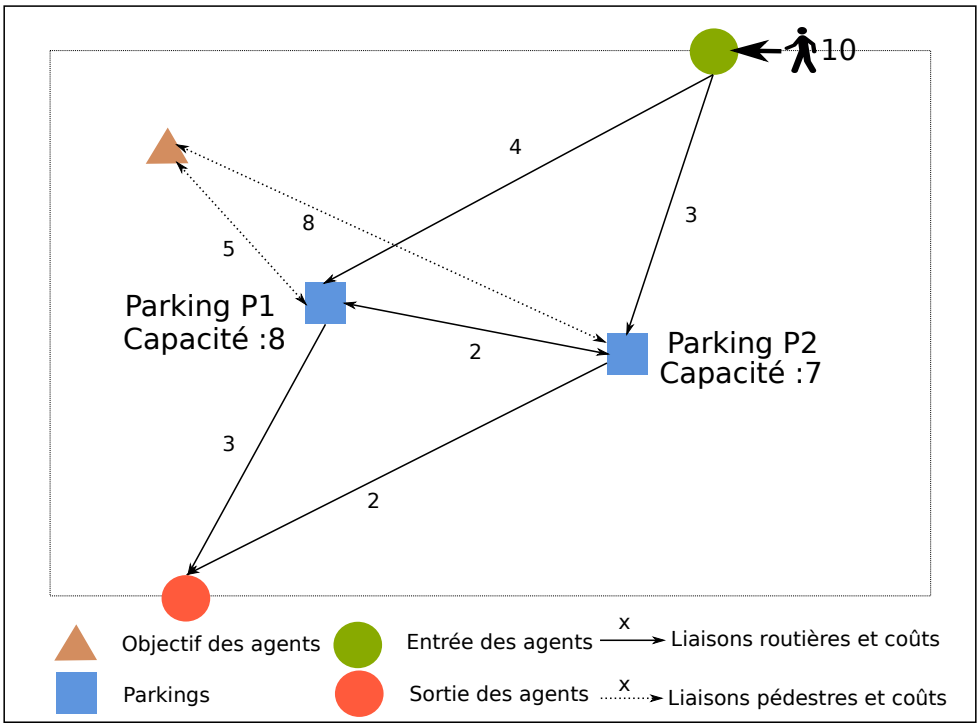


FIGURE 3.5 – 10 agents doivent atteindre le même objectif, cependant certains seront contraints de ne pas emprunter leur trajet le plus court.

le cas (théorique) où le temps de transport sur un arc est indépendant du nombre d'agents l'utilisant et qu'il n'y a pas de contrainte de capacité sur les parkings (accueil d'un nombre infini de personnes), cette mesure peut-être obtenue en résolvant indépendamment un problème TSP-PR par agent, car le parcours d'un agent n'est pas contraint par celui d'un autre. Cela deviendra le cas par la suite, principalement pour deux raisons. Premièrement, comme les parkings pourront avoir des capacités finies, la somme des temps de déplacement optimaux n'est pas la résultante des itinéraires calculés individuellement. Ce cas de figure est illustré par la figure 3.5, où 2 des 10 agents seront contraints de devoir utiliser un itinéraire qui ne leur est pas favorable pour cause de saturation d'un parking. De même, quand par la suite nous chercherons la meilleure localisation des objectifs, chaque agent influencera le placement des objectifs, et donc de ce fait le trajet des autres agents.

Un problème plus général consiste alors à déterminer les chemins de plusieurs agents aux données connues (entrée/sortie, objectifs), permettant de minimiser la somme des coûts de déplacements en respectant les contraintes de retour parking pour chaque agent. Ce problème sera désigné par le sigle **MA-TSP-PR** (Multi-Agent TSP with Parking

Constraint). Même si, comme nous venons de le voir, le MA-TSP-PR revient à une série de problèmes TSP-PR indépendants les uns des autres, la formulation générale que nous présentons a le mérite de pouvoir ensuite introduire rapidement des variables de décision relatives à la localisation des commerces et des parkings, ainsi qu'aux contraintes liées aux capacités des parkings. Différentes notations seront vues pour spécifier exactement de quelle variante du MA-TSP-PR nous parlons (si les affectations liées aux localisations entrent en jeu, s'il y a des capacités ou choix sur les parkings) dans les sections qui suivent.

3.1.3 Le problème des localisations

Plusieurs agents circuleront sur le graphe avec chacun leur problème de déplacement. Il est considéré que chaque agent va toujours utiliser son plus court chemin (le moins coûteux). Des objectifs sont ciblés par chaque agent. Ces objectifs ne sont plus considérés comme localisés, c'est à dire qu'ils n'appartiennent plus au graphe et sont remplacés sur ce dernier par des localisations d'objectifs (non-affectés). Le problème est d'optimiser cette fois-ci la localisation des objectifs, leurs affectations sur les localisations disponibles. Le coût total de déplacement des agents sera à minimiser. Pour nommer cet aspect supplémentaire sur un problème de type MA-TSP-PR nous rajouterons le suffixe **"-DL" (Destination Location)**. La figure 3.6 nous montre un exemple de problème **MA-TSP-PR-DL**.

Il est aussi possible de vouloir localiser des parkings. Il ne s'agira en fait pas vraiment de localisations. Les parkings ne sont jamais ciblés, Ce ne sont que des outils permettant d'atteindre les objectifs. Il faut simplement de permettre à chaque parking du réseau logique d'exister ou non. Ainsi, il est possible dans les modélisations de limiter le nombre de parking existant et donc de trouver les localisations de parkings optimales. Afin de nommer l'ajout de cet aspect sur un problème de type MA-TSP-PR, le suffixe "-PR" est remplacé par **"-PRL" (Parking Return and Location)**. Le problème **MA-TSP-PRL** est illustré par la figure 3.7.

Pour finir il est aussi question de pouvoir limiter le nombre d'utilisation d'un parking. C'est à dire de spécifier une capacité maximale, propre à chaque parking, d'agents qui l'utilise. Le suffixe "-PR" ou "-PRL' est remplacé respectivement dans ce cas par **"-PRC" (Parking Return and Capacity) ou "-PRCL"**.

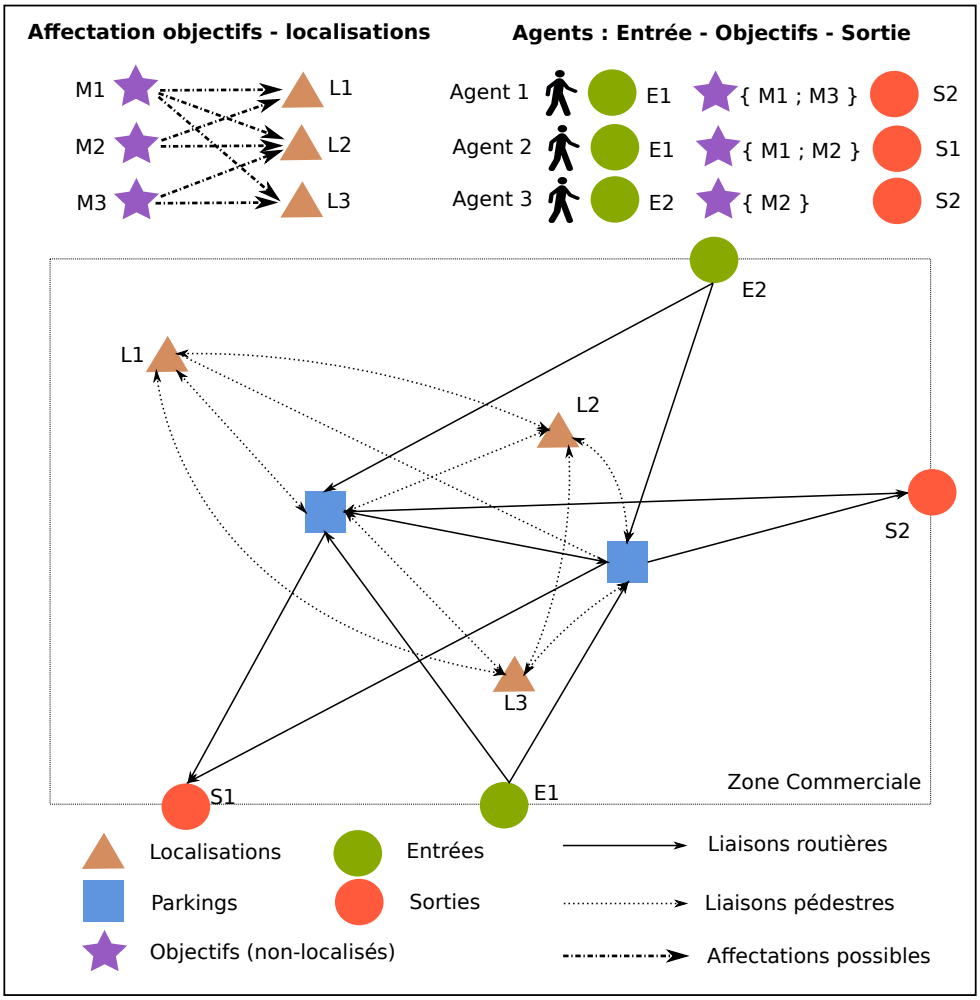


FIGURE 3.6 – Problème MA-TSP-PR-DL : Quelle est l'affectation optimale des objectifs D1,D2 et D3 sur les localisations L1, L2 et L3 qui minimise le coût de déplacement global des agents ?

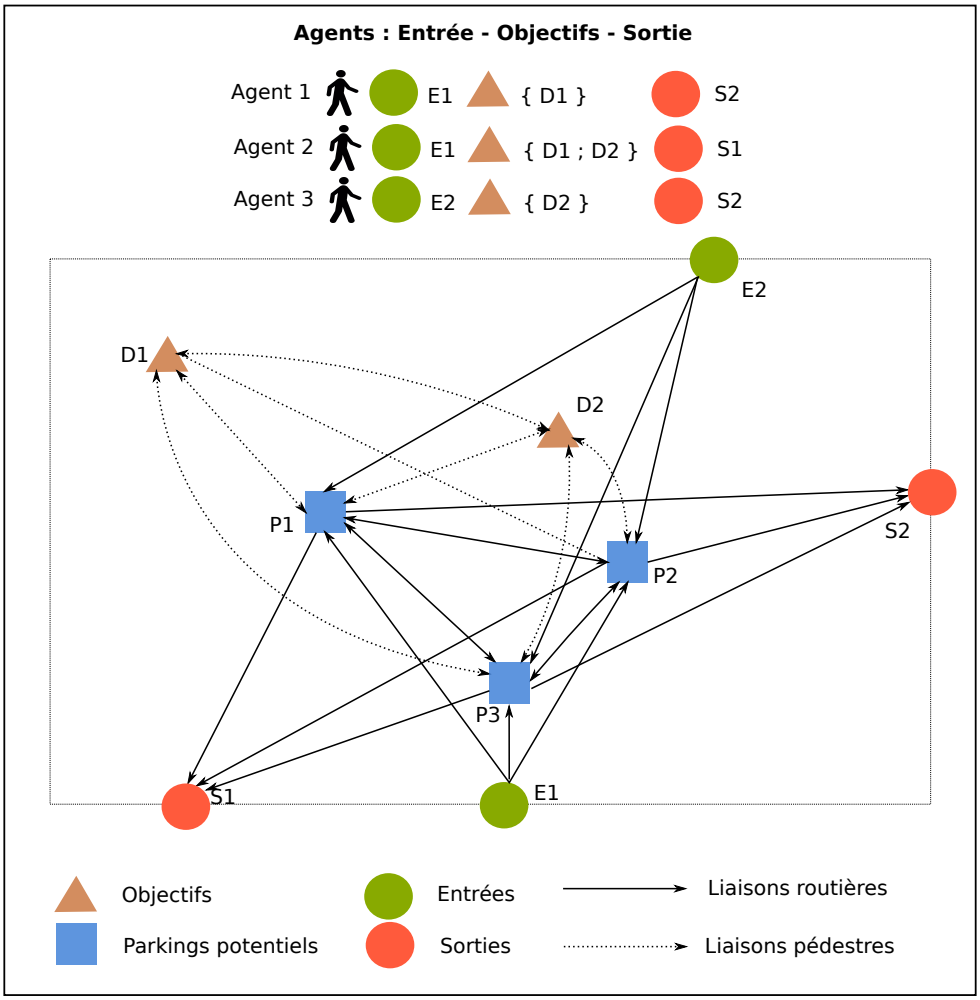


FIGURE 3.7 – Problème MA-TSP-PRL : Parmi P1, P2 et P3, quels seraient les 2 parkings, si l'on ne pouvait en avoir que 2, permettant de minimiser le coût de déplacement global ?

Ces aspects seront vus plus en détail dans la section 3.4 où ils s'appliqueront directement sur des modélisations linéaires.

Récapitulatifs des différents sigles de nos problèmes

Le sigle des différents problèmes de type **TSP-PR** que nous étudierons se décompose en 4 partie, toujours dans l'ordre suivant :

- **MA-** : Multi-Agent. Partie facultative, présente si le problème concerne plusieurs agents.
- **TSP** : Travelling Salesman Problem. Problème racine dont tous nos problèmes sont une généralisation. Cette partie est toujours présente.
- **-PRCL** : Parking Return, Capacity and Location. Explique à quelles contraintes sont soumis les parkings. "-PR" toujours présent indique qu'il y a la contraintes de retour au parking. La lettre "C", facultative, signifie quand on l'ajoute que les parkings auront une capacité maximale d'utilisation. La lettre "L" aussi facultative, indépendamment du "C", indiquera qu'il y a un choix de localisation de parking a effectuer.
- **-DL** : Destination Location. Partie facultative qui spécifie que les objectifs sont à localiser.

Mis bout à bout et en isolant entre parenthèse chaque aspect indépendant et facultatif pouvant s'ajouter au problème, cela nous donne la formulation générale suivante :

(MA-)TSP-PR(C)(L)(-DL)

3.2 Le Problème du Voyageur de Commerce avec Retour au Parking

3.2.1 Définition du TSP-PR

Voici des définitions plus formelles du TSP-PR et du MA-TSP-PR, qui ont été abordés dans la section précédente :

Définition 4. *Le problème **TSP-PR** (Travelling Salesman Problem with Parking Return) est le problème consistant à trouver le plus court chemin pour un agent cherchant à atteindre un ou plusieurs objectifs dans le réseau logique représentant la zone commerciale.*

Définition 5. *Le problème* ***MA-TSP-PR*** *(Multi-Agent Travelling Salesman Problem with Parking Return) est un problème TSP-PR avec plusieurs agents. Il s'agit là d'une concaténation de plusieurs TSP-PR réunis avec pour but de minimiser la somme de chaque agent. Le* ***MA-TSP-PR*** *n'a pas vraiment de sens en tant que tel, mais est une base commune à plusieurs autres problèmes dont nous parlerons dans la section 3.1.2.*

Soit $G = (V,E)$ un graphe orienté représentant le réseau logique des entrées, des sorties, des parkings et des objectifs.

On note :

I : l'ensemble des entrées (Input)

O : l'ensemble des sorties (Output)

P : l'ensemble des parkings

D : l'ensemble des objectifs (Destination)

A : l'ensemble des agents

Pour chaque agent a, on suppose connue son entrée de $a \in A$, notée i^a ($i^a \in I$), sa sortie o^a ($o^a \in O$) et l'ensemble d^a des objectifs qu'il souhaite visiter ($D^a \subset D$). L'ensemble V des noeuds est défini par :

$$V = I \cup O \cup P \cup D$$

On note E l'ensemble des arcs orientés reliant les noeuds de V. Les entrées possèdent uniquement des arcs sortant, en direction de chaque parking. De la même manière, les sorties disposent uniquement d'arcs entrant, provenant de chaque parking.

On note aussi $E*$ le sous-ensemble d'arc de E correspondant a des arcs considérés comme faisant partie du réseau pédestre, c'est à dire :

- reliant les noeuds de P et les noeuds de D (dans un sens comme dans l'autre),
- reliant les noeuds de D entre eux.

De la même manière, tous les autres arcs, qui sont donc considéré comme faisant partie du réseau routier, seront dans un sous-ensemble, celui-ci simplement noté $E \backslash E*$. Ce sont ceux :

- reliant les noeuds de I en direction des noeuds de P,

- reliant les noeuds de P en direction des noeuds de O,
- reliant les noeuds de P entre eux.

On note c_{ij} le coût de transport pour se rendre de i à j $\forall i,j \in E$. Ce coût peut être de diverses natures en fonction de ce que l'on souhaite minimiser. Par exemple, la distance, le temps, le prix lié au déplacement. Cela n'aura aucun impact sur la modélisation en soit, même si évidement les résultats pourront être différents. Voulant fluidifier le trafic de la zone commerciale (même si ce n'est pas notre seul objectif), nous choisissons de considérer ce coût comme étant le temps de déplacement.

Chacun des deux sous-réseaux, c'est à dire le réseau routier et le réseau pédestre respecte l'inégalité triangulaire. C'est à dire :

$$c_{ik} \leq c_{ij} + c_{jk} \quad \forall i,j,k \in V/(i,j),(j,k),(i,k) \in E*$$
$$c_{ik} \leq c_{ij} + c_{jk} \quad \forall i,j,k \in V/(i,j),(j,k),(i,k) \in E \backslash E*$$

Par contre, l'inégalité triangulaire n'est pas applicable dans le réseau logique complet. En effet, il peut être moins coûteux de se déplacer depuis un parking vers un autre parking par le réseau routier avant d'atteindre un objectif à pied plutôt que de l'atteindre directement à pied depuis ce premier parking.

3.2.2 La contrainte de Retour au Parkings (PR)

Nous avons vu que le cas particulier du problème comportant un agent et un parking correspondait à un problème standard de voyageur de commerce (TSP). La différence majeure avec un TSP réside dans la contrainte dite de "retour au parking" qui découle de la nature du problème.

Définition 6. *À chaque fois qu'un agent sort d'un parking, à pied, il ne peut ré-emprunter le réseau routier qu'après être revenu sur ce même parking, à pied. Il ne peut donc, tant qu'il n'est pas revenu sur ce parking, utiliser que le réseau pédestre. On appelle cette contrainte* ***la contrainte de Retour au Parking****.*

Cette contrainte va engendrer deux difficultés :

- les parkings sont des noeuds intermédiaires dans le chemin des agents (ils ne sont pas des objectifs à atteindre) permettant de pouvoir rejoindre les objectifs accessibles uniquement à pied.

- au niveau des trajets des agents, il faudra veiller à ce que chaque "sous-trajet" partant d'un parking à pied forme un **cycles pédestres** revenant sur ce dernier.

Définition 7. *On appelle* **cycle pédestre** *du réseau logique un cycle constitué exclusivement de liaisons pédestres.*

Trois modèles linéaires satisfaisant cette contrainte de retour au parking ont été proposés. Ils sont détaillés dans la section suivante.

3.3 Formulations et résolutions exactes

3.3.1 Formulation avec des variables "agents" et "voitures"

Ce modèle est certainement le plus intuitif. Il s'inspire des contraintes physiques sur lesquelles repose notre problématique. Il va s'agir simplement de modéliser chaque aspect physique qu'engendre la contrainte de retour au parking. Cette contrainte est due aux interactions nécessaires entre un agent et son véhicule. Physiquement, un agent déposant son véhicule sur un parking pour rejoindre des commerces à pied, se doit de récupérer celui-ci à ce même parking s'il souhaite ré-emprunter le réseau routier. Nous allons donc considérer deux "entités physiques" :

- l'agent, qui circule sur les deux types de liaison (routière et pédestre), cherchant à atteindre optimalement ses objectifs.
- sa voiture, qui circule uniquement sur les liaisons routières, suivant un chemin à déterminer de l'entrée de l'agent à sa sortie.

La voiture n'étant pas autonome, sur les liaisons routières, le trajet d'un agent est de sa voiture doivent être identiques.

Notre modèle va donc utiliser deux jeux de variables. L'un exprimant le trajet d'un agent d'une part et l'autre de sa voiture. Des contraintes viennent ensuite s'assurer que la voiture ne se déplace pas toute seule.

Rappelons que le réseau logique où l'agent se déplace est modélisé par un graphe orienté G = (V,E) défini à la section 3.2. Le chemin d'un agent dans ce réseau, de son entrée à sa sortie, peut être vue comme une succession d'arêtes empruntées, e_0,....,e_q. "l" est la longueur du chemin. L'extrémité initiale de e_0 est l'entrée de l'agent. L'extrémité terminale de

e_q est sa sortie. Deux arêtes consécutives ont une et une seule extrémité en commun.

Définition 8. *On appelle* ***étapes*** *chacune des arêtes empruntées définissant un chemin de son entrée à sa sortie. Les étapes sont identifiées par leur numéro.*

De même, le chemin de la voiture, de son entrée à sa sortie, peut être vue comme une succession d'arêtes empruntées, e_0,....,e_q. "q" est la longueur du chemin. Comme la voiture n'est pas autonome et qu'elle ne peut être utilisée sur les liens pédestres on a forcément $q \leq l$.

Définition 9. *On appelle* ***étapes de la voiture*** *les étapes correspondant aux arrêtes utilisées par la voiture. Les étapes de la voiture ont leur propre numérotation, distincte de celle de leur agent respectif, comme on peut le voir sur la figure 3.8.*

Il est important de noter que les longueurs l et q ne sont pas connues à l'avance. Pour les besoins de présentation de certains résultats, on notera k_a^* le nombre d'étapes de l'agent a dans la solution optimale du problème.

Soit $a \in A$ un agent donné et $i \in V$ et $j \in V$ deux noeuds du réseau. On considère les variables suivantes :

$$y_{ij}^{ak} = \begin{cases} 1 & si\ a\ va\ de\ i\ vers\ j\ à\ l'étape\ k \\ 0 & sinon \end{cases}$$

$$\text{et, } x_{ij}^{ak} = \begin{cases} 1 & si\ la\ voiture\ de\ a\ va\ de\ i\ vers\ j\ à\ l'étape\ k \\ 0 & sinon \end{cases}$$

Ces variables sont des variables de chemin. Il est nécessaire pour avoir un nombre fini de variables de borner le nombre d'étapes. Pour un agent a et sa voiture, on peut établir une première borne triviale de la façon suivante.

Théorème 2. *Soit D^a l'ensemble des objectifs de $a \in A$. Le nombre optimal (k_a^*) d'étapes du trajet de a est borné supérieurement par*

$$k_a^* \leq 3 * |D^a| + 1$$

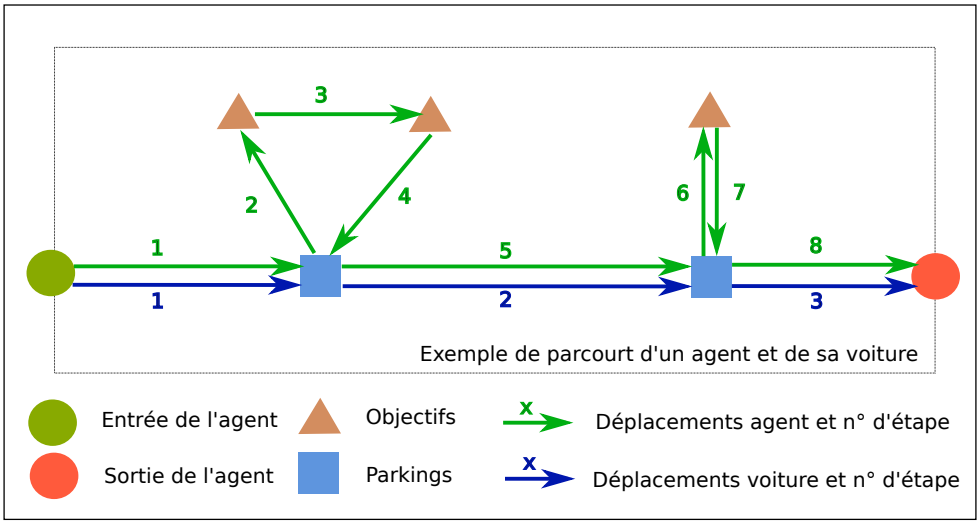


FIGURE 3.8 – Numérotation des étapes d'un agent et de sa voiture sur un exemple de chemin parcouru.

Démonstration. La preuve se déduit d'une stratégie de visite de l'agent. Nous donnons une solution de trajet correspondant à cette borne.

Si l'agent admet au moins un objectif, il devra obligatoirement réaliser 4 déplacements : se déplacer de son entrée vers un premier parking, et de ce premier parking vers le premier objectif qu'il atteindra ; ainsi qu'à l'opposé, quitter son dernier objectif pour récupérer son véhicule sur le dernier parking par lequel il est passé, et quitter ce parking en voiture pour se rendre à sa sortie. À ceci ne se rajoutera que les sous-trajets permettant de se déplacer entre deux objectifs. Ces derniers ne peuvent être que de deux natures possibles : soit ils sont constitués d'un seul déplacement, à pied, direct entre un objectif et le suivant ; soit ils sont constitués de 3 déplacements, un premier pour rejoindre à pied le parking sur lequel est la voiture de l'agent, un déplacement routier pour rejoindre un nouveau parking, et pour finir le déplacement à pied depuis ce deuxième parking pour atteindre l'objectif suivant. Au maximum donc, un sous-trajet entre deux commerces pourra comporter 3 déplacements. Le trajet de l'agent comportera un sous-trajet entre deux objectifs pour chaque objectif en plus du premier. Cette stratégie est illustrée dans la figure 3.9.

En sachant que s'il n'y a qu'un objectif l'agent va devoir effectuer 4 déplacements, et que pour chaque objectif qui se rajoutera après le premier, on pourra avoir au maximum 3 déplacements en plus. Soit D^a l'ensemble des objectifs de $a \in A$. Le nombre optimal (k_a^*) d'étapes du

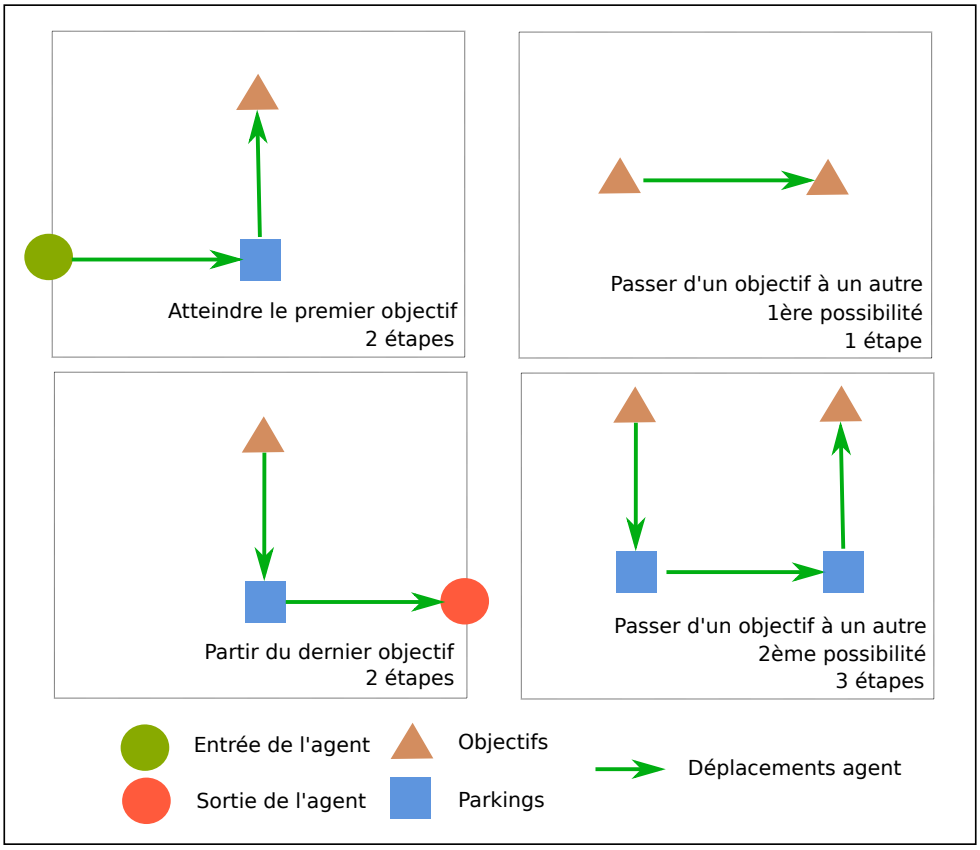


FIGURE 3.9 – Nombre d'étapes d'un agent.

trajet de a sera donc borné par

$$k_a^* \leq 4 + 3 * (|D^a| - 1)$$

qui se simplifie facilement par

$$k_a^* \leq 4 + 3 * |D^a| - 3$$

$$k_a^* \leq 1 + 3 * |D^a|$$

□

Soit K^a l'ensemble des entiers naturels compris entre 1 et cette borne.

$$\forall a \in A, \forall k \in K^a, 1 \leq k \leq 1 + 3 * |D^a|$$

De la même manière on pourrait différencier le nombre d'étapes de la voiture. Soit k_a^{**} le nombre d'étapes de la voiture de l'agent dans la solution optimale du problème. Celui-ci est borné supérieurement par

$$k_a^{**} \leq |D^a| + 1$$

Ce dernier étant toujours inférieur à k_a^* (une voiture ne peux faire plus d'étapes que son agent, car elle circule toujours avec son agent) et pour plus de clarté, on ne déclarera pas un ensemble de définition pour les étapes de la voiture. Les étapes de la voiture, au niveau de nos modélisations, utiliseront le même ensemble de définition K^a que leur agent respectif.

Le problème $(MA-TSP-PR)$ se modélise comme suit.

$$(P_1) : \text{Min} \quad \sum_{a\in A}\sum_{k\in K^a}\sum_{(i,j)\in E} c_{ij}y_{ij}^{ak}$$

$$\text{sujet-à :} \quad \sum_{(i,j)\in E} y_{ij}^{ak} \leq 1 \qquad \forall\ a\in A, k\in K^a \tag{3.1}$$

$$\sum_{j\in P} y_{i^aj}^{a1} = 1 \qquad \forall\ a\in A \tag{3.2}$$

$$\sum_{j\in P}\sum_{k\in K^a} y_{jo^a}^{ak} = 1 \qquad \forall\ a\in A \tag{3.3}$$

$$\sum_{j\in V} y_{ji}^{ak} = \sum_{j\in V} y_{ij}^{a(k+1)} \qquad \begin{array}{l}\forall\ i\in V\backslash\{I,O\},\\ k\in K^a,\ a\in A\end{array} \tag{3.4}$$

$$\sum_{j\in V} x_{ji}^{ak} = \sum_{j\in V} x_{ij}^{a(k+1)} \qquad \begin{array}{l}\forall\ i\in V\backslash\{I,O\},\\ k\in K^a,\ a\in A\end{array} \tag{3.5}$$

$$\sum_{j\in V}\sum_{k\in K^a} y_{jd}^{ak} = 1 \qquad \forall\ a\in A,\ d\in D^a \tag{3.6}$$

$$\sum_{k\in K^a} x_{ij}^{ak} = 0 \qquad \forall\ (i,j)\in E^*\ \forall\ a\in A \tag{3.7}$$

$$\sum_{k\in K^a} x_{ij}^{ak} = \sum_{k\in K^a} y_{ij}^{ak} \qquad \forall\ (i,j)\in E\backslash E^*\ \forall\ a\in A \tag{3.8}$$

La contrainte (3.1) indique qu'à une étape donnée d'un trajet d'un agent une seule arête du réseau est empruntée.

La contrainte (3.2) impose que l'étape 1 du trajet de a consiste à se rendre de son entrée vers un parking.

La contrainte (3.3) impose qu'il existe une étape k (la du trajet de a) pour laquelle il se rend du dernier parking utilisé à sa sortie.

Les contraintes (3.4) et (3.5) imposent une conservation de flux étape après étape.

La contrainte (3.6) permet de s'assurer qu'un agent visite tous les commerces de sa liste.

La contrainte (3.7) impose que la voiture ne soit pas utilisée sur les arcs correspondants au mode pédestre.

La contrainte (3.8) impose qu'il y ait autant de déplacements sur chaque liaison routière entre l'agent et sa voiture. Cela même si les numérotations d'étapes divergent entre l'agent et sa voiture (comme on a pu le constater sur la figure 3.8).

3.3.2 Re-formulation

Il est possible de trouver une nouvelle modélisation moins coûteuse en nombre de variables que la précédente. Celle-ci ne cherchera pas à intégrer des variables relatives au chemin de la voiture d'un agent pour pouvoir linéariser la contrainte de retour au stationnement.

Théorème 3. *La contrainte de retour au parking est équivalente à la contrainte suivante : Pour chaque agent a, sur chaque parking p, et en chaque étape k, il y a, parmi toutes les étapes suivantes, plus (ou autant) d'arrivées que de départ sur le réseau pédestre.*

Démonstration. Commençons d'abord par écrire ces contraintes mathématiquement. Tout d'abord la contrainte de retour au parking, initialement formulée, et que l'on appellera (C_1), est la suivante : à chaque fois qu'un agent sort d'un parking, à pied, il ne peut ré-emprunter le réseau routier qu'après être revenu sur ce même parking, à pied. Dit autrement, et du au fait que l'entrée et la sortie du graphe s'effectue depuis le réseau routier, dès lors qu'un agent sort d'un parking à pied, il doit avoir une étape de retour sur ce même parking, avec exclusivement des étapes pédestres entre ces deux étapes. Illustrée par la figure 3.10.

Contrainte (C_1) :

$$\begin{aligned} \forall a \in A, p \in P, k \in K^a & \\ \text{si} \quad & \sum_{d \in D} y_{p,d}^{a,k} = 1 \\ \text{alors} \quad & \exists\, l \in K^a, l > k \\ \text{tel que} \quad & \sum_{d \in D} y_{d,p}^{a,l} = 1 \\ \text{et que} \quad & \forall m \in K^a, k < m < l, \sum_{d_1, d_2 \in D} y_{d_1,d_2}^{a,m} = 1 \end{aligned}$$

Voyons maintenant la nouvelle contrainte proposée de retour au parking. Nous appellerons (C_2) cette formulation. Pour (C_2), on demande simplement à l'agent d'avoir, en toute étape $k \in K^a$ plus ou autant de retours pédestres sur un parking que de départs pédestres depuis ce dernier. Illustrée par la figure 3.11.

Contrainte (C_2) :

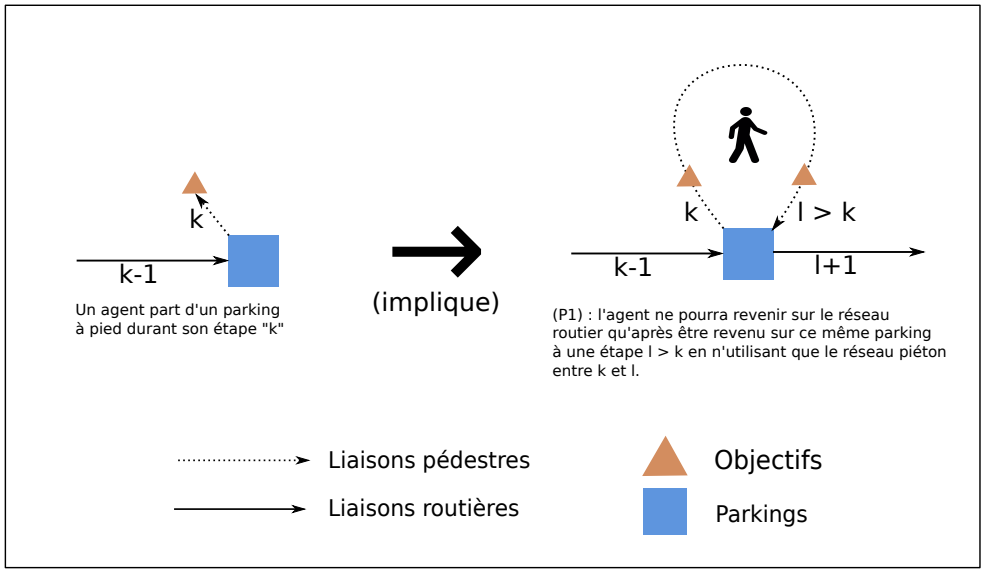


FIGURE 3.10 – Formulation (C_1) de la contrainte de retour au parking

$$\forall a \in A, p \in P, k \in K^a, \quad \sum_{d \in D} \sum_{\substack{l \in K^a \\ l \geq k}} y_{p,d}^{a,l} \leq \sum_{d \in D} \sum_{\substack{l \in K^a \\ l \geq k}} y_{d,p}^{a,l}$$

Nous devons démontrer que ces deux formulations (C_1) et (C_2) sont bien équivalentes.

Tout d'abord, le fait que (C_1) implique (C_2) est trivial. En effet, si chaque fois que l'on part d'un parking par le réseau pédestre, un retour y est assuré à une étape supérieure, on est certain que sur n'importe quel parking, à n'importe quelle étape (pour un agent donné) il y aura plus ou autant de retours à pieds sur parking que de départs. La réciproque néanmoins n'est pas aussi simple. Il nous faut donc maintenant prouver que la contrainte (C_2) implique le respect de la contrainte (C_1).

Nous considérons donc maintenant que la contrainte (C_2) est respectée. Notre but étant de prouver par la suite qu'elle implique (C_1), ce qui prouvera notre hypothèse de départ.

Pour commencer, nous allons faire ressortir une égalité sur le trajet de notre agent (ou de nos agents) qui est directement dues à la nature de notre graphe et aux caractéristiques du trajet. On sait que :

- les agents commencent et finissent leur trajet par des points d'entrée et de sortie, appartenant au réseau routier ;
- le passage du réseau routier au réseau piéton, et vice-versa, ne se réalise qu'en passant par les parkings.

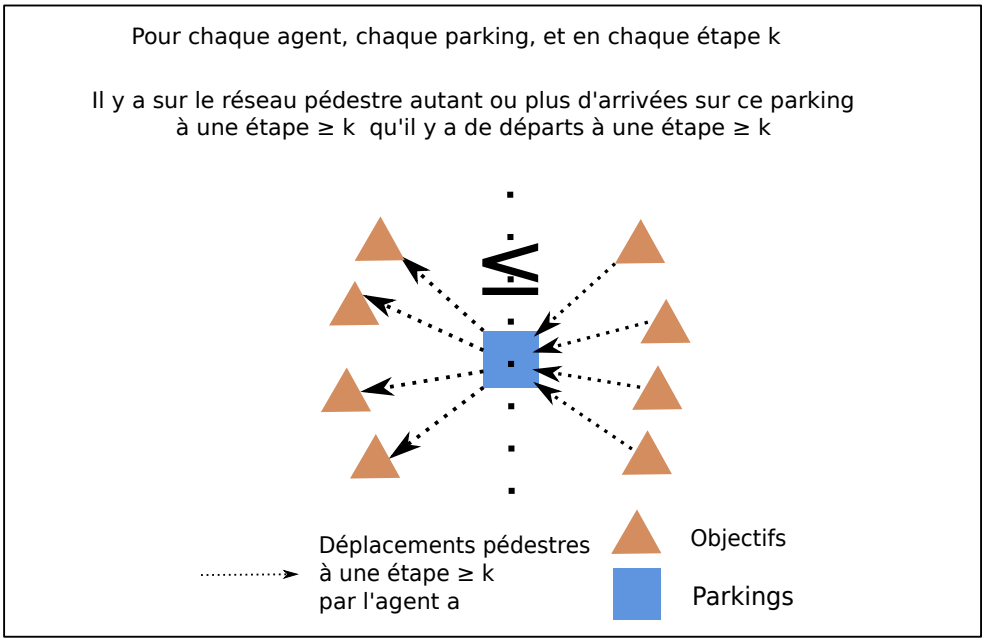


FIGURE 3.11 – Formulation (C_2) de la contrainte de retour au parking

Les agents vont donc commencer leur trajet par le réseau routier, pour finir sur ce même réseau, et vont par moments entrer sur le réseau piéton et en ressortir, éventuellement plusieurs fois. Si l'on considère l'ensemble des étapes d'un agent, il est contraint de rentrer exactement autant de fois sur le réseau pédestre qu'il en sortira. Aussi, les seuls points de contact entre notre réseau routier et notre réseau piéton sont les parkings. En découle, implicitement, l'égalité suivante :

$$\forall a \in A \sum_{p \in P} \sum_{d \in D} \sum_{k \in K^a} y_{p,d}^{a,k} = \sum_{p \in P} \sum_{d \in D} \sum_{k \in K^a} y_{d,p}^{a,k} \tag{3.9}$$

Nous allons justement nous servir de ces étapes d'entrée et de sortie du réseau piéton.

Définition 10. *Une* ***entrée dans le réseau pédestre*** *(ou piéton) est une étape où un agent s'en va d'un parking, à pied, en direction d'un objectif. De la même manière, une* ***sortie du réseau pédestre*** *est une étape de l'agent où ce dernier arrive sur un parking à pied depuis un objectif.*

Considérons un agent $a \in A$. Tout trajet possible de l'agent a admettra un nombre $i \in \mathbb{N}$ fini de départ d'un parking vers un objectif (entrée dans le réseau pédestre). De même grâce à l'équation 3.9 on sait avec certitude qu'il réalisera aussi cette même quantité i d'actions de rentrer

sur un parking depuis un objectif (sortie du réseau pédestre).

Notons, $k_j^+ \in K^a$, pour tout $j \in \mathbb{N}$ tel que $1 \leq j \leq i$, les i étapes de l'agent a, par ordre chronologique, où ce dernier réalise une entrée dans le réseau pédestre. De la même manière nommons $k_j^- \in K^a$, pour tout $j \in \mathbb{N}$, $1 \leq j \leq i$, les i étapes, par ordre chronologique, où l'agent a sort du réseau réseau pédestre (revient sur le réseau routier). Par ordre chronologique, ici, signifie simplement que pour tout r,$s \in \mathbb{N}$ tel que $1 \leq r < s \leq i$, on ait les inégalités $k_r^+ < k_s^+$ et $k_r^- < k_s^-$.

On note aussi, pour tout $j \in \mathbb{N}$ tel que $1 \leq j \leq i$, le parking utilisé par une étape d'entrée sur le réseau pédestre $p_j^+ \in P$ ou de sortie $p_j^- \in P$, j correspondant ici au numéro de l'entrée ou de la sortie du réseau pédestre concerné.

Notre agent commence son trajet sur le réseau routier. S'il réalise des étapes d'entrée et de sortie sur le réseau pédestre, la première de celle-ci sera donc l'entrée sur le réseau pédestre k_1^+, qui sera suivie plus tard par l'étape de sortie k_1^-. Les autres étapes d'entrées et de sorties suivront de la même manière. La dernière de ces étapes sera l'étape de sortie du réseau piéton k_i^-. Celle-ci se réalisera depuis un parking $p_i^- \in P$. Cette numérotation des étapes d'entrées et de sorties du réseau piéton ainsi que des parkings utilisés est illustré dans la figure 3.12.

Lors de l'entrée dans le réseau pédestre, à l'étape $k_i^+ \in K^a$ notre contrainte (C_2) doit être respectée. C'est à dire, qu'à partir de l'étape k_i^+, il doit y avoir parmi les étapes suivantes de notre agent, plus ou autant de retours sur le parking p_i^+, qui sera utilisé pour rentrer dans le réseau pédestre de cette étape, qu'il y aura d'étapes pour y rentrer. En considérant l'étape k_i^+ de l'agent et son parking utilisé p_i^+, selon l'inégalité de la contrainte $(C2)$:

$$\sum_{d \in D} \sum_{\substack{l \in K^a \\ l \geq k_i^+}} y_{p_i^+,d}^{a,l} \leq \sum_{d \in D} \sum_{\substack{l \in K^a \\ l \geq k_i^+}} y_{d,p_i^+}^{a,l}$$

Une entrée sur le réseau pédestre étant effectuée à l'étape k_i^+ , il doit y avoir une sortie du réseau pédestre à une étape supérieure sur ce même parking. Or, la seule étape de sortie de réseau pédestre ensuite est l'étape k_i^-. La seule possibilité donc qui respecte la contrainte (C_2) est que $p_i^- = p_i^+$.

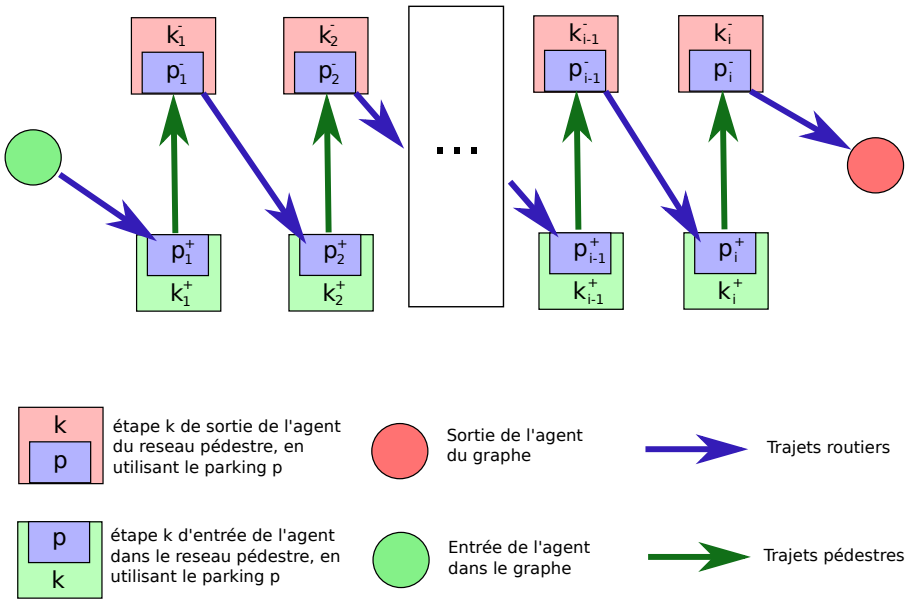


FIGURE 3.12 – Numérotation des étapes d'entrée/sortie du réseau pédestre

De la même manière, on pourrait réitérer la méthode pour déduire que le parking utilisé à l'étape k^+_{i-1} est le même que celui utilisé à l'étape k^-_{i-1}. Le parking utilisé durant l'étape k^+_{i-1} doit être utilisé plus tard pour une sortie du réseau pédestre, d'après l'équation :

$$\sum_{d\in D}\sum_{\substack{l\in K^a\\ l\geq k^+_{i-1}}} y^{a,l}_{p^+_{i-1},d} \leq \sum_{d\in D}\sum_{\substack{l\in K^a\\ l\geq k^+_{i-1}}} y^{a,l}_{d,p^+_{i-1}}$$

Il n'y a, à partir de l'étape k^+_{i-1} que deux étapes existantes de sorties du réseau pédestre k^-_{i-1} et k^-_i, et ils utilisent respectivement les parkings p^-_{i-1} et p^-_i. Il n'y a donc que l'un de ces deux parkings que pourrait utiliser l'étape k^+_{i-1}. Si l'étape k^-_{i-1} utilise le parking de p^-_i, il y aura donc maintenant à partir de l'étape k^+_{i-1} deux entrées effectuées par le parking p^-_i, et seulement deux étapes de sorties existantes : k^-_i qui utilise bien déjà p^-_i, et k^-_{i-1} qui sera donc contrainte pour que l'inégalité 3.9 soit respectée d'utiliser aussi p^-_i. Donc $p^+_{i-1} = p^-_i = p^-_{i-1}$, ce qui reviendrait au même résultat que l'unique deuxième hypothèse, si $p^+_{i-1} = p^-_{i-1}$. Donc, dans les 2 cas, $p^+_{i-1} = p^-_{i-1}$.

On peut remonter le fil de la même manière jusqu'aux étapes k^+_1 et k^-_1 pour arriver à la conclusion que les entrées et les sorties sur le réseau pédestre, se font à chaque fois l'une à la suite de l'autre, en utilisant le

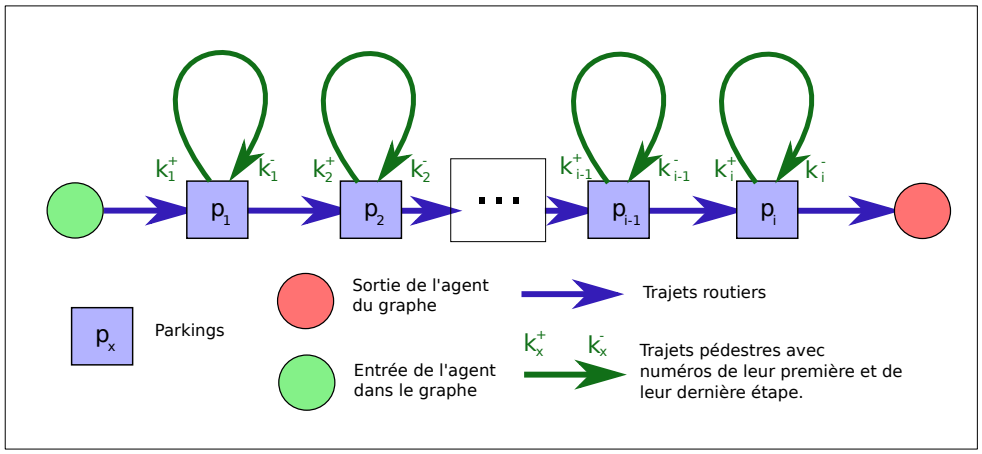


FIGURE 3.13 – Déduction du trajet sous Contrainte (C_2) d'après la précédente numérotation

même parking pour sortir du réseau que l'étape précédente par lequel l'agent y est rentré. C'est à dire que pour tout $j \in \mathbb{N}$, $1 \leq j \leq i$, on retrouve $p_j^+ = p_j^-$. Cela sans avoir pu entre les deux étapes k_j^+ et k_j^- utiliser le réseau routier (car aucune étape d'entrée ou de sortie du réseau piéton n'est entre k_j^+ et k_j^-). Ce qui respecte exactement notre contrainte (C_1). On considérera le parking $p_j = p_j^+ = p_j^-$ qui est le parking depuis lequel s'effectuera l'entrée sur le réseau piéton k_j^+ ainsi que sa sortie k_j^-, que l'on utilisera sur notre figure 3.13 afin de montrer que la contrainte (C_1) est bien respectée. La contrainte (C_1) et la contrainte (C_2) sont donc bien équivalentes.

□

Cette reformulation nous permet de linéariser notre contrainte du retour de parking sans avoir, comme dans la précédente méthode, à faire intervenir des variables représentant le trajet de la voiture. En faisant disparaître les variables x, la complexité en nombre de variables est grandement réduite et autorise une première expérimentation numérique de la qualité de la formulation proposée. La méthodologie de génération des instances ainsi que les résultats des tests effectués sont donnés dans la section 3.5, après avoir vu, dans la section suivante, une autre méthode de modélisation.

Notre deuxième méthode modélise donc le problème $(MA-TSP-$

PR) comme suit.

$$\begin{aligned} \text{Min} \quad & \sum_{(i,j)\in E}\sum_{a\in A}\sum_{k\in K^a} c_{ij}y_{ij}^{ak} \\ \text{sujet-à :} \quad & (3.1),(3.2),(3.3),(3.4),(3.6) \\ & \sum_{d\in D}\sum_{\substack{l\in K^a \\ l\geq k}} y_{p,d}^{a,l} \leq \sum_{d\in D}\sum_{\substack{l\in K^a \\ l\geq k}} y_{d,p}^{a,l} \quad \forall a\in A, p\in P, k\in K^a \qquad (3.10) \end{aligned}$$

Les contraintes (3.1),(3.2),(3.3),(3.4),(3.6) vérifient que les liaisons parcourues forment un chemin dans le graphe. La contrainte (3.10) est notre nouvelle contrainte de retour au parking, et remplace les précédentes contraintes liées au trajet de la voiture.

3.3.3 Modélisation en sous-trajets : variables décisionnelles f, n et l

La deuxième reformulation se base sur le constat suivant. Chaque trajet d'agent visitant des objectifs souhaités peut être décomposé en 4 types de sous-trajets élémentaires. Nous avions déjà pu le voir lorsque nous avons chercher à compter le nombre maximum d'étapes d'un agent sur la figure 3.9 de la section 3.3.1 :

- T_1. Partir de son entrée du système, jusqu'à un parking puis se rendre à pied à un objectif.
- T_2. Partir d'un objectif, pour aller a pied jusqu'à un autre objectif.
- T_3. Partir d'un objectif, récupérer sa voiture sur le parking où elle se trouve, aller en voiture sur un autre parking, puis se rendre à pied à un autre objectif.
- T_4. Partir d'un objectif, récupérer sa voiture sur le parking où elle se trouve et se rendre à la sortie du système.

Pour chaque agent $a \in A$, on peut alors définir trois types de variables de décision décrites ci-dessous.

• Les sous-trajets de type T_1 dits "First". Ils représentent le premier trajet effectué par un agent, de son entrée vers le premier objectif visité. Ils sont définis par :

$$f_{pd}^a = \begin{cases} 1 & \textit{si l'agent } a \textit{ va de son origine vers l'objectif } d \\ & \textit{en se garant sur le parking } p. \\ 0 & \textit{sinon} \end{cases}$$

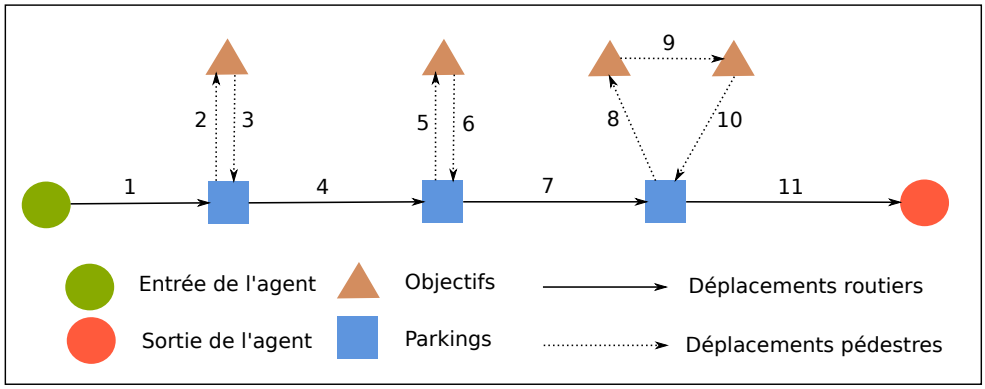


FIGURE 3.14 – Numérotation des $k \in K^a$ étapes dans le modèle par liaisons sans voiture.

• Les sous-trajets de type T_2 et T_3 dits "Next" Ils représentent les trajets effectués entre deux objectifs par un agent, qu'il soit avec changement de parking ou non. Ils sont définis par :

$$n^{ak}_{d_1 p_1 p_2 d_2} = \begin{cases} 1 & \textit{si l'agent } a \in A \textit{ va de l'objectif } d_1 \textit{ vers } d_2 \textit{ pour son } k^{eme} \\ & \textit{déplacement en utilisant le ou les parking(s) } p_1 \textit{ et } p_2 \\ 0 & \textit{sinon} \end{cases}$$

Il est important de voir ici que l'indice "k" se réfère au numéro d'étape des déplacements entre objectifs. A chaque fois que l'agent se déplace entre deux objectifs, cette étape de déplacement est numérotée. Comparativement au modèle précédent la numérotation proposée ici est une sorte "d'agrégation" de la précédente. Ce découpage se fait sans perte d'information pertinente pour l'optimisation. En effet, auparavant la numérotation des étapes relatives aux variables pouvait donner le trajet de la figure 3.14. Au lieu de considérer une à une les étapes, le découpage de la figure 3.15 en sous-trajets peut être fait.

Quatre types de sous-trajets correspondants aux cas T_1 T_2, T_3, T_4 peuvent être observés sur l'exemple. Les trajets du type T_2 et T_3 peuvent par extension se généraliser en un seul type. Il s'agit en effet pour l'agent d'aller d'un objectif à un autre en changeant (T_3) ou non (T_2) de parking. Car le fait de ne pas changer de parking induit que le déplacement de l'objectif origine à celui de destination s'est nécessairement fait à pied. Ce qui est bien le cas pour T_2.

La variable $n^{ak}_{d_1 p_1 p_2 d_2}$ permet donc d'interpréter deux cas de figure. Si $p_1 \neq p_2$ cela induit un changement de parking. Le déplacement de d_1 à

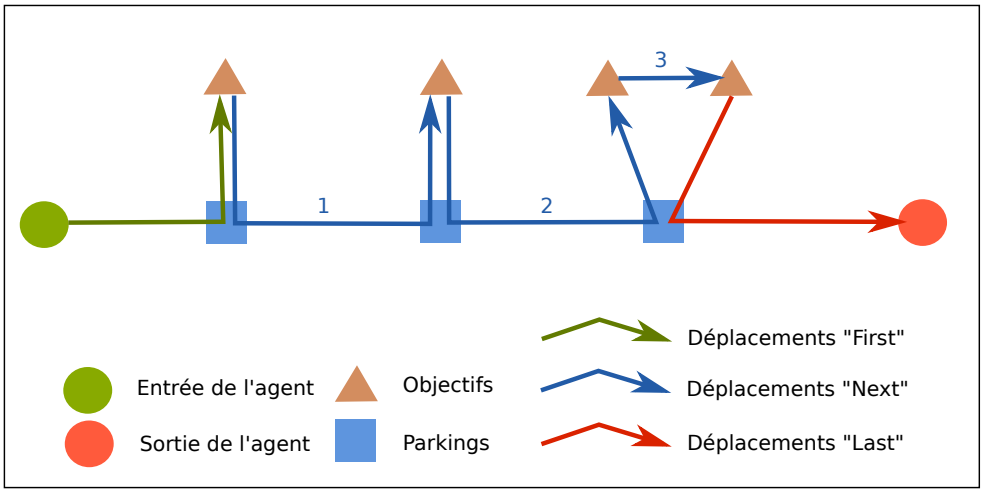


FIGURE 3.15 – Numérotation des k étapes "Next" dans le modèle en sous-trajet.

d_2 s'est fait en allant de d_1 au parking p_1, où se trouve la voiture. En l'utilisant pour se rendre au parking p_2. Puis de p_2 à rejoindre à pied d_2. Sinon $p_1 = p_2$, l'agent s'est rendu de d_1 à d_2 à pied, sans changer de parking, p_1 (et donc aussi p_2) étant le parking sur lequel se trouve son véhicule.

L'indice k représente la suite (arithmétique de raison 1) des numéros d'étapes de déplacement d'objectif en objectif comme indiqué dans l'illustration. Contrairement aux modèles par liaison avec ou sans véhicule, on sait de manière exacte le nombre d'étapes qui seront réellement effectuées par l'agent. En effet, il dépend maintenant du nombre de trajets entre objectifs. Le nombre d'objectifs d'un agent est fixe. Dans un contexte de minimisation du coût (l'agent n'aura jamais à passer deux fois par le même objectif), le nombre de trajets entre ces objectifs à effectuer est donc égal au nombre de ses objectifs moins 1. Il est donc défini par :

$$i.e\ 1 \leq k \leq |D^a| - 1$$

Nous redéfinissons K^a pour la numérotation des étapes n. Son rôle ici sera exactement le même que dans nos anciens modèles, aux différences que seul n nécessite d'être numéroté et que l'on connaît exactement le nombre d'étapes inter-objectif qui seront réalisées (contrairement au nombre de liaisons dans E, précédemment). K^a, pour $a \in A$, est donc maintenant l'ensemble des entiers naturels compris entre 1 et $|D^a| - 1$ inclus. Tout ce qui sera vu par la suite dans ce manuscrit utilisera la

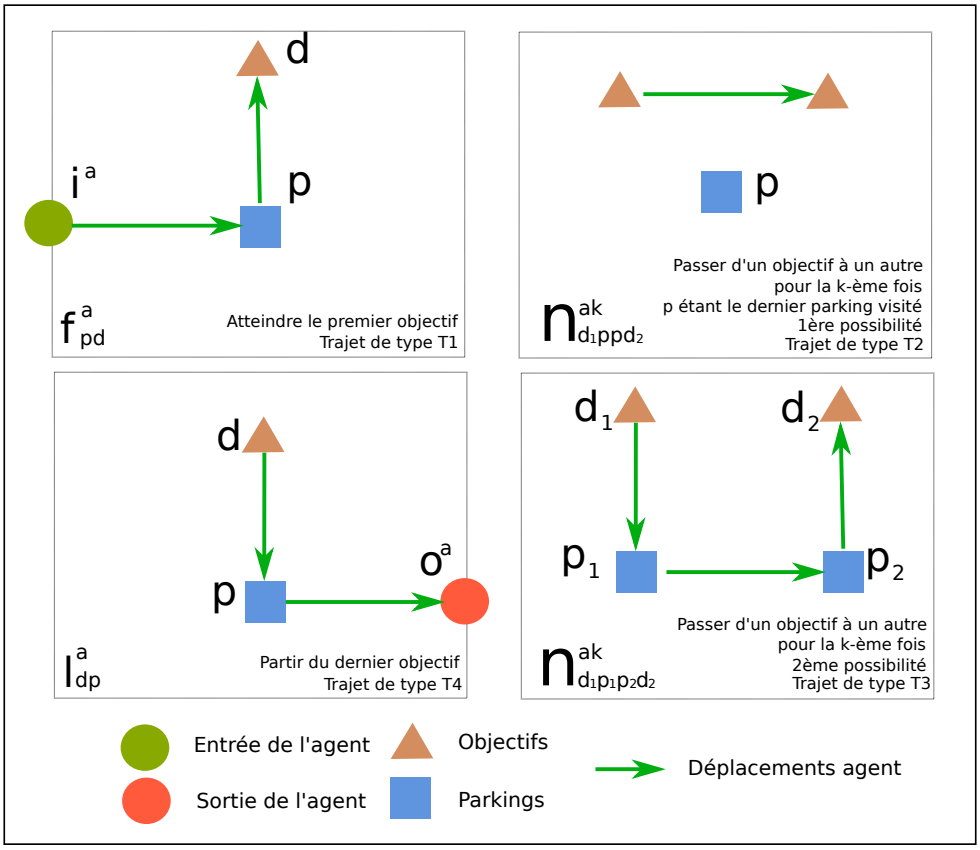


FIGURE 3.16 – Décomposition en sous-trajets élémentaires, pour chaque agent $a \in A$.

méthode en sous-trajet, il ne pourra donc pas y avoir confusion entre les deux différents K^a, la précédente définition ne sera plus utilisée.

$$\forall a \in A, \forall k \in K^a, 1 \leq k \leq |D^a| - 1$$

• Les sous-trajets de type T_4 que nous appellerons "Last". Ils représentent le dernier trajet effectué par un agent, partant de son dernier objectif en récupérant sont véhicule sur le dernier parking utilisé. Ils sont définis par :

$$l^a_{dp} = \begin{cases} 1 & \textit{si l'agent } a \in A \textit{ part du système depuis } d \\ & \textit{en reprenant sa voiture sur le parking } p \\ 0 & \textit{sinon} \end{cases}$$

La figure 3.16 illustre chaque type de sous-trajets, associé à leur variable décisionnelle respective. Comme l'on peut le remarquer, i^a et o^a sont absents des variables décisionnelles f et l, car ils sont directement déductibles de a.

Pour écrire les contraintes auxquelles est assujetti le problème, il est intéressant de distinguer le cas où l'agent ne visite qu'un objectif, du cas de celui où il en visite plus.

Ainsi si l'agent $a \in A$ n'a qu'un objectif $d \in D$, on a nécessairement :

$$f^a_{p,d} = l^a_{d,p}, \forall\ p \in P$$

En effet si a se gare en premier sur un parking p, il se rendra à pied à l'objectif d, reviendra à pied au parking duquel il quittera la zone commerciale. Ce cas correspond à un parking p pour lequel

$$f^a_{p,d} = l^a_{d,p} = 1$$

Pour tous les autres parkings sur lesquels a ne s'est pas rendu, on a

$$f^a_{p,d} = l^a_{d,p} = 0$$

Si le nombre d'objectifs est strictement supérieur à 2, il y a au moins une étape n et donc les étapes f et l ne doivent plus être liées entre-elles mais avec n. On a d'une part :

$$\begin{cases} f^a_{p_1,d_1} = \sum\limits_{p_2 \in P, d_2 \in D} n^{a,1}_{d_1,p_1,p_2,d_2} \\ l^a_{d_1,p_1} = \sum\limits_{p_2 \in P, d_1 \in D} n^{a,|D^a|-1}_{d_2,p_2,p_1,d_1} \end{cases} \forall\ a \in A, p_1 \in P, d_1 \in S$$

La première contrainte exprime que si l'agent s'est garé sur p_1 pour aller à l'objectif d_1 ($f^a_{p_1,d_1} = 1$) alors, à l'étape suivante (indice $k = 1$ des variables n) de son trajet, pour atteindre son second objectif d_2, il utilisera nécessairement ce même parking p_1 pour quitter d_1. Rappelons qu'il est possible que $p_1 = p_2$ ce qui signifierait que l'agent s'est rendu de d_1 à d_2 à pied. Dans ce cas là, l'agent ne sera pas passé par p_1, mais on considérera que ce dernier a été emprunté quand même, pour sortir de d_1 et pour se rendre à d_2. Bien évidement, le coût associé à ce sous-trajet utilisant deux fois ce parking p_1 sera le coût du trajet pédestre entre les deux objectifs. De façon analogue, si la sortie de la zone a été faite en quittant un parking p_1 atteint à pied d'un objectif d_1, alors la dernière étape de visite des objectifs (indicée $|D^a| - 1$ pour les variables n) a consisté à aller vers l'objectif d_1 en provenance d'un autre objectif d_2. Le voyage de d_2 à d_1 s'étant réalisé grâce aux parkings p_2 et p_1.

D'autre part, viendra se rajouter la contrainte de continuité entre les étapes n, s'il y en a plusieurs (donc si le nombre d'objectif est supérieur à 3, pour qu'il y ait au moins 2 liaisons inter-objectifs) :

$$\forall\ a \in A, \forall p_1 \in P, \forall d_1 \in D, \forall k \in \mathbb{N} / 1 \leq k \leq |D^a - 2|$$
$$\sum_{p_2 \in P, d_1 \in D} n^{a,k}_{d_2,p_2,p_1,d_1} == \sum_{p_2 \in P, d_2 \in D} n^{a,k+1}_{d_1,p_1,p_2,d_2}$$

La numérotation en "k" des étapes de n (déplacements objectif à objectif) empêche la formation de cycle.

De nouvelles constantes de données initiales surviennent, les coûts des sous-trajets, qui seront pré-calculés en fonction des 4 cas que l'on a pu voir. Elles ne sont pas indispensables, on pourrait réaliser notre modèle tout aussi bien en utilisant nos précédents coûts par liaison. Elles permettent surtout un peu plus de clarté. Ceci notamment pour ne pas avoir besoin de décomposer les n dans la fonction de minimisation en fonction de l'égalité ou non des deux parkings utilisés pour un sous-trajet inter-objectifs. Nous les déclarons ici :

- Soit c_{ipd} le coût des "premiers" sous-trajets, pour $i \in I$, $p \in P$ et $d \in D$. $c_{ipd} = c_{ip} + c_{pd}$.
- Soit c_{dpo} le coût des "derniers" sous-trajets, pour $o \in O$, $p \in P$ et $d \in D$. $c_{pdo} = c_{pd} + c_{do}$.
- Soit $c_{d_1p_1p_2d_2}$ le coût des sous-trajets inter-objectifs, pour p_1,$p_2 \in P$, d_1,$d_2 \in D$. Le coût sera calculé différemment selon si on a l'égalité $p_1 = p_2$. Si $p_1 = p_2$, $c_{d_1p_1p_2d_2} = c_{d_1d_2}$, sinon $c_{d_1p_1p_2d_2} = c_{d_1p_1} + c_{d_1d_2} + c_{d_2p_2}$.

Notre nouvelle fonction de minimisation se décomposera maintenant en 3 parties (f, n et l) comme suit.

$$\text{Min} \quad \sum_{a \in A} \sum_{d \in D} \sum_{p \in P} c_{i^a pd} * f^a_{pd}$$
$$+ \sum_{a \in A} \sum_{d_1,d_2 \in D} \sum_{p_1,p_2 \in P} \sum_{k \in K^a} c_{d_1p_1p_2d_2} * n^{ak}_{d_1p_1p_2d_2}$$
$$+ \sum_{a \in A} \sum_{d \in D} \sum_{p \in P} c_{dpo^a} * l^a_{dp}$$

Nous reprenons ensuite le même type de contraintes que précédemment pour assurer un chemin de l'entrée des agents jusqu'à la sortie en

utilisant les sous-trajets.

La contrainte (3.1) vérifiant la limite du nombre de flux par agent par étape à 1 se décompose maintenant en 3 contraintes. Ces nouvelles contraintes ne vont plus limiter mais forcer cette valeur de 1 (car l'on sait maintenant exactement le nombre d'étapes) :

$$\sum_{p\in P,\ d\in D} f^a_{pd} = 1 \qquad \forall\ a \in A \tag{3.11}$$

$$\sum_{p\in P,\ d\in D} l^a_{dp} = 1 \qquad \forall\ a \in A \tag{3.12}$$

$$\sum_{p,q\in P,\ d_1,d_2\in D} n^{ak}_{d_1p_1p_2d_2} = 1 \quad \forall\ a \in A,\ k \in \mathbb{N},\ k \in K^a \tag{3.13}$$

(3.11) : Il doit y avoir un "First" par agent.

(3.12) : Il doit y avoir un "Last" par agent.
(3.13) : Il doit y avoir un "Next" par "étape inter-objectif.agent" le nombre d'étapes inter-objectif dépendant du nombre de magasin que doit voir l'agent.

Les précédentes contraintes (3.2) et (3.3) demandant un flux sortant de l'entrée de l'agent ne sont plus nécessaires dans cette modélisation. Les entrées et sorties n'interviennent plus dans les variables décisionnelles. La contrainte (3.4) qui permettait à la fois continuité du chemin et empêchait les cycles externes au chemin de l'entrée à la sortie, va se décomposer elle en 4 contraintes, comme nous l'avons vu précédemment.

$$f^a_{pd} = l^a_{dp} \qquad \begin{array}{l}\forall a \in A,\ K^a = \{\emptyset\},\\ d \in D,\ p \in P\end{array} \tag{3.14}$$

$$f^a_{p_1d_1} = \sum_{p_2\in P}\sum_{d_2\in D} n^{a,1}_{d_1p_1p_2d_2} \qquad \begin{array}{l}\forall a \in A,\ 1 \in K^a,\\ d_1 \in D,\ p_1 \in P\end{array} \tag{3.15}$$

$$\sum_{p_2\in P}\sum_{d_2\in D} n^{ak}_{d_2p_2p_1d_1} = \sum_{p_2\in P}\sum_{d_2\in D} n^{a,k+1}_{d_1p_1p_2d_2} \qquad \begin{array}{l}\forall a \in A,\ k,k+1 \in K^a,\\ d_1 \in D,\ p_1 \in P\end{array} \tag{3.16}$$

$$\sum_{p_2\in P}\sum_{d_2\in D} n^{a(|D^a|-1)}_{d_2p_2p_1d_1} = l^a_{d_1p_1} \qquad \begin{array}{l}\forall a \in A,\\ d_1 \in D,\ p_1 \in P\end{array} \tag{3.17}$$

(3.14) concernera le cas où l'agent n'a qu'un seul objectif, vérifiant la continuité entre f et l.

(3.15),(3.16),(3.17) concernera le cas où l'agent a plus d'un objectif. Dans ce cas il doit y avoir continuité de flux entre f et le premier n (3.15), entre le dernier n et l (3.16), et entre chaque étape de n (3.17).

Pour finir il nous reste la contrainte demandant aux agents d'atteindre leurs objectifs. les objectifs sont obligatoirement atteints soit par un f soit par un n, il faudra donc vérifier pour chaque objectif de l'agent que l'un de ces deux types de sous-trajet le concernant l'atteigne

$$\sum_{p\in P} f^a_{p,d_1} + \sum_{p_1,p_2\in P, d_2\in D, k\in K^a} n^{ak}_{d_2p_2p_1d_1} = 1 \quad \forall a \in A,\, d_1 \in D^a \quad (3.18)$$

Reprenant les contraintes vu précédemment, notre modèle en sous-trajet est :

$$\begin{aligned}
\text{Min} \quad & \sum_{a\in A}\sum_{d\in D}\sum_{p\in P} c_{i^a pd} * f^a_{pd} \\
& + \sum_{a\in A}\sum_{d_1,d_2\in D}\sum_{p_1,p_2\in P}\sum_{k\in K^a} c_{d_1p_1p_2d_2} * n^{ak}_{d_1p_1p_2d_2} \\
& + \sum_{a\in A}\sum_{d\in D}\sum_{p\in P} c_{dpo^a} * l^a_{dp} \\
\text{sujet-à :} \quad & (3.11),(3.12),(3.13),(3.14), \\
& (3.15),(3.16),(3.17),(3.18)
\end{aligned}$$

3.3.4 Expérimentations numériques

Nous avons vu dans les sections précédentes 3 méthodes de modélisation d'un même problème, le MA-TSP-PR. Ces 3 modélisations sont exactes, par rapport à notre approche de la problématique. C'est à dire que toutes trois trouveront la solution optimale pour minimiser le même coût. Ce n'est donc bien évidement pas sur leur performance de minimisation que nous allons les comparer mais en termes de temps d'exécution en mobilisant CPLEX pour résoudre les modèles codés en OPL (Open Programming Langage). Tous les tests que nous allons voir ont été réalisés sur un ordinateur quad-core de 3.2 GHz de fréquence disposant de 8 Go de mémoire vive. Le but ne sera pas vraiment de savoir quel est le temps d'exécution, mais de comparer les 3 modèles pour savoir lequel sera le plus rapide.

Hormis quelques exceptions qui seront précisées, les tests ont tous été

réalisés de la manière suivante :

1. Génération d'un nombre prédéfini de situations aléatoires appartenant au même type / à la même complexité.
2. Écriture des fichiers de données de ces situations pour les 3 modélisations.
3. Résolutions des 3 modélisations pour chaque situation. On garde les temps de résolution et on vérifie la cohérence des résultats (même valeur optimale).
4. On modifie ces situations pour en changer un paramètre. Par exemple, on ajoute 1 ou plusieurs parkings.
5. On résout les modélisations avec les nouvelles données générées. On garde les temps de résolution, on vérifie la cohérence, et on revient à l'étape 4 jusqu'à l'arrêt des itérations.

La condition d'arrêt (le nombre d'itérations) sera fixée suivant les cas pour différentes raisons :

- Le nombre d'itérations effectuées est suffisant pour avoir un visuel correct de l'impact ou non du facteur étudié.
- Les itérations suivantes pourraient engendrer une saturation en mémoire vive, ce qui fausserait nos estimations en temps.
- On évitera, sauf quand cela pourrait avoir de l'intérêt, de monter à plus de 100 secondes pour la résolution d'un seul problème. Cela évite d'une part le risque de saturation en mémoire d'un test isolé. D'autre part chaque figure résulte d'un très grand nombre de résolutions.

Plus de précision quant à la manière de générer les tests aléatoires sont présentes en annexe B.2. Des résultats détaillés des tests en question concernant cette section sont présentés en annexe C.1.

Comparaison des 3 modèles

Les 3 modélisations étudiées sont équivalentes en terme de solutions trouvées, c'est donc sur leur temps de résolution qu'elles sont comparées. Pour les problèmes MA-TSP-PR (sans optimisation de localisation) en règle général la modélisation prenant en considération la voiture des agents reste la plus lente, et la modélisation en sous-trajet la plus performante. Exception faite du cas où le graphe soit très grand pour une petite complexité du problème. Cette exception concerne par exemple le cas où l'on ai beaucoup d'objectifs sur le graphe (objectifs localisés) sans

augmenté le nombre d'objectifs par agent et pour un nombre faible de parking. En MA-TSP-PR les agents sont indépendants les uns des autres. Si on fixe le nombre de leur objectifs (objectifs ciblés par les agents) et que l'on augmente le nombre total d'objectifs localisés sur le graphe, le problème de chacun des agents ne sera que légèrement augmenté par le graphe qui contiendra de plus en plus d'objectifs que l'agent ne souhaite pas voir. Dans ce cas là, on notera une baisse de performance du modèle en sous-trajet, non pas parce que le problème sera trop complexe à résoudre, mais parce que le nombre de variables pour ce modèle-ci augmentera plus vite.

Augmentation de complexité

La complexité de nos modèles est difficilement quantifiable. La complexité en nombre de variables ou même en nombre de contraintes ne sont pas vraiment des points de comparaisons fiables pour jauger l'évolution de la complexité d'une résolution d'un modèle d'optimisation linéaire. De plus en utilisant CPLEX qui réalise beaucoup de simplification qu'on ne cherchera pas ici à étudier. La mesure la plus juste est d'évaluer cette montée en complexité directement par des tests numériques. Plusieurs constat ont pu être effectué :

- Pour un même type de test (même nombre d'agents, d'objectifs par agent, de parkings, d'objectifs) le temps de résolution peut être très variable. Il est possible de voir un test ayant le même nombre de chaque variable qu'un autre test obtenir des temps de résolution dix fois supérieur.
- L'augmentation du temps de résolution moyen en fonction du nombre d'agents ou d'objectifs est plutôt faible et semble linéaire.
- L'augmentation du temps de résolution moyen en fonction du nombre de parkings est sensiblement plus fort, l'augmentation semble exponentielle.
- L'augmentation du temps de résolution en fonction du nombre d'objectifs par agent est la plus forte semblant être de l'ordre du factoriel. Ceci étant, dans notre contexte d'étude, un nombre d'objectifs par agent de 2 ou 3 (voir 4) est suffisant. Les cas d'agents souhaitant atteindre plus de 3 commerces en parcourant une zone commerciale reste très minoritaire.

Conclusions

Il semble se dégager que le modèle en sous-trajet est le plus performant pour l'optimisation d'un problème MA-TSP-PR, même si les problèmes MA-TSP-PR n'ont pas vraiment lieu d'être sans autres leviers d'optimisation ou contraintes. La modélisation en liaisons avec voiture par contre est toujours moins performante que celle sans voiture, raison pour laquelle nous l'éliminons déjà à ce stade. Il est difficile d'établir de règle précise quand à l'augmentation de temps de résolution en fonction des quantités des différents éléments rentrant en jeu dans le problème. Néanmoins il reste possible de dégager quelques règles approximatives d'évolution du temps de résolution en fonction de l'augmentation d'un paramètre. Par exemple on peut voir que le nombre d'agent, principale donnée qui risque d'atteindre une quantité très importante par rapport au contexte, engendre une évolution du temps de résolution à peu près linéaire. Ce qui est primordial si l'on souhaite pouvoir atteindre des quantités d'agents plus réalistes. Le seule paramètre qui posera très vite problème s'il amplifie trop est le nombre d'objectif par agent, qui lui, au contraire, n'a pas pour vocation d'atteindre des valeurs trop haute, dans notre contexte.

Dans la section suivante seront ajouté les principaux leviers à optimiser, à savoir les localisations des objectifs ainsi que des parkings. De nouveaux tests sur ces problèmes plus complets seront réalisés.

3.4 Optimisation des localisations et capacités

3.4.1 Localisations des objectifs

Introduction

Les modèles vus jusqu'à présent nous permettent de connaître le coût minimal de déplacement d'agents visitant des objectifs sur un graphe considérant les parkings (et les déplacements à pied et en voiture). Or notre objectif n'est pas tant d'estimer ce coût minimal que de pouvoir agir sur différents points de notre zone d'étude afin de minimiser encore plus ce coût. Un de ces points est la localisation des objectifs. Dans cette partie donc, nous chercherons à adapter notre modèle afin de pouvoir répondre à la question suivante : parmi différentes localisations possibles, quelle est l'affectation des objectifs optimale (ou une des affectations optimales) qui

permet à nos agents d'atteindre leurs objectifs à moindre coût total ?

Notre problème ne se place donc plus maintenant au niveau des agents mais au niveau du gestionnaire de la zone étudiée, qui cherche à l'améliorer en jouant sur la localisation des objectifs. Ce nouveau point de vue va apporter plusieurs changements :

- Les agents connaissent toujours la liste des objectifs qu'ils ont envie de voir, mais ces objectifs ne sont plus localisés (l'ensemble D n'appartient donc plus au graphe G).
- Il existe des localisations d'objectifs, qui elles, remplacent les précédents objectifs sur le graphe G.
- Chaque objectif de D admet une liste d'affectations possibles parmi les localisations, pouvant différer pour chaque objectif.
- D'un point de vue décisionnel, chaque objectif devra être affecté à une et une seule de ses localisations possibles.
- De même, chaque localisation ne pourra être affectée que par un seul objectif au maximum.

Soit W l'ensemble des localisations possibles.

Soit U l'ensemble des affectations permises. Tout $u \in U$ permet l'affectation d'un objectif noté d^u dans une localisation notée $w^u.d^u \in D$ et $w^u \in W$.

Soit z^u, la variable décisionnelle d'affectation :

$$z_u = \begin{cases} 1 & \textit{si le choix de localisation de l'objectif } d^u \\ & \textit{se porte sur la localisation } w^u \\ 0 & \textit{sinon} \end{cases}$$

Il est à noter que la fonction à minimiser ne changera pas.

Bien évidement il va y avoir des impacts sur les contraintes des modèles :

- Affectations d'objectifs : nouvelles contraintes à ajouter pour permettre l'affectation des éléments de D sur les localisations de W.
- Atteindre les objectifs : les agents ne doivent plus atteindre, sur le graphe, les "objectifs" de leur liste mais à la place les localisations affectées par les objectifs de leur liste.
- Autres contraintes liées aux localisations : toute contrainte qui s'appuyait sur les objectifs en tant qu'élément du graphe G s'appuiera maintenant sur les localisations d'objectifs à la place.

Modélisation

L'intégration de ce nouveau levier d'optimisation peut s'effectuer sur les 3 modélisations linéaires précédemment abordées (en liaisons avec/-sans voiture, et en sous-trajets). Le principe restant similaire pour les 3 modèles, seul le modèle mathématique en sous-trajet prenant en compte ce nouveau levier est présenté. Voici les changements apportés au modèle en sous-trajets.

• Affectations d'objectifs.

Se rajoutent à notre modèle les contraintes suivantes :

$$\sum_{\substack{u \in U \\ d^u = d}} z_u = 1 \quad \forall\, d \in D \tag{3.19}$$

$$\sum_{\substack{u \in U \\ w^u = w}} z_u \leq 1 \quad \forall\, w \in W \tag{3.20}$$

(3.19) demande d'avoir exactement 1 localisation affectée pour chaque objectif, tandis que (3.20) permet à chaque localisation d'être affectée au maximum par 1 objectif. On admet qu'il puisse y avoir plus de localisations possibles que d'objectifs. Si c'est le cas, certaines localisations ne seront pas affectées dans les solutions.

• Atteindre les objectifs.

La contrainte (3.18) demandant d'atteindre les objectifs qui appartenaient au graphe est remplacée par la suivante :

$$\begin{aligned} \sum_{p \in P} f^a_{pw_1} + \sum_{p_1, p_2 \in P} \sum_{w_2 \in W} \sum_{k \in K^a} n^{ak}_{w_2 p_1 p_2 w_1} & \\ \geq \sum_{\substack{u \in U \\ w_1 = w^u \\ d^u \in D^a}} z_u & \qquad \forall a \in A,\, w_1 \in W \end{aligned} \tag{3.21}$$

(3.19) demande à toute localisation affectée par un objectif de l'agent $a \in A$ que ce dernier l'atteigne au moins une fois (que ce soit par son premier déplacement f ou un déplacement inter-objectif n).

• Autres contraintes liées aux localisations.

Chacune des autres contraintes (3.11) à (3.17) vues dans notre modèle en sous-trajet initial, devra porter cette fois-ci sur les localisations de l'ensemble W au lieu de l'ensemble D.

$$\sum_{p\in P}\sum_{w\in W} f^a_{pw} = 1 \qquad \forall\ a \in A \tag{3.22}$$

$$\sum_{p\in P}\sum_{w\in W} l^a_{wp} = 1 \qquad \forall\ a \in A \tag{3.23}$$

$$\sum_{p_1,p_2\in P}\sum_{w_1,w_2\in W} n^{ak}_{w_1p_1p_2w_2} = 1 \qquad \forall\ a \in A,\ k \in K^a \tag{3.24}$$

$$f^a_{pw} = l^a_{wp} \qquad \begin{array}{l}\forall a \in A\ /\ K^a = \{\emptyset\},\\ w \in W,\ p \in P\end{array} \tag{3.25}$$

$$f^a_{p_1w_1} = \sum_{p_2\in P}\sum_{w_2\in U} n^{a,1}_{w_1p_1p_2w_2} \qquad \begin{array}{l}\forall a \in A\ /\ 1 \in K^a,\\ w_1 \in W,\ p_1 \in P\end{array} \tag{3.26}$$

$$\sum_{p_2\in P}\sum_{w_2\in W} n^{ak}_{w_2p_2p_1w_1} = \sum_{p_2\in P}\sum_{w_2\in W} n^{a,k+1}_{w_1p_1p_2w_2} \qquad \begin{array}{l}\forall a \in A,\ k,(k-1) \in K^a,\\ w_1 \in W, p_1 \in P\end{array} \tag{3.27}$$

$$\sum_{p_2\in P}\sum_{w_2\in W} n^{ak}_{w_2p_2p_1w_1} = l^a_{w_1p_1} \qquad \begin{array}{l}\forall a \in A, w_1 \in W, p_1 \in P,\\ k \in K(a),\ k+1 \notin K(a)\end{array} \tag{3.28}$$

Le problème $(MA-TSP-PR-DL)$ se modélise comme suit :

$$\begin{aligned}\text{Min} \quad & \sum_{a\in A}\sum_{w\in W}\sum_{p\in P} c_{i^apw} * f^a_{pw}\\ + \quad & \sum_{a\in A}\sum_{w_1,w_2\in W}\sum_{p_1,p_2\in P}\sum_{k\in K^a} c_{w_1p_1p_2w_2} * n^{ak}_{w_1p_1p_2w_2}\\ + \quad & \sum_{a\in A}\sum_{w\in W}\sum_{p\in P} c_{wpo^a} * l^a_{wp}\end{aligned}$$

sujet-à :

$$(3.19),(3.20),$$
$$(3.21),$$
$$(3.22),(3.23),(3.24),(3.25),(3.26),(3.27),(3.28).$$

Les contraintes (3.19) et (3.20) sont de nouvelles contraintes permettant l'affectation des objectifs de D dans les localisations de W.

(3.21) est la contrainte demandant aux agents d'atteindre la locali-

sation de chacun de leurs objectifs,correspond à la contrainte (3.18) du précédent modèle sans relocalisation.

Enfin, (3.22) à (3.28) sont de simples ré-écritures des contraintes (3.11) à (3.17) où l'on utilise l'ensemble des localisations W au lieu de l'ensemble des objectifs D qui n'appartient plus au graphe G.

3.4.2 Localisations et capacités des parkings

Introduction

Nous avons vu comment améliorer notre modèle pour lui permettre d'agir sur la localisation des objectifs. Nous souhaitons de la même manière pouvoir agir au niveau des parkings. Néanmoins, la méthode qui est utilisée sur ces deniers n'est pas comparable à la précédente. Ceci en particulier car les parkings ne sont sur le graphe que des outils permettant aux agents de rentrer dans le réseau piéton, et non des objectifs. Un agent ne vise jamais une entité parking pour ce qu'elle est, mais simplement l'utilise parce que sa position sur le réseau complet lui est favorable. Au contraire de ce qui a été fait pour la localisation des objectifs, il n'y a pas besoin d'entité "parking non-localisé" à devoir affecter sur des "localisations de parking".

Nous souhaitons pouvoir agir de 2 manières sur les parkings :

- Limiter le nombre existant de parkings. Cet enjeu est un peu comparable à notre précédente affectation d'objectif. Ce point se résume à pouvoir répondre à la question suivante : parmi un certain nombre de localisations disponibles pour les parkings, où devrais-je localiser un nombre limité (et inférieur à ce nombre de localisations) de parkings de façon optimale afin de réduire le coût global de déplacement des agents.
- Limiter la capacité des parkings. Il s'agira non pas d'une optimisation supplémentaire, mais seulement de rajouter une contrainte afin d'éviter une sur-utilisation de certains parkings, au-delà de leur capacité d'accueil.

Limiter le nombre existant de parkings

Pour ce premier point, il n'y aura pas d'entité parking non-localisée à devoir affecter sur diverses localisations. Ceci car les parkings ne sont pas des buts en soit mais simplement des outils pour les agents. On parle donc ici à la place d'existence de parkings. Nous considérons la présence

de tous les parkings possibles (toutes les localisations) au niveau de notre graphe. Une variable artificielle est intégrée au modèle afin de mesurer si un parking est effectivement utilisé ou non. Nous définissons l'utilisation d'un parking comme étant le fait qu'un agent l'atteigne. Si un parking a été utilisé par au moins un agent, nous considérons qu'il "existe".

Soit $s^p, \forall p \in P$ une variable artificielle booléenne représentant l'existence d'un parking (si ce dernier a été utilisé par un agent), définie par les contraintes suivantes :

$$s^p \geq f^a_{pw} \qquad \forall p \in P, w \in W, a \in A \quad (3.29)$$

$$s^{p_1} \geq n^{a,k}_{w_1 p_2 p_1 w_2} \qquad \begin{array}{l} \forall p_1,p_2 \in P, w_1,w_2 \in W, \\ a \in A, k \in K^a \end{array} \quad (3.30)$$

$$\begin{array}{c} s^{p_1} \leq \sum\limits_{a \in A} \big(\sum\limits_{w \in W} f^a_{p_1 w} \\ + \sum\limits_{w_1,w_2 \in W} \sum\limits_{p_2 \in P} \sum\limits_{k \in K^a} n^{a,k}_{w_1 p_2 p_1 w_2} \big) \\ + S^{p_1} \end{array} \qquad \forall p_1 \in P \quad (3.31)$$

La contrainte (3.29) force s^p à prendre la valeur de 1 si ce dernier est atteint lors du premier sous-trajet d'un agent.
La contrainte (3.30) force s^p à prendre la valeur de 1 si ce dernier est utilisé en deuxième lors d'un sous-trajet inter-objectif d'un agent.
Enfin, si aucun sous-trajet de début ou inter-objectifs d'agent n'a fixé s^p à 1, on lui demande par la contrainte (3.31) de prendre la valeur de 0. On remarquera une variable qui n'est pas encore définie pour le moment sur cette dernière contrainte. Il s'agit de S^{p_1}. Nous allons la définir après avoir expliqué son utilité.

Dans un contexte réel, on peut vouloir "contraindre" un parking à exister. Il est assez facile d'imaginer la situation où l'on souhaite construire plusieurs nouveaux parkings et chercher à connaître leurs emplacements optimaux, mais sans avoir à toucher aux parkings déjà existants, qu'ils soient finalement utilisés ou non par les agents durant notre optimisation. C'est à cela que sert la donnée d'entrée binaire S^p définie pour tout $p \in P$. De manière formelle, si S^p vaut 1, s^p doit prendre la valeur de 1, quelle que soit l'utilisation qui en sera faite par les agents. Au niveau des contraintes, l'ajout de S^p sur la contrainte (3.31) permet justement que la valeur s^p ne soit pas forcée à 0 en cas de non-utilisation par les agents si S^p vaut 1. Il faut bien évidement rajouter à cela une contrainte

supplémentaire demandant, au contraire, à s^p de prendre la valeur de 1 si S^p vaut 1 :

$$s^p \geq S^p \quad \forall p \in P, \tag{3.32}$$

Nous avons donc maintenant parfaitement contraint notre variable artificielle s^p représentant l'existence d'un parking, si celui-ci est utilisé par un agent. Ceci tout en permettant, par le biais d'une donnée d'entrée S^p, à forcer son existence qu'il soit utilisé ou non. Il ne nous reste plus alors qu'à définir une donnée d'entrée qui représentera la limitation maximale du nombre de parkings existants et contraindre les variables s^p à respecter ce nombre limite d'existence.

Soit Ω la limite du nombre maximal de parkings "existants" autorisés. La contrainte demandant aux s^p de respecter cette limite est la suivante :

$$\sum_{p \in P} s^p \leq \Omega \tag{3.33}$$

Limiter la capacité des parkings

Outre l'existence ou non d'un parking, il peut aussi être très important de pouvoir limiter le nombre de fois qu'il sera utilisé. La méthode est similaire à la précédente, mais plus simple. Nous retrouvons de nouveau une variable artificielle, qui compte cette fois-ci le nombre de fois qu'un parking est utilisé. Il suffit ensuite de contraindre ce nombre d'utilisations à ne pas dépasser une valeur fixée.

Soit t^p une variable artificielle représentant le nombre (entier) d'utilisation d'un parking, sa valeur est fixée par la contrainte suivante :

$$\begin{aligned} t^{p_1} = \sum_{a \in A} \Big(\sum_{w \in W} f^a_{p_1 w} \\ + \sum_{w_1, w_2 \in W} \sum_{p_2 \in P} \sum_{k \in K^a} n^{a,k}_{w_1 p_2 p_1 w_2} \Big) \end{aligned} \quad \forall p_1 \in P \tag{3.34}$$

Soit T^p la donnée d'entrée qui fixera la capacité maximale d'utilisation d'un parking $p \in P$ (nombre entier). La contrainte qui demande que le nombre d'utilisations d'un parking ne soit jamais supérieur à sa limite

est la suivante :

$$t^p \leq T^p \quad \forall p \in P \tag{3.35}$$

Modélisation

Le problème $(MA-TSP-PRCL-DL)$ se modélise comme suit.

$$\begin{aligned}
\text{Min} \quad & \sum_{a\in A}\sum_{w\in W}\sum_{p\in P} c_{i^a pw} * f^a_{pw} \\
+ \quad & \sum_{a\in A}\sum_{w_1,w_2\in W}\sum_{p_1,p_2\in P}\sum_{k\in K^a} c_{w_1p_1p_2w_2} * n^{ak}_{w_1p_1p_2w_2} \\
+ \quad & \sum_{a\in A}\sum_{w\in W}\sum_{p\in P} c_{wpo^a} * l^a_{wp}
\end{aligned}$$

sujet-à :

(3.19) à (3.28),

(3.29) à (3.33),

(3.34) et (3.35)

Pour cette ajout, aucune modification sur les contraintes du (MA-TSP-PR-DL), c'est à dire les contraintes (3.19 à 3.28), ne sont nécessaires.

Les contraintes (3.29) à (3.33) permettent de fixer un nombre limite de parkings existants, ainsi que de forcer certains parkings à faire partis des existants. On peut retirer ce groupe de contraintes s'il n'y a pas ce choix de parking à réaliser (si le problème est un $MA-TSP-PRC-DL$).

Les contraintes (3.34) et (3.35) permettent la limitation de capacité sur l'utilisation d'un parking. On peut retirer ce groupe de contraintes s'il n'y a pas de capacité de parking à fixer (si le problème est un $MA-TSP-PRL-DL$).

3.4.3 Expérimentations numériques

Dans ces expérimentations numériques nous cherchons à avoir 3 informations :

- Quel modèle entre "Sans voiture" ou "en sous-trajets" est le plus performant en terme de temps de résolution lorsque l'on intègre

les possibilités de relocalisation d'objectifs ?
- Comment augmente la complexité (temps de résolution) lorsque l'on intègre les possibilités de relocalisation d'objectifs ?
- Comment augmente la complexité (temps de résolution) lorsque l'on augmente le nombre d'agents dans une situation ou tous les objectifs sont entièrement relocalisables (chaque objectif peut affecter toutes les localisations) ?

Comme précédemment, les résultats des tests dans une version plus détaillée sont présentés en annexe C.2.

Comparaison des 2 modèles (en liaison sans voiture et en sous-trajet)

Cette fois-ci, avec l'ajout du levier de localisation des objectifs il ne fait plus de doute que le modèle en sous-trajet se détache très significativement. Le modèle avec liaisons sans voiture est abandonné suite à ces tests, considéré comme beaucoup moins performant en temps de résolution.

Augmentation de complexité

On retrouve à peu près les mêmes constats (sauf pour l'augmentation du nombre d'objectifs) que pour les tests précédents, avec de nouveaux résultats liés aux leviers rajoutés :

- L'augmentation du temps de résolution en fonction du nombre d'agents est plutôt faible et semble linéaire.
- L'augmentation du temps de résolution en fonction du nombre de parkings ou de localisation d'objectifs (sans augmenter le nombre d'objectif à localiser) est sensiblement plus fort, l'augmentation semble exponentielle.
- L'augmentation du temps de résolution en fonction du nombre d'objectifs par agent est la plus forte semblant être de l'ordre du factoriel. Ceci étant, dans notre contexte d'étude, un nombre d'objectif par agent de 2 ou 3 (voir 4) est suffisant. Les cas d'agents souhaitant atteindre plus de 3 commerces en parcourant une zone commerciale reste très minoritaire.
- Si l'on augmente simultanément le nombre de localisations d'objectif et d'objectifs, en affectations possibles complètes (chaque objectif pouvant affecter chaque localisation), l'augmentation de temps de résolution semblent être de l'ordre du factoriel.

- Si on limite le nombre de parkings utilisable (MA-TSP-PRL-DL) le temps de résolution est augmenter de manière assez importante. Une augmentation du temps de résolution entre $x2$ et $x100$ a pu être constatée pour le rajout de ce levier. Il est à noter que les cas les plus longs à résoudre en MA-TSP-PRL-DL sont ceux pour lesquels la limite en nombre de parkings est très faible, le pire cas étant pour 2 parkings.

Conclusions

Seul la modélisation en sous-trajet sera conservée car la plus efficace en temps de résolution. Les augmentations de complexité restent cependant importantes même avec cette modélisation et pour un nombre de noeuds relativement faible. Même s'il n'est pas possible d'établir de règles fixes quant à cette augmentation de temps de résolution il reste assez évident aux vues des résultats que pour des jeux de données proche de celles d'une zone commerciale les résolutions ne seraient même pas envisageables.

Pour cette raison, dans la section suivante sont présentées deux simplifications exactes réalisables pour la modélisation de nos problèmes. Une de ces simplifications par exemple est de pré-traiter les données d'entrée afin d'éliminer celles qui peuvent facilement être considérées comme non-pertinentes. En effet, le principal défaut de la modélisation en sous-trajet reste l'importante quantité de sous-trajet, en particulier ceux inter-commerces, qui sont engendrés par l'augmentation du nombre de noeuds du graphe. Des heuristiques simples y sont aussi abordée, afin de voir s'il est possible par un calcul très rapide de trouver des solutions performantes (même si non-optimale).

Chapitre 4

Simplifications et Heuristiques

Sommaire

4.1 Simplifications par pré-traitement et décomposition du problème

4.1.1 Introduction

La modélisation en sous-trajet est, de loin, la plus performante en terme de temps d'exécution pour résoudre nos différents problèmes de type MA-TSP-PR. Mais pour des zones d'études assez complexes, de l'ordre par exemple d'un centre commercial, le temps de résolution peut s'avérer beaucoup trop élevé, au plus le nombre d'affectations possibles d'objectifs sur localisation est grand. Ceci étant, des simplifications supplémentaires sont encore possibles.

Les deux simplifications qui suivent, assez liées l'une à l'autre d'ailleurs, sont présentées ici sur un problème MA-TSP-PR-DL (Multi-Agent Travelling Salesman Problem with Parking Return and Destination Location). Elles pourraient aussi être ré-adaptables en conservant exactement la même méthodologie, afin d'intégrer aussi la localisation de parking. Plus de précision à ce sujet est expliqué indépendamment dans les sections dédiées à chacune de ces méthodes.

4.1.2 Simplification des données d'entrées : pré-traitement des sous-trajets pertinents

Le principe de cette méthode de simplification sera un pré-traitement sur nos sous-trajets afin de déterminer, avant optimisation, ceux qui auront une chance de faire partie des trajets de nos agents, et donc surtout, d'éliminer tous les sous-trajets non pertinents. Pour cela, nous allons considérer plusieurs sous-problèmes de taille très réduite avec pour chacun un seul de nos agents voulant accéder à ses objectifs pour un jeu d'affectation déjà connu. C'est à dire une grande quantité de sous-problèmes de type TSP-PR. Le sous-problème de chaque agent, avec chaque jeu d'affectation possible de ses objectifs sera traité. Une fois chaque sous-problème traité, nous connaîtrons tous les sous-trajets pouvant être empruntés par des agents, et donc pourront de ce fait éliminer les sous-trajets non-pertinents. Nous verrons lors des tests que la quantité de sous-trajet, dont dépendent nos variables décisionnelles de type f, n ou l, sera grandement réduite, ce qui de ce fait réduira considérablement le temps d'exécution pour la résolution du problème.

Cette méthode, associée à un problème de type MA-TSP-PR-DL, est évidement une simplification exacte, qui est équivalente au modèle d'origine, car seul les sous-trajets qui n'ont aucune chance d'être sélectionnés dans la solution optimale seront supprimés.

Une adaptation de cette méthodologie est aussi faite avec les choix de localisations des parkings, en traitant de la même manière, en plus, toute les possibilités d'existences de parkings dans les sous-problèmes de chaque agent. Ceci afin de retirer chaque sous-trajet possible, quelque soit le choix d'affectations d'objectifs et d'existences pour les parkings que l'on pourrait faire. Cette méthode est donc adaptable (et a été testée) aux MA-TSP-PRL et aux MA-TSP-PRL-DL, même si l'explication détaillée qui est faite ici se place dans dans un cas de MA-TSP-PR-DL. Les problèmes intégrants une capacité de parking peuvent aussi être simplifiés par cette méthode, il s'agira cette fois-ci de traiter l'ensemble des possibilités de disponibilité de parking (dans le pire des cas, l'ensemble $\mathscr{P}(P)\backslash\{\}$, qui est l'ensemble de tous les ensembles que l'on peut former avec les éléments de l'ensemble P, sauf l'ensemble vide).

Méthode :

La méthode sera illustrée par un exemple le plus simple possible de MA-TSP-PR-DL. Car plus de parkings ou de commerces engendreraient une très grande quantité de liaisons à représenter en plus, l'illustration deviendrait de ce fait illisible. En exemple donc, la figure 4.1 nous montre une zone constituée de 2 localisations pour placer 2 objectifs. 2 agents devront respectivement atteindre 1 ou 2 des objectifs, ceci en provenant de différentes entrées et sorties. Il y a aussi 2 parkings sur cet exemple.

• Étape 1 : On récupère chaque catégorie possible d'agent.

Une catégories d'agent est définie par 3 critères :
— Les objectifs
— Le point d'entrée
— Le point de sortie

Dans le cas où tous les objectifs peuvent affecter toutes localisations existantes, on peut considérer comme critère de catégorie le nombre d'objectifs au lieu des objectifs eux-mêmes. Cela rend le pré-traitement beaucoup plus rapide (moins de catégorie à traiter, on évite les pré-traitements identiques réalisés plusieurs fois). Ce sera le cas pour nos tests. On associe

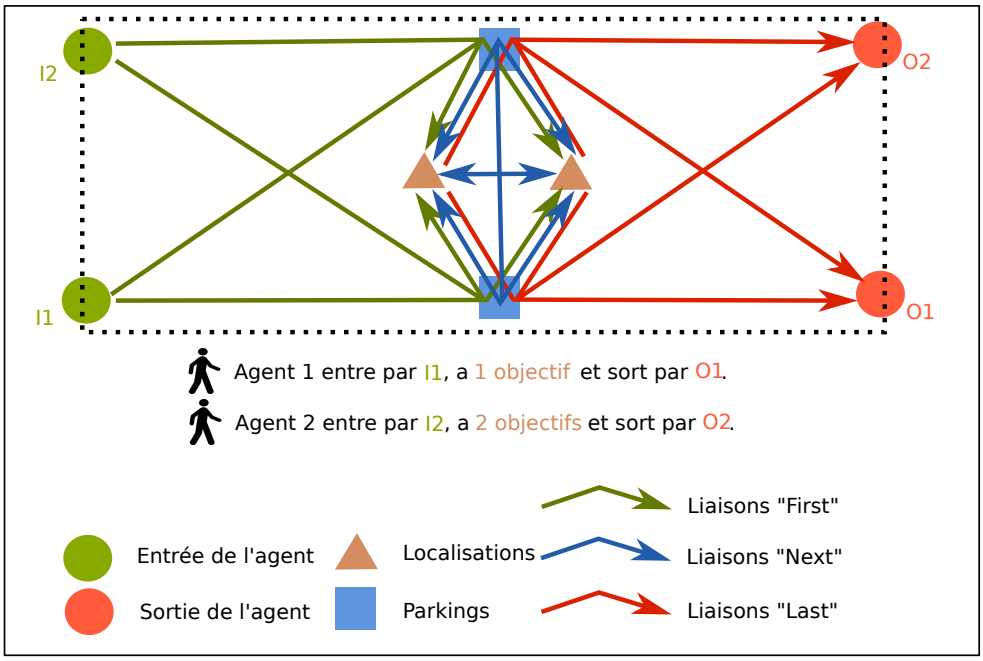


FIGURE 4.1 – Simplification par pré-traitement des sous-trajets pertinents - MA-TSP-PR-DL d'exemple. 2 objectifs en tout, chacun pouvant affecter chacune des 2 localisations.

à chacune de ces catégories toutes les possibilités de localisation (d'après le nombre d'objectifs). En découle de nombreux (mais très réduit en terme de complexité) TSP-PR. Dans chacun de ces sous-problèmes, un agent unique cherche à atteindre une ou plusieurs localisations fixes en utilisant les liaisons, avec un point d'entrée et un point de sortie connus. Sur la figure sont présentés 4.2 les TSP-PR résultants de cette zone d'exemple pour cette première étape. Pour l'agent 1, les deux localisations possibles de son unique objectif devront être traitées de manière individuelle. Pour l'agent 2, ses 2 objectifs ne peuvent affecter qu'un seul ensemble de localisations possible (c'est le résultat du TSP-PR qui choisira l'ordre de passage le plus rapide entre ces deux points, qu'importe les affectations des deux objectifs sur ces deux localisations), il n'y a donc qu'un seul TSP-PR à résoudre pour ce dernier.

- Étape 2 : On résout ces problèmes de transport.

Le trajet optimal pour chacun de ces micro-problèmes, composé de plusieurs Il est illustré par la figure 4.3, où l'on retrouve les trajets solutions (en tant qu'exemple) de nos TSP-PR associés. Chacune de ces liaisons sera stockée dans une liste globale, qui recensera toutes les liaisons qui ont été trouvées pertinentes.

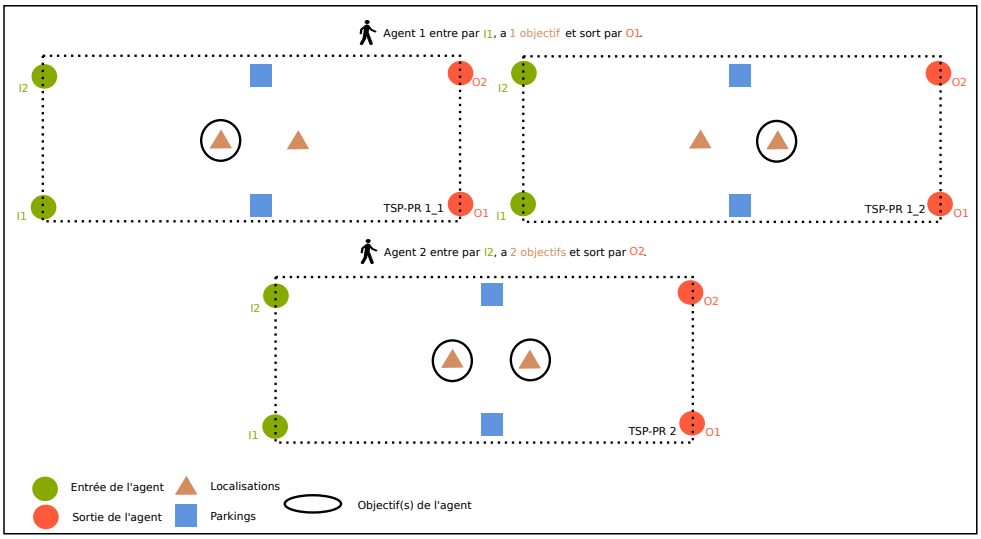


FIGURE 4.2 – Simplification par pré-traitement des sous-trajets pertinents - Étape 1

Définition 11. *Une **liaison pertinente**, dans un problème MA-TSP-PR-DL ou MA-TSP-PRL (et leurs variantes), est une liaison qui est utilisée pour une certaine affectation des objectifs et/ou des parkings dans le trajet le plus court d'un des agents. Les liaisons n'étant pas des liaisons pertinentes, considérées donc comme des **liaisons non-pertinentes**, ne font parties de la solution optimale au TSP-PR d'aucun agent, quelques soient les choix effectués au niveau des affectations des objectifs ou des parkings. Ces dernières peuvent donc être exclues du problème sans perturber la solution optimale.*

• Étape 3 : Une fois tous les sous-problèmes résolus, on récupère notre liste remplie durant l'étape 2.

Cette liste contient l'intégralité des liaisons pouvant être utilisées pour trouver le trajet optimal de n'importe quel agent sur n'importe quelle localisation d'objectif possible. Elle remplacera notre précédente liste de liaisons pour l'optimisation linéaire. La figure 4.4 nous montre le graphe allégé résultant de l'exemple que nous avons suivi étape par étape pour présenter cette méthode. Bien évidement, notre MA-TSP-PR-DL n'est pas encore résolu à ce stade, cette simplification n'a pour but que d'alléger le graphe des liaisons en sous-trajets avant optimisation.

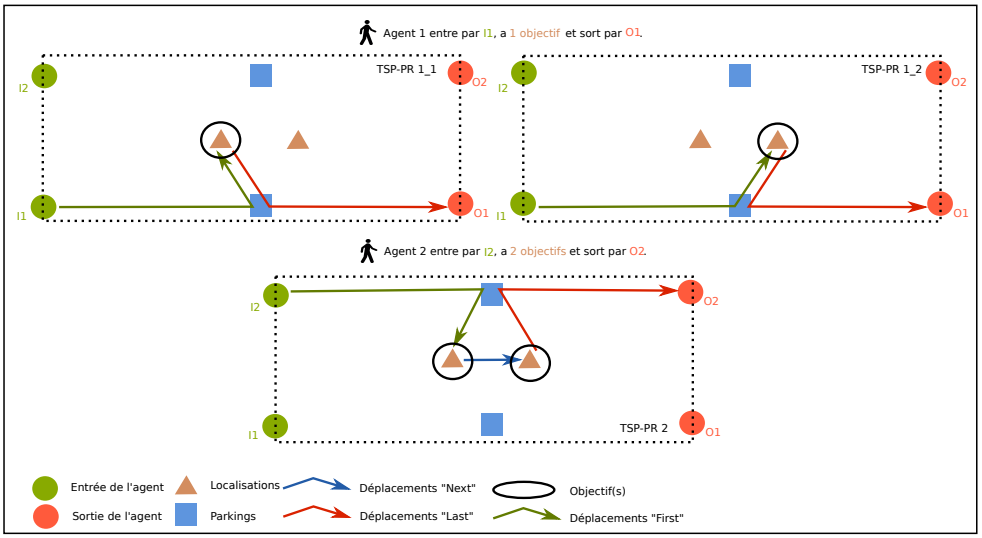


FIGURE 4.3 – Simplification par pré-traitement des sous-trajets pertinents - Étape 2

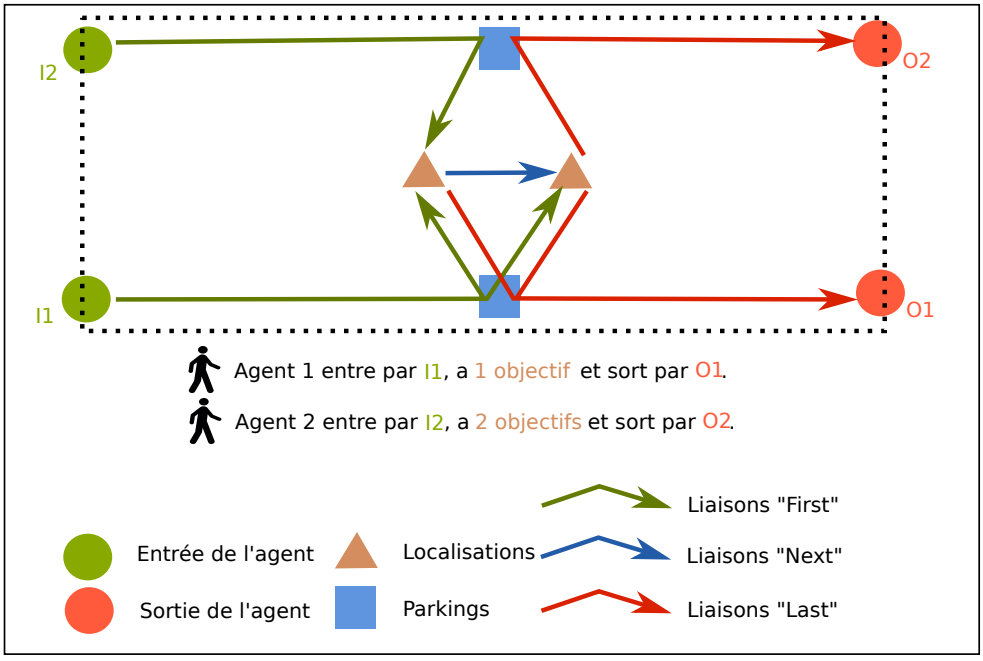


FIGURE 4.4 – Simplification par pré-traitement des sous-trajets pertinents - Résultat final : graphe allégé uniquement constitué des sous-trajets pertinents

4.1.3 Décomposition Trajets / Affectations

Un autre aspect qui rend notre problème complexe est sa composition de deux problèmes : un problème de localisation (affectation des objectifs sur les localisations, choix de parkings) avec un problème de transport (MA-TSP-PR). En s'appuyant sur un début de méthodologie similaire à la première simplification, nous allons permettre la décomposition de ce problème. Dans un premier temps, comme pour la précédente simplification, nous allons réaliser des sous-problèmes de type TSP-PR pour chaque catégorie d'agent et pour chaque jeu d'affectations possible. Au lieu des liaisons pertinentes, nous conserverons cette fois-ci les coûts minimaux obtenus pour chaque jeu de localisations, d'entrée et de sortie. Ce dernier nous servira dans un nouveau modèle linéaire qui cherchera à localiser les commerces en fonction des coûts de chaque trajet complet possible. Aussi, comme pour la précédente, cette simplification n'aura aucun impact sur le résultat.

Méthode :

• Étape 1 : Identique à celle de la première simplification. On récupère les catégories d'agent et on modélise les TSP-PR pour chaque jeu d'affectations possible de ses objectifs.

• Étape 2 : On résout ces problèmes de transport. Nous obtenons pour chacun le trajet optimal, ayant un coût de transport optimal. Chacun de ces coûts sera stocké dans une liste et associé à ses différentes localisations et entrées/sorties.

• Étape 3 : Une fois tous les TSP-PC résolus, on récupère notre liste de coût. Cette liste de coût servira de donnée pour notre nouveau modèle présenté dans la section suivante.

Une grande partie de cette méthodologie est identique en tout point à la précédente simplification, raison pour laquelle celle-ci n'est pas illustrée étape par étape (on pourrait reprendre les figures 4.1 à 4.3 pour expliquer les étapes 1 et 2). La figure 4.5 permet cependant d'illustrer le nouveau problème auquel nous sommes confrontés en sortie de ce pré-traitement. Ceci en s'appuyant justement sur l'exemple des figures 4.1 à 4.3 (le même graphe et les mêmes agents).

Cependant, il est important de noter que cette méthode s'adaptera beaucoup plus difficilement aux problèmes prenant en compte des choix

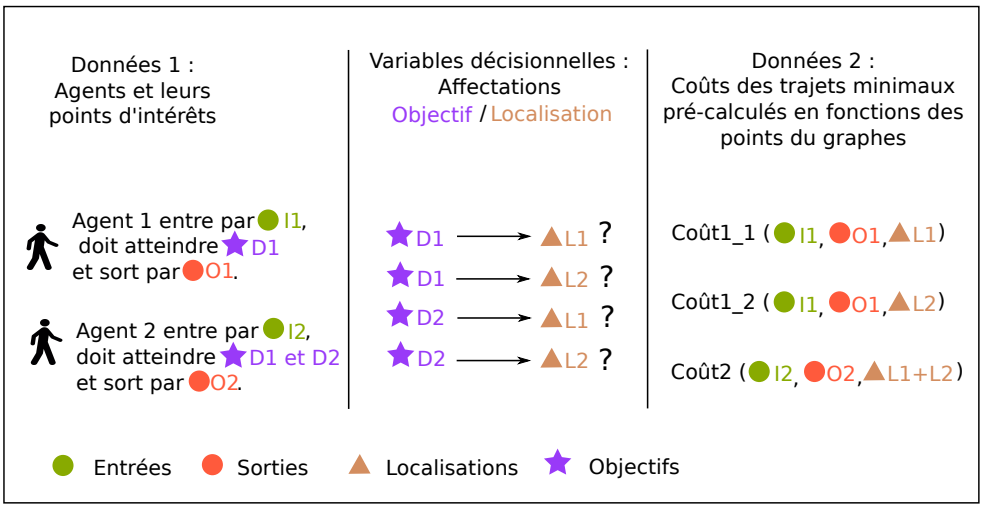


FIGURE 4.5 – Simplification par décomposition Trajets / Affectations : Le problème d'affectation après pré-traitement des trajets complets possibles.

au niveau des parkings. La liste de coûts associés aux différentes listes de localisation à atteindre et aux différentes entrées et sorties possibles, déjà conséquente, deviendra vite très grande si on lui associe aussi en plus la liste de tous les parkings pouvant être disponibles ($\mathscr{P}(P)\backslash\{\}$). Cette méthodologie n'a pas été testée dans un problème MA-TSP-PRL et MA-TSP-PRC (et leurs variantes) pour cette raison, ne convenant pas pour ces type de problème.

4.1.4 Modélisation du problème d'affectation pour un MA-TSP-PR-DL après une décomposition Trajets / Affectations

Notre nouveau modèle simplifié, qui ne se contentera lui que d'attribuer les affectations optimales, s'appuiera sur de nouvelles données d'entrées (que l'on récupère dans la première partie du traitement, étape 3).

Certaines de ces données sont similaires à nos modèles complets :

- les noeuds du graphe sauf les parkings (liste des entrées / sorties et localisations) ;
- la liste des objectifs ;
- les agents (avec leur entrée/sortie et leurs objectifs visés) ;
- les affectations possibles (objectifs -> localisations).

Nous aurons aussi de nouvelles données nécessaires :

- Une matrice des coûts du chemin le plus court pour chaque jeu d'entrée/sortie et groupe de localisations qui pourrait être ciblé par nos agents.

La nouvelle matrice de coût associe le coût du chemin le plus court à chaque combinaison possible d'entrée/sortie et de localisation. On prendra en considération uniquement les combinaisons pouvant réellement exister, en fonction des agents, de leurs objectifs et des affectations possibles (d'après notre pré-traitement.

Soit $\mathscr{D}$ l'ensemble des sous-ensembles possibles d'objectifs de D ("possible" signifie qu'il existe au moins un agent qui a exactement ce sous-ensemble d'objectifs), $\mathscr{D} \subset \mathscr{P}(D)$.

$$\forall D' \in \mathscr{D}, D' \subset D.$$

Chacune des listes d'objectifs de nos agents est incluse dans cet ensemble.

$$\forall a \in A, D^a \in \mathscr{D}$$

Inversement, tout élément de cet ensemble est aussi la liste exacte d'objectifs d'au moins un de nos agents :

$$\forall D' \in \mathscr{D}, \exists a \in A / D^a = D'$$

Soit $\mathscr{W}$ l'ensemble des sous-ensembles possibles de localisations de W (d'après les ensembles de localisations que peuvent prendre les éléments de $\mathscr{G}$), $\mathscr{W} \subset \mathscr{P}(W)$. Par mesure de simplicité on pourra récupérer directement cet ensemble d'ensembles durant le pré-traitement.

$$\forall W' \in \mathscr{W}, W' \subset W.$$

La figure 4.6 illustre les relations entre un ensemble de localisations W, l'ensemble de ses sous-ensembles $\mathscr{P}(W)$, et un ensemble des sous-ensembles possibles $\mathscr{W}$ qui pourrait entrer en jeu pour la résolution d'un MA-TSP-PR-DL par cette simplification. On retrouve bien évidement la même relation entre D et $\mathscr{D}$.

Soit $C^{i,o}_{W'}$ la matrice des coûts. Pour tout $i \in I, o \in O, W' \in \mathscr{W}$, le coût $C^{i,o}_{W'}$ correspond au coût du plus court chemin pour atteindre toutes les localisations du sous-ensemble $W' \subset W$ en partant de i et en sortant par o.

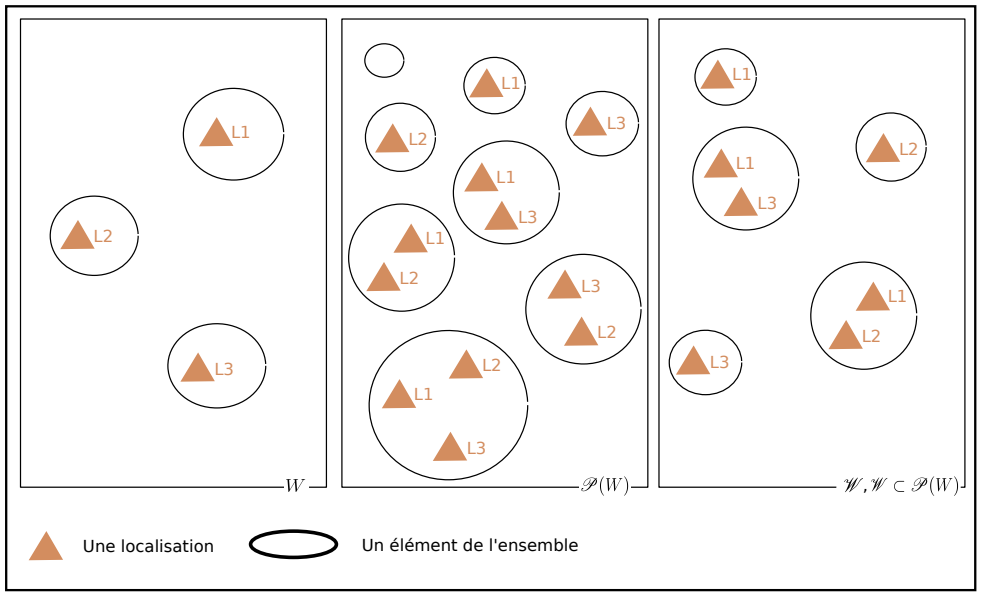


FIGURE 4.6 – Un exemple d'ensemble $\mathscr{W}$ en fonction de l'ensemble W et de l'ensemble $\mathscr{P}(W)$ associé.

Nous utiliserons en variable décisionnelle pour l'affectation d'un objectif sur une localisation la variable z_u, $u \in U$, déjà utilisée précédemment (qui vaut 1 si l'objectif d^u affecte la localisation w^u).

Soit $Z_{W'}^{D'}$, une nouvelle variable artificielle qui permet la linéarisation entre plusieurs variables décisionnelles d'affectations z_u, telle que :

$$Z_{W'}^{D'} = \begin{cases} & \textit{si les objectifs de l'ensemble } D' \in \mathscr{D} \\ 1 & \textit{affectent uniquement les localisations de} \\ & \textit{l'ensemble } W' \in \mathscr{W}\textit{, de même dimension } (|D'| = |W'|) \\ 0 & \textit{sinon} \end{cases}$$

La fonction d'optimisation quant à elle minimisera la somme du coût de déplacement de chaque agent en fonction de l'ensemble de localisations choisi pour satisfaire ses objectifs (dépendant directement des valeurs de z_u), de son entrée et de sa sortie :

$$\text{Min} \sum_{a \in A} \sum_{W' \in \mathscr{W}} Z_{W'}^{D^a} * c_{W'}^{i^a, o^a}$$

Les deux contraintes liées à l'affectation sont conservées. C'est à dire le fait qu'un objectif affecte une et seulement une localisation, et qu'une localisation ne soit affectée au maximum que par un objectif.

À ces dernières nous rajouterons une contrainte demandant un et un seul sous-ensemble de localisation de $\mathscr{W}$ affecté pour chaque sous-ensemble d'objectifs de $\mathscr{D}$, avec un nombre d'éléments identiques entre les deux sous-ensembles (autant de localisations que d'objectifs) :

$$\forall D' \in \mathscr{D}, \sum_{\substack{W' \in \mathscr{W} \\ |W'|=|D'|}} Z_{W'}^{D'} = 1$$

La fonction d'optimisation étant une minimisation, et les coûts de déplacement étant toujours positif, si le nombre d'éléments diffère entre l'ensemble de localisations et l'ensemble d'objectifs, Z prendra obligatoirement la valeur de "0".

Enfin, une quatrième et dernière contrainte fixera les relations entre Z et z. Si une localisation w n'est affectée par aucun des objectifs de l'agent a, alors pour tout $W' \in \mathscr{W}$ tel que $w \in W'$, $Z_{W'}^{D'} == 0$.

$$\forall a \in A, w \in W, \sum_{\substack{W' \in \mathscr{W} \\ w \in W'}} Z_{W'}^{D^a} \leq \sum_{\substack{u \in U \\ w^u = w \\ d^u \in D^a}} z_u$$

Le modèle d'affectation pour un MA-TSP-PR-DL, après pré-traitement des trajets :

$$\begin{aligned}
\text{Min} \quad & \sum_{a \in A} \sum_{W' \in \mathscr{W}} y_{W'}^{D^a} * c_{W'}^{i^a, o^a} & & \\
\text{sujet-à :} \quad & \sum_{\substack{u \in U \\ d^u = d}} z_u = 1 & \forall\ d \in D & (4.1) \\
& \sum_{\substack{u \in U \\ w^u = w}} z_u \leq 1 & \forall\ w \in W & (4.2) \\
& \sum_{\substack{W' \in \mathscr{W} \\ |W'|=|D'|}} y_{W'}^{D'} = 1 & \forall D' \in \mathscr{D}, & (4.3) \\
& \sum_{\substack{W' \in \mathscr{W} \\ w \in W'}} y_{W'}^{D^a} \leq \sum_{\substack{u \in U \\ w^u = w \\ d^u \in D^a}} z_u & \forall a \in A, w \in W & (4.4)
\end{aligned}$$

4.1.5 Expérimentations numériques

Les tests de ces deux simplifications ont été réalisés de la manière suivante :

1. Plusieurs situations aléatoires de même complexité sont générés (voir Annexe 2 - Génération de tests), avec pour chacune leur fichier de données respectif pour OPL, associé à notre modélisation MA-TSP-PR-DL.
2. On effectue pour chaque situation un pré-traitement sur les données pour réduire la liste de liaisons à celles pertinentes uniquement (première simplification). On génère le fichier de données pour CPLEX résultant pour notre modélisation MA-TSP-PR-DL en sous-trajet.
3. On effectue une recherche de la matrice des coûts en fonction des trajets individuels optimaux pour chaque situation. On génère les fichiers de données pour notre modélisation d'affectations sans trajet pour le MA-TSP-PR-DL.
4. Pour chaque situation, on résout les 3 modèles.
5. Enfin, on compare les trois méthodes, en fonction du temps nécessaire, pour la résolution du problème.

Les résultats de ces tests sont présentés en détail dans l'annexe C.3.

Conclusions aux expériences numériques

Il est possible de diminuer énormément le temps de résolution avec un pré-traitement des données pour éliminer les sous-trajets non-pertinents. En particulier s'il y a un nombre élevé de parkings. La simplification en décomposant le problème, pré-traitant la partie trajet dans un premier temps pour poser uniquement un problème d'affectation est aussi très intéressant, donnant même de légèrement meilleurs résultats (temps de résolution) que la première simplification. Dans un cas de MA-TSP-PRL ou de MA-TSP-PRC (et leurs variantes) il est préférable d'utiliser la première simplification car la seconde risquerait d'engendrer beaucoup trop de données (la matrice de coût dépendrait à la fois des entrées/sorties possibles, des lieux à atteindre et des parkings existant). Il est conseillé de ne l'utiliser donc qu'en cas de problème MA-TSP-PR-DL (localisation d'objectifs uniquement). Dans la section suivante des heuristiques sont testées afin de voir s'il est possible avec un calcul beaucoup plus rapide, sans optimisation linéaire, de trouver des affectations d'objectifs acceptables même sans être forcément optimales.

4.2 Heuristiques de placement des objectifs dans un MA-TSP-PR-DL

4.2.1 Introduction

Précédemment nous avons vu des méthodes exactes, par modélisation linéaire, permettant de connaître la localisation optimale des ressources dans le problème MA-TSP-PR-DL. Ces méthodes exactes vont permettre aussi de tester plusieurs heuristiques. Plusieurs heuristiques sont présentés et testées afin de les comparer, non plus au niveau du temps car leur temps d'exécution sera relativement négligeable, mais au niveau de leurs résultats. En effet, contrairement à toutes les méthodes qui ont été vues, ces heuristiques n'assureront pas de trouver une solution optimale.

Les heuristiques sont présentés ici de la plus simpliste à d'autres de plus en plus élaborées. Le problème sera vu ici comme étant un problème "P-median" d'affectation, ne prenant pas en compte le trajet complet des agents. Les heuristiques qui sont proposés s'inspirent principalement des méthodes heuristiques souvent utilisées pour ce type de problème : la méthode gloutonienne et la recherche locale. Ces méthodes sont bien évidement réadaptées ici à notre contexte et notre graphe. Le placement de chaque objectif peut donc être vu comme étant un problème "1-median". Il s'agit de trouver son emplacement idéal parmi plusieurs localisation pour satisfaire au mieux ses **utilisateurs** (réduire leurs coûts de déplacement). La méthode gloutonne consiste à résoudre un à un ces problèmes "1-median", objectif par objectif, en ne permettant à chaque fois le placement que sur une localisation qui n'est pas déjà affectée. Suite à cela, on peut effectuer une recherche locale en essayant de déplacer les différents objectifs pour réduire le coût. Cette fois-ci, l'affectation sur une localisation déjà affectée par un autre objectif est autorisée (il y aura à ce moment-là un échange de localisation entre les deux objectifs).

Définition 12. *On définit comme* ***utilisateur d'un objectif*** $d \in D$ *tout agent* $a \in A$ *qui doit atteindre* d*, autrement dit, dont la liste d'objectifs* D^a *respecte la relation* $d \in D^a$*. On note* A^d *la liste des utilisateurs d'un objectif* $d \in D$*.*

$$\forall a \in A^d, d \in D^a$$

4.2.2 Heuristique gloutonne sur le coûts aux entrées et sorties des utilisateurs

Résumé de la méthode : Algorithme glouton de placement des objectifs en fonction de leurs coûts aux entrées/sorties de leurs utilisateurs. Sans recherche locale.

Il s'agira ici simplement de placer un par un les objectifs (dans un ordre de traitement aléatoire) sur les localisations. La qualité qui permettra de choisir la localisation des objectifs sera le coût aux entrées et sorties des utilisateurs de l'objectif.

De manière plus détaillée, pour chaque objectif $d \in D$, nous regardons chaque agent $a \in A$ l'ayant dans sa liste d'objectifs ($d \in D^a$). Pour chacune des localisations possibles de cet objectif, nous calculons la somme des coûts minimums pour que les utilisateurs accèdent à cette localisation depuis leur entrée, ce à quoi nous rajoutons de la même manière la somme des coûts minimums pour que ces mêmes utilisateurs accèdent à leur sortie depuis cette localisation.

Soit H_w^d le coût d'un placement, par cette méthode heuristique, d'un objectif $d \in D$ sur une localisation $w \in W$. H_w^d se calcule donc ainsi :

$$H_w^d = \sum_{a \in A^d} (\min_{p \in P} C_{i^a pw} + \min_{p \in P} C_{wpo^a})$$

La localisation $w \in W$ qui sera choisie pour cet objectif $d \in D$ sera celle qui engendre le plus petit coût H_w^d. Une fois un objectif placé, on passe au suivant, et la localisation qui vient d'être choisie n'est plus disponible (les objectifs suivant ne pourront pas l'affecter). La figure 4.7 permet d'illustrer la méthode de choix d'un placement d'un objectif $d \in D$.

Il est à noter que cette méthode est adaptée au cas où chaque objectif puisse affecter toutes les localisations disponibles à cet effet. Si ce n'est pas le cas, l'algorithme glouton peut se retrouver avec des commerces restant ne pouvant pas affecter les localisations restante. Dans ce cas, si cela se produit, il faudra relancer (probablement plusieurs fois) ce processus, avec un ordre de traitement des objectifs différents. Cela jusqu'à obtenir un ordre de traitement qui admettra une affectation complète des objectifs.

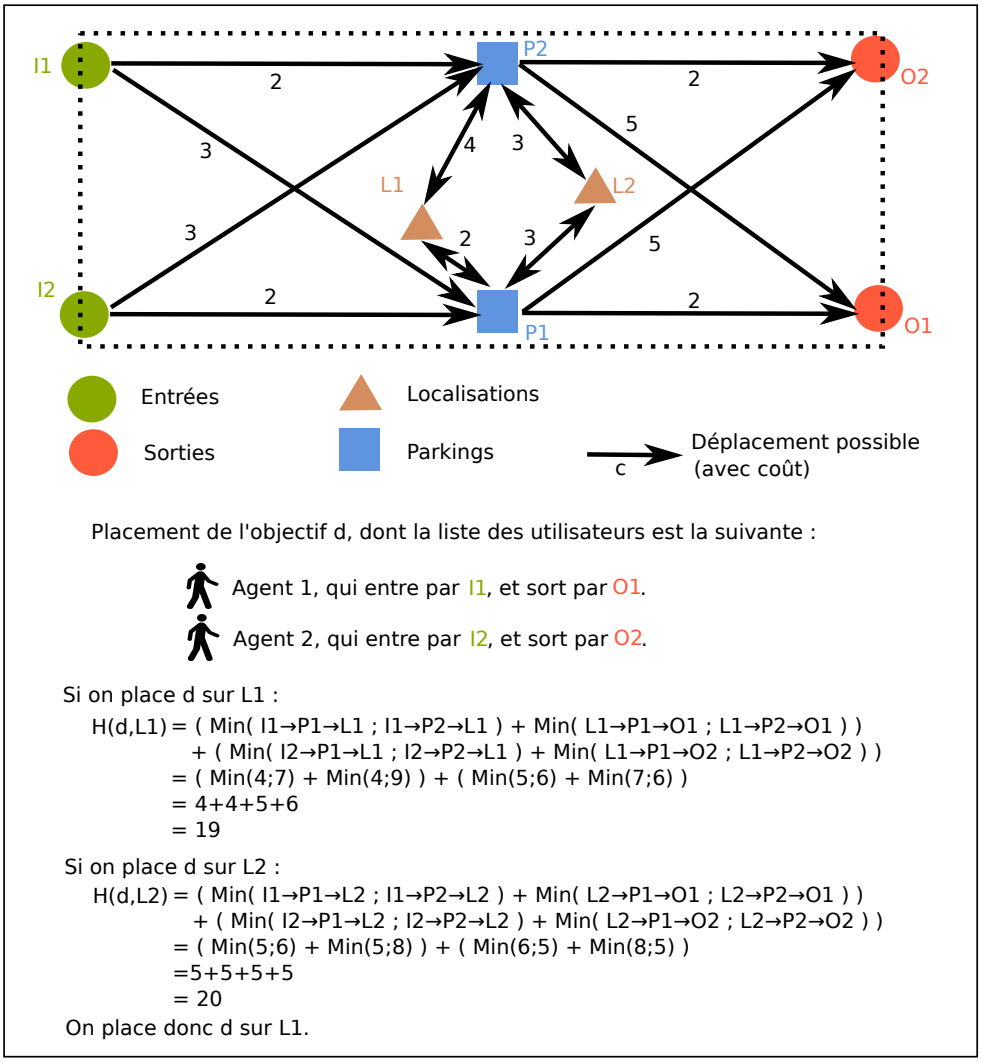


FIGURE 4.7 – Placement d'un objectif $d \in D$ par méthode gloutonne en fonction des distances aux entrées et sorties des utilisateurs $a \in A^d$.

4.2.3 Heuristique gloutonne & recherche locale sur le coûts aux entrées et sorties des utilisateurs

Résumé de la méthode :

1. Algorithme glouton de placement des objectifs en fonction de leur distance aux entrées et sorties de leurs utilisateurs.
2. Recherche locale en fonction de la distance aux entrées et sorties des utilisateurs.

La première étape de cette heuristique est exactement la même que pour l'heuristique précédente. On place un par un les objectifs $d \in D$ sur la localisation $w \in W$ qui n'est pas déjà affectée et donnant le plus petit coût H^d_w.

Il s'agira ensuite, après avoir trouvé l'ensemble des affectations de la première heuristique, de faire une recherche locale en tentant des changements d'affectations. Chaque déplacement d'objectif va être tenté. Tenter un déplacement signifie que l'on va calculer l'impact sur le coût total, d'un déplacement d'objectif. Si ce déplacement s'avère bénéfique (le déplacement tend à diminuer le coût) on le réalise, sinon ce déplacement n'est pas effectué.

Si on veut calculer le coût du déplacement de l'objectif $d \in D$ actuellement en localisation $W_1 \in W$ vers la localisation $w_2 \in W$, il y aura deux cas possibles à distinguer :

- Si w_2 n'est pas affectée : il faudra simplement calculer la différence entre $H^d_{w_2}$ et $H^d_{w_1}$. Cela représente la différence du coût aux entrées et sorties de ses utilisateurs entre la nouvelle position de d et son ancienne position. Autrement dit, il s'agit de la conséquence qu'aura ce déplacement de d sur le total de tous les coûts des objectifs aux entrées et sorties de ses utilisateurs.
- Si w_2 est déjà affectée, par l'objectif $d_2 \in D$, ce sera un échange de localisations entre d et d_2 : il faudra bien évidement aussi calculer la différence entre $H^d_{w_2}$ et $H^d_{w_1}$ mais en lui ajoutant la différence entre $H^{d_2}_{w_1}$ et $H^{d_2}_{w_2}$ (le coût de déplacement de d_2 partant de w_2 pour aller sur w_1).

Soit $\Delta H^d_{w_1 w_2}$ le coût de déplacement d'un objectif $d \in D$ depuis une localisation w_1 vers une localisation w_2 non affectée. $\Delta H^d_{w_1 w_2}$ se calcule comme tel :

$$\Delta H^d_{w_1 w_2} = H^d_{w_2} - H^d_{w_1}$$

Si $\Delta H^{d}_{w_1 w_2}$ est négatif, réaliser ce déplacement apporte un gain. Il faut donc le réaliser (si w_2 n'est pas déjà affecté).

Soit $\Delta H^{d_1 d_2}_{w_1 w_2}$ le coût d'un échange de localisation entre un objectif $d_1 \in D$ affectant la localisation $w_1 \in W$ et un objectif $d_2 \in D$ affectant la localisation $w_2 \in W$. $\Delta H^{d_1 d_2}_{w_1 w_2}$ se calcule comme tel :

$$\Delta H^{d_1 d_2}_{w_1 w_2} = \Delta H^{d_1}_{w_1 w_2} + \Delta H^{d_2}_{w_2 w_1}$$

$$\Delta H^{d_1 d_2}_{w_1 w_2} = H^{d_1}_{w_2} - H^{d_1}_{w_1} + H^{d_2}_{w_1} - H^{d_2}_{w_2}$$

De la même manière, si $\Delta H^{d_1 d_2}_{w_1 w_2}$ est négatif, réaliser cet échange de localisation apporte un gain. Il faut donc le réaliser.

Une fois que l'on a tenté chaque déplacement possible :

- Si un déplacement a été effectué durant le processus, on recommence (car un déplacement réalisé pourrait engendrer un nouveau déplacement bénéfique possible). On reprend chaque objectif un par un. Sur chacun on retente chaque déplacement possible.
- Si aucun déplacement n'a été effectué durant un cycle complet d'objectifs, on arrête le processus (il s'agira du point d'arrêt de cette heuristique).

Le coût total ayant une borne minimale (dans tous les cas il est supérieur à 0), les déplacements ne s'effectuant que si il y a diminution stricte du coût total et le nombre d'affectations possibles des objectifs sur les localisations étant fini, le point d'arrêt de cette méthode est certain.

4.2.4 Heuristique gloutonne & recherche locale sur le coûts aux entrées, sorties et objectifs des utilisateurs

Résumé de la méthode :

1. Algorithme glouton de placement des objectifs en fonction de leur distance aux entrées et sorties de leurs utilisateurs.
2. Recherche locale en fonction de la distance aux entrées/sortie des utilisateurs ainsi que la distance aux autres objectifs des utilisateurs.

Cette méthode heuristique est similaire à la précédente. Il est cependant rajouté la prise en compte des coûts inter-objectifs. Ces dernier ne sont cependant pas intégrables à l'étape 1 (algorithme glouton) car tous les objectifs ne sont pas encore placés. Un poids est fixé sur les distances inter-objectif. Car les coûts inter-objectifs et les coûts aux entrées/sorties ne sont pas forcément des éléments équivalents à comparer. On cherchera à faire varier ce poids afin de trouver le poids qui admettra les meilleurs résultats lors des expériences numériques.

Soit α un poids qui multipliera les coûts inter-objectifs. Ce paramètre est un nombre décimal positif : $\alpha \in \mathbb{D}^+$.

Nous aurons besoin pour déterminer les nouveaux coûts de déplacement (recherche locale de l'étape 2) de la variable d'affectation $z_u, u \in U$, que nous renommerons ici, pour plus de clarté, z_w^d, l'affectation d'un objectif $d \in D$ sur une localisation $w \in W$.

Même s'il ne sera, en tant que tel, jamais utilisé, notons Q_w^d le coût d'un placement d'un objectif $d \in D$ sur la position $w \in W$, en considérant les distances inter-objectifs. Les simples placements ne s'effectuent que durant l'étape 1, et les distances inter-objectifs ne seront pas prises en compte à cette étape 2. Néanmoins, nous nous servirons de ce coût pour calculer ensuite les coûts de déplacement et d'échange de localisation d'objectifs.

$$Q_w^d = H_w^d + \alpha * \sum_{a \in A^d} \sum_{\substack{d_2 \in D^a \\ d_2 \neq d}} \sum_{w_2 \in W} (\min_{p_1, p_2 \in P} z_{w_2}^{d_2} * C_{w p_1 p_2 w_2})$$

Cette écriture, vis à vis de la pondération, reste très discutable. Elle a été choisie au début certainement à tord, au dépens d'une pondération plus traditionnelle de type $\alpha * A + (1 - \alpha) * B$ pour un $\alpha \in [0; 1]$. En effet cette deuxième écriture aurait été plus pertinente pour prendre en compte le spectre complet des possibilités de pondération entre les coûts aux entrées/sorties et les coûts inter-objectifs. En particulier lorsque l'on veut mettre tout le poids (ou presque) sur l'un ou l'autre des deux membres de l'équation. Néanmoins, il sera avéré lors des expérimentations numériques que les valeurs les plus efficaces pour ce α tel que présenté dans la formulation choisie sont proches de 1. Cette formulation choisie ne pose donc aucun problème et a été conservée.

Nous utiliserons la notation $\Delta Q_{w_1 w_2}^d$ pour exprimer le nouveau coût d'un déplacement d'un objectif $d \in D$ depuis une localisation $w_1 \in W$

vers une localisation $w_2 \in W$. Son calcul sera le suivant :

$$\Delta Q^d_{w_1 w_2} = Q^d_{w_2} - Q^d_{w_1}$$

On pourrait aussi exprimer $\Delta Q^d_{w_1 w_2}$ en fonction de $\Delta H^d_{w_1 w_2}$ de la manière suivante :

$$\Delta Q^d_{w_1 w_2} = \Delta H^d_{w_1 w_2} + \alpha * \sum_{a \in A^d} \sum_{\substack{d_2 \in D^a \\ d_2 \neq d}} \sum_{w_3 \in W} (z^{d_2}_{w_3} * (\min_{p_1, p_2 \in P} C_{w_2 p_1 p_2 w_3} - \min_{p_1, p_2 \in P} C_{w_1 p_1 p_2 w_3}))$$

Et enfin notons $\Delta Q^{d_1 d_2}_{w_1 w_2}$ pour exprimer le nouveau coût d'échange de localisation entre un objectif $d_1 \in D$ affectant la localisation $w_1 \in W$ et un objectif $d_2 \in D$ affectant la localisation $w_2 \in W$. Nous retrouverons encore les égalités :

$$\Delta Q^{d_1 d_2}_{w_1 w_2} = \Delta Q^{d_1}_{w_1 w_2} + \Delta Q^{d_2}_{w_2 w_1}$$
$$\Delta Q^{d_1 d_2}_{w_1 w_2} = Q^{d_1}_{w_2} - Q^{d_1}_{w_1} + Q^{d_2}_{w_1} - Q^{d_2}_{w_2}$$

Ici encore, il est possible aussi d'exprimer $\Delta Q^{d_1 d_2}_{w_1 w_2}$ en fonction de $\Delta H^{d_1 d_2}_{w_1 w_2}$:

$$\Delta Q^{d_1 d_2}_{w_1 w_2} = \Delta H^{d_1 d_2}_{w_1 w_2} + \alpha * [\sum_{a \in A^{d_1}} \sum_{\substack{d_3 \in D^a \\ d_3 \neq d_1}} \sum_{w_3 \in W} (z^{d_3}_{w_3} * (\min_{p_1, p_2 \in P} C_{w_2 p_1 p_2 w_3} - \min_{p_1, p_2 \in P} C_{w_1 p_1 p_2 w_3})) + \sum_{a \in A^{d_2}} \sum_{\substack{d_3 \in D^a \\ d_3 \neq d_2}} \sum_{w_3 \in W} (z^{d_3}_{w_3} * (\min_{p_1, p_2 \in P} C_{w_1 p_1 p_2 w_3} - \min_{p_1, p_2 \in P} C_{w_2 p_1 p_2 w_3}))]$$

Le processus de recherche locale s'effectue de la même manière que pour la méthode précédente, en utilisant cette fois-ci les nouvelles fonctions de coût de déplacement ou d'échange de localisation (qui prennent en compte les distances inter-objectif). Quand une tentative de déplacement ou d'échange donne un coût négatif, il y a un bénéfice sur le coût

global et donc on effectue le changement. On s'arrête quand on arrive sur une situation où plus aucun changement ne s'avère bénéfique.

4.2.5 Évaluation des heuristiques

Pour évaluer les heuristiques proposées nous allons nous servir de nos modèles linéaires d'optimisation. Grâce à ces modèles il est possible de connaître la valeur optimale à atteindre. Néanmoins, il ne serait pas objectif de comparer simplement le résultat des heuristiques avec cette valeur optimale. En effet, la comparaison entre ces deux valeurs n'apporte aucune indication quant aux valeurs que l'on obtiendrait avec une affectation non-optimisée. Une heuristique pourrait parfaitement donner un résultat supérieur de seulement 1% de l'optimal (rappel : notre optimisation est une minimisation) tout en réalisant en réalité les pires affectations possibles. De plus, notre méthode de réalisation de test à la chaîne va générer des situations avec des agents venant de toutes parts de manière totalement aléatoire. De ce fait la différence entre la meilleure et la pire localisation devrait normalement être relativement faible, par rapport à un cas réel. Le résultat obtenu si l'on compare le coût minimal obtenu si on localise les objectifs par notre heuristique ou par l'optimisation linéaire sera donc peu significatif.

Pour ces raisons, il est nécessaire d'avoir un 3ème coût total de comparaison. Il y aurait pour répondre à cette attente deux possibilités : le coût de déplacement pour la pire affectation possible ou le coût de déplacement pour une affectation "moyenne".

Le coût de déplacement pour la pire affectation possible est très difficile à estimer. En effet, il faudrait trouver le coût le plus grand, par les affectations, pour des agents qui eux réalisent toujours leur plus court chemin. Problème d'optimisation beaucoup plus complexe que notre problème initiale. Par contre l'estimation du coût de placement d'une affectation moyenne est assez facilement abordable. Il suffit de générer une affectation aléatoire et d'en résoudre le MA-TSP-PR, cela répété un grand nombre de fois. C'est donc une affectation aléatoire qui sera utilisée comme troisième point de comparaison et qui fera office de valeur moyenne pour un grand nombre de tests.

Nous avons donc deux points de comparaison, entre lesquels devraient se situer nos heuristiques. Nous évaluerons l'indice de performance selon ces points :

- Le coût pour une affectation aléatoire, qui fixe la valeur minimale à atteindre. Ce coût sera trouvé après résolution linéaire d'un MA-TSP-PR en prenant ces affectations aléatoires comme fixes. Cette valeur sera le "0" sur notre échelle de performance.
- Le coût pour une localisation optimale par résolution linéaire du MA-TSP-PR-DL. Cette valeur sera l'indice "1" sur notre échelle de performance.
- Le coût pour une affectation déterminée par l'heuristique testée, par résolution linéaire du MA-TSP-PR en prenant ces affectations comme étant fixes.

Ces coûts seront calculés sur des centaines de tests de MA-TSP-PR-DL générés (à minima) de complexité identique (même quantité pour chaque entité : parkings, objectifs, agents, etc...). De plus il est à noter que les coûts optimaux de ces MA-TSP-PR-DL resteront sur un même ordre de grandeur : la surface d'étude sera à chaque fois de mêmes dimensions et le coût des liaisons sera déterminé par distance euclidienne.

Soit N l'ensemble des MA-TSP-PR-DL de tests que l'on effectue. Pour tout $n \in N$ on note :

- $\mathscr{C}_1^n$ le coût de déplacement optimal de ce MA-TSP-PR-DL ;
- $\mathscr{C}_0^n$ le coût de déplacement optimal d'un MA-TSP-PR résultant après affectation aléatoire ;
- $\mathscr{C}_*^n$ le coût de déplacement optimal d'un MA-TSP-PR résultant après affectation par heuristique.

L'indice de performance de notre heuristique se calcule ainsi :

$$\text{Performance} = \frac{\sum\limits_{n\in N} \mathscr{C}_0^n - \sum\limits_{n\in N} \mathscr{C}_*^n}{\sum\limits_{n\in N} \mathscr{C}_0^n - \sum\limits_{n\in N} \mathscr{C}_1^n}$$

Il est bien évident que la fiabilité de notre estimation de performance dépendra du nombre de tests effectués. Plus N sera grand, plus $\sum\limits_{n\in N} C_0^n$ sera proche de la somme d'une affectation "moyenne" pour tous ces cas. Précisons également que rien empêche l'indice de performance de prendre une valeur négative. Ce cas signifierait que les affectations choisies par l'heuristique tendent plutôt à maximiser le coût total qu'à le minimiser. Le résultat du placement effectué par cette heuristique serait dans ce cas pire qu'un placement aléatoire, en moyenne.

En plus de notre indice de performance, nous nous intéressons aussi

à un deuxième indice que nous appelons indice de LOT (Localisations Optimales Trouvées). Ce dernier correspond au pourcentage de fois où l'heuristique aura trouvé un jeu de localisations optimales (dont le coût résultant du MA-TSP-PR de ce placement est strictement égale au coût trouvé par modélisation linéaire du MA-TSP-PR-DL complet).

$$\text{LOT} = \frac{\sum_{n \in N} (\mathscr{C}_*^n == \mathscr{C}_1^n)}{|N|}$$

4.2.6 Expérimentations numériques

Nous générons une centaine de situations à étudier pour chacun des tests pour réduire le bruit qui pourrait être généré à la fois par les heuristiques qui ne sont plus exactes et surtout par celui généré par le placement aléatoire qui fera office de moyenne. Comme précisé dans la précédente section, on commencera tout d'abord pour chaque situation, à lui associer un placement aléatoire (jeu d'affectations) et un placement résultant de l'heuristique. On résout ensuite le MA-TSP-PR-DL de cette situation pour trouver son coût minimal, puis les 2 MA-TSP-PR avec les deux différents placements (aléatoire et heuristique). Une fois les 100 situations traitées nous comparons la somme des coûts trouvés en MA-TSP-PR-DL (valeur optimale dont on aimerait se rapprocher avec les heuristiques) avec la sommes des coûts trouvés en MA-TSP-PR avec placement aléatoire ainsi que celle avec le placement choisi par notre heuristique.

Afin de faciliter ce processus toutes les heuristiques sur les même jeux de données. Donc en réalité nous ne comparerons pas uniquement 3 résultats totaux mais beaucoup plus. Pour chaque situation il y aura donc, en jeux de données : un premier, pour le MA-TSP-PR-DL, un deuxième, celui du MA-TSP-PR en placement aléatoire, et autant de jeux supplémentaires que l'on testera d'heuristiques. Les heuristiques 1 et 2 vues dans ce chapitre engendre donc 2 jeux de données à optimiser. Pour la 3ème heuristique, il y a un paramètre α à faire varier, dont la valeur idéale est à déterminer. Afin de connaître le paramètre α pour lequel elle est la plus efficace, un jeu de donnée est généré pour chaque valeur de ce paramètre comprise entre 0.2 et 1.9, par palier de 0.1. Ce qui rajoute 18 jeux de données supplémentaires pour tracer la courbe de performance et de LOT de la troisième heuristique en fonction de α. Pour chaque situation testée il y a donc en tout 1 MA-TSP-PR-DL et 20 MA-TSP-PR à résoudre.

Le détail des résultats de ces tests sont répertoriés dans l'annexe C.4.

Conclusions des tests

Les tests réalisés montrent tout d'abord les performances très faibles de la première heuristique, qui fournit des résultats très proches d'un placement aléatoire. Elle est donc à exclure. Il paraît donc très important de réaliser une recherche locale et ne pas se limiter à un simple algorithme glouton.

La deuxième méthode est acceptable, car très simple à réaliser pour des résultats toujours plus proches de l'optimal que d'un placement aléatoire. Toutefois, son résultat ne semble pas très stable en fonction de la complexité de la situation, puisque pour ces trois tests les indices de performance obtenus sont très variés : de 59.9% à 77.4%. Cet écart très important amène à penser que sur certaines complexités de situation cette méthode pourrait donner des résultats largement plus décevants. De plus, elle est nettement surpassée par la 3ème méthode, tant en termes de performance que de LOT, pour une stabilité bien meilleure.

Pour la troisième méthode, pour les trois tests réalisés et en faisant la moyenne des résultats obtenus, une valeur pour α de 1.3 semble être à la fois la plus performante, pour un indice de performance de 91.38% et ayant les meilleures chances de trouver le jeu d'affectations optimal, pour un indice de LOT de 33.33%. De plus, toutes les valeurs α comprises entre 1.2 et 1.4 ont obtenu pour chaque test un indice de performance supérieur à 90% et un indice de LOT supérieur à 30%. Une variation légère de ce paramètre ne semble donc pas avoir de trop fortes conséquences sur la qualité de l'affectation.

L'ajout de la prise en compte des interactions entre objectifs avec un $\alpha = 1.3$ fait passer, en moyenne pour ces 3 tests, d'un indice de performance de moins de 69% à un indice de performance de plus de 91%. La perte de performance vis à vis de l'affectation optimale (MA-TSP-PR-DL) par rapport à une affectation aléatoire a été divisée environ par 3.6 (on passe de 31% de perte à 8.6%). L'indice de LOT lui aussi a été nettement amélioré, passant d'une moyenne de 14.7% à 33.3%, soit plus du double.

4.2.7 Conclusions

Des simplifications exactes ont été formulés afin de pouvoir prétendre, dans un temps plus acceptable, résoudre des problèmes MA-TSP-PR (et variantes) un peu plus importants. Il ne serait pas vraiment envisageable ceci étant de résoudre le problème d'une zone commerciale complète en une seule fois par ces procédés mais on pourrait à minima envisager grâce à ces simplification de traiter ces dernières en plusieurs secteurs. Pour une zone complète, une heuristique étudiée semble donner des résultats encourageant pour le problème MA-TSP-PR-DL, au moins sur de petites instances. Il serait intéressant, même si ce ne sera pas réalisé dans cette étude, d'étudier sa performance sur de plus grandes instances. Le problème reste que pour connaître sa performance sur de grandes instances, il faut à minima être capable de résoudre le MA-TSP-PR associé, avec des affectations fixées, qui donnera le coût total pour les agents. Aussi, de la même manière, des heuristiques permettant de donner des résultats pour les problèmes MA-TSP-PRL seraient nécessaires. Celles développées dans ce chapitre ne concerne que le problème MA-TSP-PR-DL.

Ceci étant, cette étude s'arrêtera là quant à la résolution des problèmes tels qu'ils ont été vus jusqu'à présent : un problème intemporel d'une situation donnée qui ne prend pas en compte les congestions de trafic. De manière théorique les modélisations et heuristiques vues jusqu'à présent permettent de trouver des affectations optimales ou performantes, que ce soit pour les commerces ou parkings. Mais dans la réalité il pourra y avoir de fortes congestions de trafic engendré par les déplacements choisis qui déformeront du tout au tout les coûts qui ont été utilisés pour déduire l'ensemble de choix d'affectations et de déplacements. En gardant à l'esprit que nos modélisations linéaires sont déjà très coûteuses en temps de calcul, il ne serait pas envisageable d'introduire de fonction linéaire sur les coûts des liaisons pour prendre en compte le trafic. Il ne semble pas envisageable d'introduire une prise en compte de la congestion directement au niveau de nos modèles. C'est pour cela que dans le chapitre suivant, une méthodologie sera vue, permettant non pas de prendre en considération directement la congestion, mais simplement d'affiner nos résultats en fonction de congestion constatées (par simulation) après optimisation d'un problème.

Chapitre 5

Couplage avec MatSim

Sommaire

5.1 Introduction

5.1.1 Intérêt du couplage

Les différentes méthodologies vues jusqu'à présent ne tiennent pas compte d'un aspect très important, qui est la prise en compte du trafic, des ralentissements dûs à une trop forte utilisation d'une même liaison. C'est en effet un aspect important à prendre en compte si l'on considère que le temps de transport a une incidence sur le coût (ce n'est pas obligatoirement le cas, par exemple si l'on souhaite minimiser la distance

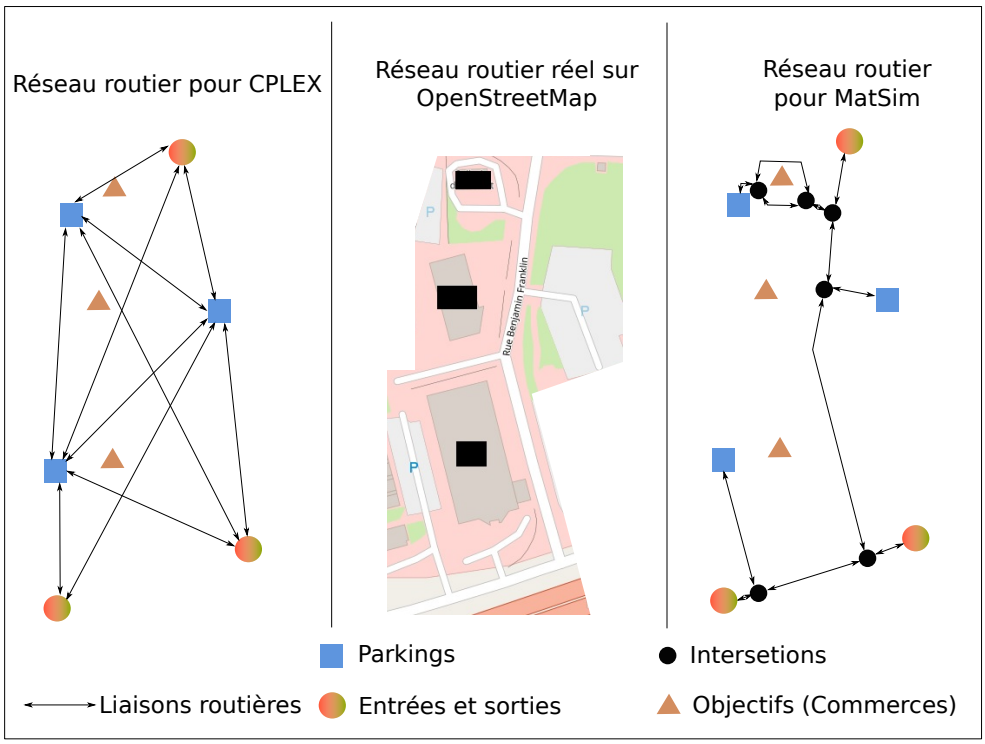


FIGURE 5.1 – Comparaison des graphes routiers pour CPLEX et pour MatSim.

parcourue). Si l'on considère le temps comme une composante du coût de déplacement, il sera intéressant de coupler ces méthodologies avec une simulation multi-agents, réalisée avec MatSim, qui est un SMA permettant des simulations très réalistes des trafics et accessibilités. Cette simulation permettra de recalculer les nouveaux temps d'accès en tenant compte de la circulation résultante des choix de localisation.

Avant cela, il est important de rappeler que le graphe de notre modélisation linéaire représente une simplification de la zone d'étude. Nous n'avons sur ce dernier que les points d'intérêts (les parkings, les objectifs, les entrées et les sorties). Les liaisons entre deux de ces noeuds représentent ainsi le trajet le plus court pour passer d'un de ces points à un autre. Pour la simulation multi-agents, nous prendrons un graphe contenant l'intégralité des axes routiers. En effet, les impacts sur le trafic ne peuvent être réalistes que si nous avons le réseau routier dans son ensemble. Une même route peut être utilisée sur plusieurs liaisons de notre modélisation linéaire. L'utilisation d'une liaison sur notre précédent graphe simplifié peut avoir une incidence sur le trafic des autres liaisons ayant des routes en commun. La figure 5.1 illustre ces différences, en partant, au centre, d'une même zone d'étude.

Une fois la simulation multi-agents effectuée (d'après les résultats de notre précédente optimisation), nous obtiendrons deux éléments intéressants :

- Le coût de la simulation : en récupérant les temps de la simulation nous pourrons connaître (par le calcul) le coût total de déplacement des agents. Cela sera un bonne évaluation plus proche de la réalité que notre modèle linéaire pour juger de l'efficacité d'un choix de localisation, par comparaison entre les itérations.
- Les nouveaux temps moyens d'accès : on recalcule les temps moyens de déplacement pour chaque liaison entre points d'intérêt (par exemple d'une entrée à un parking) du réseau logique de notre modèle linéaire.

Suite à cette simulation, le modèle d'optimisation est mis à jour avec les nouveaux temps d'accès et donc les nouveaux coûts. On réalise ensuite une nouvelle optimisation basés sur ces nouveaux coûts. Nous appelons une **boucle CPLEX-MatSim** l'enchaînement ainsi réalisé d'une optimisation linéaire sous CPLEX, suivie par une simulation sous MatSim, recalculant les temps d'accès.

Après une première boucle CPLEX-MatSim d'initialisation et le premier recalcul des temps, d'autres boucles CPLEX-MatSim poursuivent le processus. Il peut y avoir un point d'arrêt lorsque un résultat d'optimisation CPLEX (localisations et trajets) est identique au précédent : nous avons une solution stable. Cela ne signifie pas qu'une simulation meilleure serait impossible, mais que notre couplage MatSim avec CPLEX ne donnera pas de nouveaux résultats sur les boucles suivantes. En revanche si ce n'est pas le cas la boucle peut continuer (par une simulation MatSim).

Les boucles CPLEX-MatSim se répètent donc en espérant converger vers une situation stable. La convergence n'est pas garantie, ainsi il peut être nécessaire de se fixer un nombre limité d'itérations (de boucles). Qu'il y ait ou non convergence sur une situation stable (qui ne bouge plus de par ses affectations), l'ensemble d'affectation ayant donné le plus petit coût de simulation (MatSim) sera retenu comme la meilleure solution trouvée.

Bien entendu, tout ceci n'est pas suffisant pour définir la méthodologie qui sera mise en place dans ce cadre précis. Dans la section suivante plus de détails seront donnés quant à la mise en place du procédé, du point de

vue de la communication entre l'optimisation linéaire et MatSim. Mais cela ne sera que lors d'expérimentations numériques présentes dans ce chapitre que nous pourrons affiner, en fonction des résultats obtenus, comment mettre à jour les données de l'optimisation linéaire. C'est aussi en fonction de ces résultats que pourra être décidé, si besoin, quand arrêter le bouclage entre ces deux outils s'il n'y a pas de convergence.

5.1.2 Mise en place

Le passage de MatSim vers CPLEX a été réalisé par écriture et lecture de fichiers. MatSim aura besoin de 3 fichiers pour effectuer une simulation :

- le fichier "config.xml", la configuration de la simulation, qui restera la même pour tous nos tests.
- Le fichier "network.xml", qui définit la carte de notre étude, un graphe représentant le réseau routier réel de la zone.
- le fichier "plan.xml", des agents, qui définit le nombre d'agents et leur trajet.

L'optimisation sous CPLEX n'a besoin que de deux fichiers :

- Le fichier des données (*.dat).
- Le modèle (*.mod) utilisé pour l'optimisation linéaire qui lui est statique le même quelle que soit la zone d'étude. Le modèle, codé en OPL (Optimization Programming Language), permet aussi après résolution de créer directement le fichier du trajets des agents nécessaire à MatSim.

Avant de commencer la méthode de couplage il faut disposer d'une carte de la zone étudiée. C'est à dire du graphe du réseau routier réel, de la longueur entre les différents noeuds, de la liste des points d'intérêt (parkings, entrées, sorties, localisations d'objectifs) ainsi que de la correspondance entre les noeuds et des points d'intérêt. Un traitement informatique de ces données est nécessaire pour générer le fichier du graphe pour MatSim, qui est généré avant le couplage, une seule fois, et ne sera jamais modifié au fil des boucles. On connaît aussi la liste d'agents avec leurs objectifs, et les localisations possibles pour chaque objectif.

Les deux seuls fichiers qui seront modifié au fil des boucles sont donc le fichier du trajet des agents pour MatSim, et le fichier des données de l'optimisation linéaire pour CPLEX.

Voici les étapes d'un couplage entre MatSim et CPLEX :

- Initialisation : Un premier passage MatSim pour donner des temps initiaux à notre modélisation linéaire. On simule le passage d'un agent sur chaque trajet possible entre les différents points d'intérêt.Un fichier provisoire du trajet de ces faux-agents, contenant un agent pour chaque liaison entre point d'intérêt est créé pour permettre cette simulation d'initialisation. On créé après la simulation le fichier des données pour l'optimisation linéaire. Un traitement informatique est bien évidement nécessaire pour cela. Les résultats de la simulation MatSim sont récupérés depuis les résultats contenus dans un fichier de sortie de MatSim : "output_events.xml" (voir annexe B.4).
- Étape 1 - Partie CPLEX : On lance une optimisation linéaire de notre modèle sous CPLEX. Notre modèle, écrit en OPL permet de générer directement le fichier du trajet des agents ("plan.xml") nécessaire à MatSim, résultant de l'optimisation (voir annexe B.3.2). Aux trajets "réels" des agents, on peut, si nécessaire, rajouter un agent factice pour chaque trajet entre points d'intérêt qui n'a jamais été parcouru. On garde en mémoire les affectations choisies.
- Étape 2 - Partie MatSim : On lance une simulation MatSim d'après le fichier "network.xml" de la situation initiale et du fichier des agents dernièrement créé. On récupère les temps entre les points d'intérêt grâce au fichier de sortie "output_event.xml" généré par MatSim. Ce dernier nous permet de connaître les nouveaux temps moyens entre les points d'intérêt, mais aussi le temps total de trajet. Jusqu'à la réalisation du nombre souhaité de boucles, on se sert de ces nouveaux coûts pour générer un nouveau fichier de données pour CPLEX, et l'on retourne à l'étape 1. Dans tous les cas, on note en résultat l'affectation qui a eu lieu pour cette boucle (étape 1) et le résultat obtenu par la simulation sans compter les agents factices (étape 2).
- Étape finale : On compare les résultats obtenus, pour trouver la meilleure affectation parmi celles qui ont été simulées (cela ne sera pas forcément la dernière).

Schémas expliquant les échanges entre MatSim est CPLEX, notamment avec les échange/création de fichier.

5.1.3 Agents factices

Durant la section "Mise en place" (5.2), il a été question d'agents factices. Ils peuvent en effet être utilisés à 2 moments distincts. Ces agents sont considérés comme factices car n'existant pas dans le cadre de l'optimisation. Ils sont utilisés pour palier à des absences de temps pour des liaisons dans certains cas.

Pour l'étape d'initialisation, nous devons récupérer des temps de déplacement pour chaque trajet entre les points d'intérêt possibles. Les "vrais agents" ont des objectifs qui n'affectent, à ce moment là, aucune localisation. Nous n'avons donc aucune idée des chemins qui vont être parcourus. C'est la raison pour laquelle on peut réaliser une première simulation via MatSim avec un fichier d'agents constitué exclusivement d'agents factices. Ces derniers feront chacun un simple trajet entre deux points d'intérêt. Toutes les liaisons possibles entre point d'intérêt sont traitées chacune par un agent factice. Cette étape est facultative si on a déjà des temps fournis disponibles entre chaque point d'intérêt.

On peut utiliser aussi durant les boucles des agents factices sur les axes qui n'ont pas été utilisés par la solution de l'optimisation à l'étape 1. Deux liaisons différentes entre nos points d'intérêt (selon le graphe entre points d'intérêt pour CPLEX) peuvent utiliser des routes communes (selon le graphe complet des routes, pour MatSim). Et donc, même si elle n'est pas utilisée, une liaison entre deux points d'intérêt pourrait admettre des ralentissements sur une route (graphe MatSim) qu'elle a en commun avec une autre liaison très empruntée (graphe CPLEX). C'est pourquoi il est plus judicieux de rajouter un agent factice sur cet axe afin d'en connaître le temps. Ce temps ne sera pas forcément le temps à vide, même si aucun agent n'emprunte cette liaison entre points d'intérêt. Il est très important de préciser que cet ajout d'agent factice n'est pas du tout obligatoire : le besoin ou non d'agents factices lors des boucles, pour calculer les trajets non-traités, dépendra de la méthode choisie de mise à jour des temps entre les boucles. Cet aspect sera traité plus en détail dans une section dédiée à cette mises à jour des temps.

5.1.4 Plusieurs réseaux routiers

Lors de l'introduction nous avons fait la liste des leviers sur lesquels nous pourrions agir. Il était fait état des localisations des commerces (que

nous avons traité dans la section 3.4.1), et des localisations de parkings (traité dans la section 3.4.2), ainsi qu'au niveau de la quantité de places (ce point sera abordé au cour du chapitre 6. Même si cela ne donnera pas lieu dans cette étude a des expérimentations numériques, il serait aussi possible d'effectuer des choix au niveau du réseau routier lui même.

Ce levier malheureusement ne peux être intégré directement au niveau des variables liées à nos liaisons. Quelles que soient les modélisations effectuées, les liaisons de nos graphes (du réseau logique) ne correspondent pas aux liaisons du réseau physique. Néanmoins, il sera possible de choisir entre différents réseaux physiques discrétisés et définir la meilleure possibilité. Si la personne chargé de l'aménagement d'une zone veut connaître, parmi plusieurs constructions sur le réseau routier, la meilleure option (simultanément aux autres leviers d'optimisation), il pourra définir les différents réseaux physiques résultant de toutes ces possibilités. Nous aurons donc à ce moment-là, toujours un seul réseau logique (en termes de noeuds et de liaisons) mais plusieurs réseaux physiques possibles. L'impact de ces différents réseaux physiques sur le réseau logique portera simplement sur le coût des liaisons de ce dernier.

Pour permettre donc aux modèles d'optimiser le réseau routier, conjointement aux autres leviers, il suffira donc de permettre l'utilisation de plusieurs matrices de coûts des liaisons, une pour chaque possibilité de réseau physique. Les coûts de liaison ne dépendront plus seulement des points d'intérêts par lesquels passer mais aussi du réseau réel. Le choix du réseau réel à choisir, parmi un ensemble de réseau réel possible, sera un nouveau choix supplémentaire à optimiser. Une variable de décision supplémentaire viendra donc se greffer à nos modélisations afin de permettre de trouver quel est le réseau physique optimal, tout en optimisant à la fois le trajet des agents, les localisations des commerces/objectifs et/ou les localisations des parkings.

Ce levier n'est vraiment pertinent que si l'on est capable d'estimer et surtout de différencier entre eux, les différents coûts entre liaisons en fonction des différents jeux de constructions possibles (réseaux physiques complets). Pour cela MatSim serait un outil essentiel. Le bouclage entre MatSim et CPLEX sera lui aussi possible à la condition, après chaque optimisation linéaire réaliser via CPLEX, de lancer non plus une mais autant de simulation qu'il y a de réseau réel (graphe pour MatSim) différent. Chacune de ces simulations fournira les coûts des liaisons pour le réseau réel correspondant.

La méthodologie permettant de prendre en compte ce levier est simple, cependant, sa mise en place est assez conséquente. En effet il faudrait générer plusieurs réseaux physiques. De plus l'intégration de cette nouvelle variable décisionnelle multipliera le nombre de variables actuellement existantes liées au trajet des agents par le nombre de réseaux à comparer. Car il faudra, pour conserver un problème linéaire, créer des variables artificielles qui réunissent les entités de choix de trajet précédemment définies avec ce nouveau choix de réseau. L'ajout de ce levier ne restera ici que théorique : aucune expérimentations numériques sur ce dernier n'a été effectué.

5.2 Expérimentations numériques

5.2.1 Simulation avec une génération aléatoire, réseau logique uniquement

De premiers tests numériques ont été réalisés avec une approche simplifiée. La principale difficulté pour réaliser ce bouclage sur un grand nombre de situations aléatoires est le réseau physique qui doit être établi pour MatSim. S'il est facile d'établir une méthode de génération aléatoire de données pour l'optimisation (fournir des réseaux logiques, à minima cohérents), cela serait beaucoup plus complexe pour le réseau physique. Dans un premier temps donc ont été réalisés des tests en considérant simplement le réseau logique en tant que réseau physique aussi.

S'il peut sembler pertinent de faire état de cette méthodologie de test, nous n'en présenterons pas de résultats numériques. En soit cette méthodologie admet beaucoup trop de différences, s'éloigne beaucoup trop de ce que l'on aura dans la réalité. Néanmoins, ce cas particulier et aussi un des cas les plus difficile à résoudre par cette méthodologie de couplage. Pour cette raison, il a été intéressant à minima de l'utiliser pour tester les différents "*réglages*" possibles que l'on pourrait faire pour le couplage.

Cette méthodologie de couplage justement qui n'a été présentée pour le moment qu'en grandes lignes. En effet, un des points les plus importants n'a pour le moment pas du tout été abordé : comment mettre à jour les temps/coûts ? La réponse pourrait sembler triviale au premier abord. On serait tenter de répondre simplement : "*en remplaçant les anciens*

coûts (en temps) de la matrice par les nouveaux temps trouvés par la simulation". Déjà, ce n'est pas la seule option possible, mais par ailleurs il s'agit probablement de l'une des pires. Dans la section suivante nous commencerons à présenter un tests plus réaliste, qui prendra en compte un réseau réel pour MatSim. C'est après cette présentation que sera abordé l'aspect de la mise à jour des coûts.

5.2.2 Étude d'une situation fictive basée sur une zone réelle, avec réseau logique et réseau physique

L'objectif ici est principalement de réaliser un couplage entre CPLEX et MatSim sur un exemple réaliste, et pas obligatoirement réel. Tout en se basant sur une zone d'étude qui elle sera réelle, certaines de nos données seront donc inventées afin de permettre de faciliter l'étape d'acquisition des données pour la réalisation des tests qui vont suivre.

Certains éléments de la situation par contre ont pu être ajusté légèrement pour une raison bien précise. Nous souhaitons en particulier atteindre une augmentation des temps liée à la congestion supérieure à 100%. Cela signifie que, par rapport aux temps de déplacement des itinéraires prévus, si les routes étaient vides, le temps de déplacement trouvé par simulation sera au minimum le double. Pourquoi cette valeur ? Parce que au dessus de 100% d'augmentation de temps liée aux congestions, plus de la moitié du temps de trajet n'est plus déductible depuis notre simple optimisation mais ne pourra être pris en considération que par le bouclage entre CPLEX et MatSim. Il s'agit donc d'un bon palier de test pour la méthode afin que le résultat théorique à vide ne viennent pas prédominer (et donc engendrer un biais) sur la tentative d'optimisation au niveau de la congestion, à l'aide de ce bouclage. Au plus les congestions seront faibles, au moins le bouclage n'aura d'intérêt. Si les congestions sont très faibles, la première optimisation trouvera même directement une solution stable qui ne sera jamais remise en question boucle après boucle.

Voici comment notre situation de test est construite :

- La zone géographique de ce test est une partie de la zone commerciale du Pontet, plus précisément celle autour de l'*Avenue Marc Lepoutre*. Cette zone a été choisie car disposant d'un nombre assez

importants d'espaces non occupé (sans construction), d'un réseau routier qui encadre déjà bien ces zones, et de quelques magasins et parkings déjà existants qui seront considérés comme fixes. Aucune vérification n'a été faites sur l'état administratif des espaces libres que l'on considérera comme disponible pour ce test.

- 4 boutiques existant dans cette zone seront des objectifs fixes (non-relocalisables) pour le test.
- 4 enseignes fictives veulent implanter chacune un magasin sur cette zone.
- 9 emplacements fictifs possibles sont proposés pour les 4 nouveaux commerces. Les tailles de magasin sont ignorées, chaque commerce prend un des emplacements à lui seul.
- 2 parkings existants sont considérés, ceux qui desservent les 4 magasins présents sur la zones. Leur localisation respective est simplifié par un point au centre de chacun.
- 7 nouveaux emplacements de parkings sont proposés pour permettre de correctement desservir les nouveaux commerces.
- 10 points d'intersection (à partir desquels on peut partir dans 3 directions différentes sur le réseau routier) permettent de constituer notre réseau physique. 4 d'entre eux sont déjà des points d'intersection présents dans le réseau physique réel de la zone, les 6 autres permettent d'atteindre les nouveaux parkings.
- Le réseau physique s'appuie sur l'existant. Les axes routiers internes à des zones de parkings sont simplifiés en un seul point de parking accessible depuis plusieurs noeuds du graphe routier, l'unique rond point présent dans la zone devient simplement un point d'intersection. Différents accès aux parkings fictifs sont ajoutés (entre 2 et 4 accès par parkings, depuis un noeuds d'intersection ou un autre parking).
- Les entrées/sorties ne sont pas dissociées (il n'y a pas de sens uniques dans cette zone) et correspondent exactement aux 3 points d'accès dans le polygone considéré. Il est à noter que deux de ces accès se rejoignent un peu plus loin hors de ce polygone, il serait possible de prendre en considération ce chemin disponible entre les 2 ou ne considérer qu'un point. Cependant le réseau physique routier imaginé propose aussi un chemin, encore plus rapide, à l'intérieur de la zone.
- Ce test n'aura pas de limite en termes d'utilisation de parking.

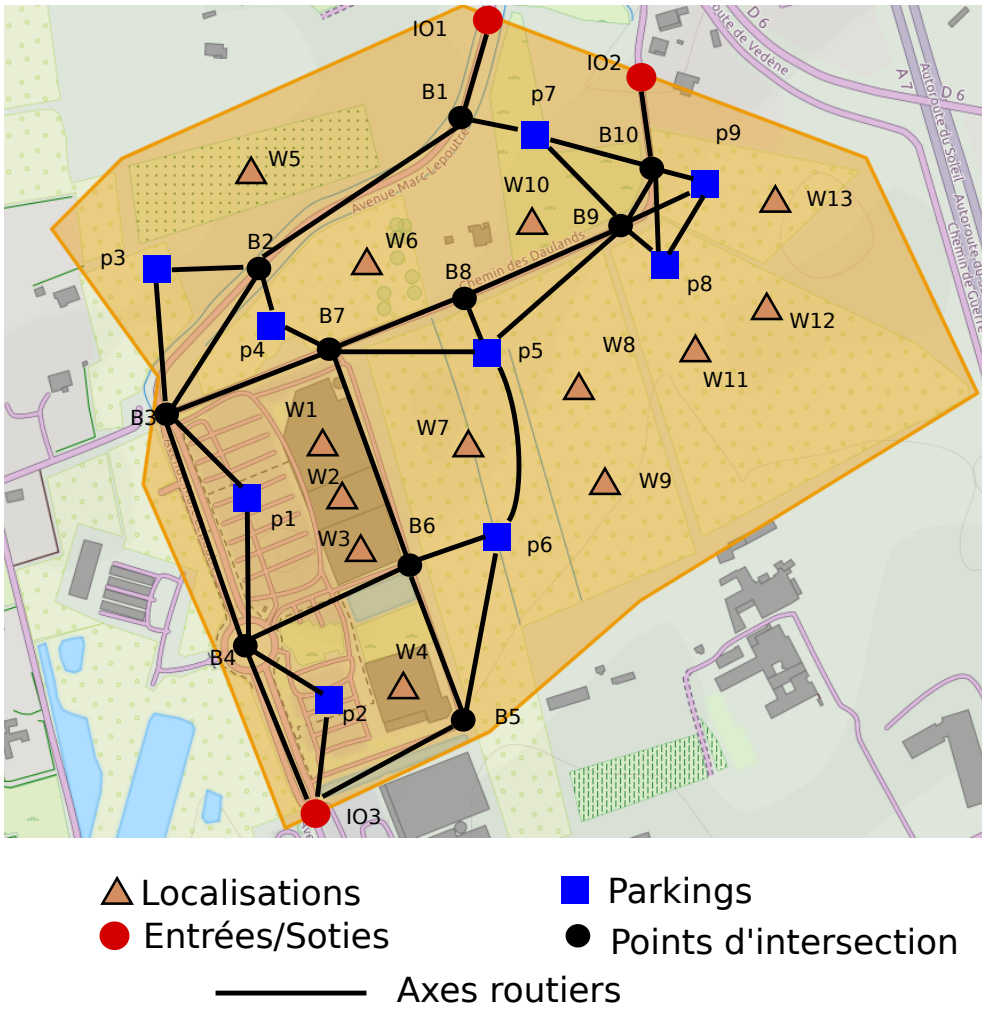


FIGURE 5.2 – Réseau physique routier imaginé pour la zone d'étude.

Un élément particulièrement important est à noter : le nombre de parking pouvant exister est restreint. Seulement 2,3 ou 4 parkings peuvent être choisis parmi les 7 nouveaux emplacements proposés (pour un total respectivement de 4,5 ou 6 parkings, en considérant les 2 parkings fixes déjà existants). Les 3 cas seront testés indépendamment, mais avec le même jeu d'agent, afin d'observer les variations de temps liées au nombre de parkings construits

La figure 5.2 montre une carte de notre zone d'étude, délimitée par le polygone orange.

Nous avons donc à résoudre un problème de type MA-TSP-PRL-DL (avec des relocalisations à la fois de commerces et de parkings), pour lequel sera testée une méthode de couplage CPLEX-MatSim.

Concernant les agents de ce test :

- Les objectifs ciblés des agents restent aléatoires mais ont maintenant des probabilités non homogènes suivant les objectifs, pour augmenter un peu les écarts de coût total entre un bon et un mauvais placement des infrastructures, tout en étant plus réaliste qu'une très forte homogénéité sur la quantité de visiteurs pour chaque commerce. Du commerce 1 à 4 (fixés) les probabilités que cet objectif soit sélectionné au moment d'un choix d'objectif sont respectivement de 5%,10%,15% et 20%. De même pour les commerces 5 à 8 (à localiser), dont les probabilités respectives sont aussi de 5%,10%,15% et 20%.
- Les entrées/sorties sélectionnées sont aléatoires de manière homogène.
- Le nombre d'objectifs par agent cette fois-ci ne sera plus fixe. Une probabilité à chaque agent sera choisie pour plusieurs nombres d'objectifs possibles : 75% de n'avoir qu'un objectif, 20% d'en avoir 2, 5% d'en avoir 3.
- Afin de faciliter les optimisations à réaliser, la situation n'est parcourue que par 741 agents, ce qui correspond, par rapport à nos proportions choisies en nombre d'objectifs, à environ 1.000 visites effectuées. Cependant, on considère que chacun de ces agents représentent en réalité 10 personnes physiques ayant les mêmes propriétés. On considère donc que dans la réalité nous avons 7410 personnes, qui réaliseront un total de 10.000 visites, et qui seront représentés dans nos modélisations par 741 agents.
- les capacités routières (qui par défaut sont à 500 usagers par heures) ont été réduite à 480 usagers par heure, avant apparition de ralentissements. Le léger ajustement de cette valeur a pour principal but de veiller à avoir pour tous nos tests une augmentation de temps liées à la congestion supérieur au total de temps à vide lui-même. La vitesse des axes routiers est de 13.8 mètres par seconde soit 50km/h.
- Les temps de début de parcours, pour MatSim, sont aléatoirement répartis sur une période de 6 heures.

5.2.3 Mise à jour des temps

Au cours de la section 5.1.2 il est question, suite au calcul des temps par MatSim, de mettre à jour la matrice des temps pour CPLEX. Cette mise à jour reste néanmoins quelque chose de paramétrable. En effet, si

les premiers tests ont été réalisé en considérant une mise à jour directe (remplacer directement l'intégralité de la matrice des coûts par la nouvelle matrice calculée), il s'avère qu'en cas de forte congestion ce n'est pas du tout une méthode efficace.

Plusieurs méthodes de mises à jour ont été testée, principalement sur des situations aléatoires mais aussi pour les plus pertinentes sur notre situation du pontet :

— Mise à jour "directe" des temps : Les temps trouvés par la dernière simulation via MatSim viennent directement remplacer les précédents temps, boucle après boucle.

— Mise à jour par les temps moyens entre les temps précédents et les nouveaux : Il s'agit simplement, après simulation via MatSim, de remplacer les temps de la matrice des coûts par la moyenne pour chaque liaison entre le nouveau temps calculé par simulation et le précédent, issu de la matrice ayant servie lors de la dernière optimisation linéaire.

— Mise à jour par les temps moyens entre toutes les simulations déjà réalisée : Même principe que la méthode précédente mais cette fois-ci il faut conserver chaque résultat de simulation MatSim, boucle après boucle, et de mettre à jour la matrice de coûts avec les moyennes des temps trouvés pour toutes les simulations.

— Conservation du "pire" temps pour chaque liaison : Lors de la mise à jour des temps, après simulation MatSim, on ne met à jour le coût d'une liaison uniquement si le temps de la dernière simulation est plus élevé que le précédent. On conserve de cette manière uniquement la pire congestion qui peut avoir lieu sur cet axe.

Après de longues séries de tests, il a été conclu que pour les jeux de données aléatoires ainsi que pour la zone du pontet, la méthodologie de mise à jour des temps la plus efficace est celle de conservation du pire temps. Cela peut s'expliquer pour la raison qu'il s'agit de la seule de ces 4 méthodes qui garde réellement de manière définitive des informations de temps qui dissuade fortement de s'approcher de solutions déjà testées donnant lieu à un très fort trafic.

Cette mise à jour en l'état n'est pas non plus parfaite. De très forte congestions dans certaines solutions testées peuvent donné lieu, pour certains axes, à une augmentation de temps brutal et totalement punitive pour les prochaines boucles. C'est à dire, si une solution testée demande

à tous les agents d'emprunter un même axe, l'augmentation de temps pour cet axe pourrait être multiplié d'un coup par 100 ou 1.000, ce qui empêchera totalement toutes les optimisations via CPLEX qui suivront de choisir cet axe. Alors même qu'une utilisation moindre de ce dernier par un meilleur agencement donnerait de bonnes solutions.

Pour éviter ce problème, qui a été constaté lors de nos tests, il est conseillé de limiter l'impact de cette mise à jour du pire temps par un seuil maximal d'augmentation possible lors de la mise à jour. Par exemple, pour les tests qui seront présentés dans la section suivante, il a été décidé d'empêcher la mise à jour du pire temps de dépasser le double du temps précédent. C'est à dire, si le temps précédent d'une liaison était de "10" (sans unité ici), et que la nouvelle simulation MatSim trouve un temps de "100", ce temps sera mis à jour par "20" (le double de "10") à la place. Cette progression plus douce commencera donc par dissuader l'optimisation linéaire une utilisation trop importante de cet axe, sans être aussi punitive que précédemment. Au plus la progression de ces temps sera douce, au moins il y aura de risques de rater une bonne solution lors des boucles causée par une trop forte augmentation de temps sur un axe intéressant. Cependant, en contrepartie, le nombre de boucles nécessaires sera beaucoup plus important avant d'atteindre une situation stabilisée (qui ne change plus pour les boucles suivantes).

Un autre point est très important à souligner. Grâce à cette méthodologie du pire temps, il est possible (et cela a été fait pour les tests qui sont présentés) de ne plus se servir des agents factices lors des boucles. Des agents factices sont utilisés pour la phase d'initialisation des temps à vide (une simulation qui ne contient que des agents factices), mais n'ont plus vraiment d'utilité lors des boucles entre CPLEX et MatSim. Ils sont nécessaires pour toutes les autres méthodes décrites mais sont aussi une source de bruits. En effet ces agents factices eux-mêmes peuvent aggraver une congestion présente, et donc augmenter les temps par leur présence. Le fait de pouvoir se passer des agents factices pour la méthode de récupération des pires temps permet donc en plus de réduire le bruit dû à ces derniers.

Les expérimentations numériques sont présentés juste après la section suivante, qui elle aborde un dernier point, celui de l'utilisation d'une méthode de simplification pour l'optimisation linéaire.

5.2.4 Simplification

Les tests réalisés pour cette zone d'étude du Pontet ont dans un premier temps été réalisé sans aucune simplification (section 4.1.1). La simplification par pré-calcul des sous-trajets pertinents n'ayant, à cet instant là, pas était développée pour les problème de type MA-TSP-PRL-DL (avec un nombre limité de parkings) comme dans le cas présent.

Les optimisations linéaires pour cette situation, qui semblent à l'oeil être assez simples, sont en réalité d'une très grande complexité. D'une part la quantité de données présentes dans les matrices des coûts est très importante mais la quantité de contraintes qui seront générées par OPL à partir de ces données, pour représenter le problème, l'est encore plus. Le bouclage entre nos deux outils pour résoudre ce MA-TSP-PRL-DL en considérant un nombre de parking maximal de 5 à pu être réalisé sans simplification, mais demandaient à chaque optimisation une capacité en mémoire vive (RAM) de l'ordre de 100 Go pour un temps d'exécution autour de 5 heures sur des serveurs disposant de 48 coeurs et d'une fréquence CPU de 1.3 GHz. Le temps d'exécution peut certes changer suivant la machine utilisée pour la résolution, mais le besoin de cette quantité de mémoire vive reste très contraignant. Lors du passage au même problème avec seulement 4 parkings, l'une des boucles est devenu même impossible à réaliser sur nos serveurs pour cause de saturation en mémoire vive (plus de 250Go de RAM, après 10 heures d'exécution).

Pour palier à ce problème donc, une méthode équivalente à la simplification par pré-calcul des sous-trajets pertinents a été développée et testée pour être fonctionnelle dans un cas de MA-TSP-PRL-DL. Les résolutions qui demandait, sans simplification, une quantité de RAM de plus de 100 Go (voir plus de 200Go parfois) pour des temps d'exécution variant entre 5 et 10 heures, ne demande plus que, après simplification, environ 3.5Go de mémoire vive pour un temps d'exécution autour de 5 minutes (sur une machine d'une fréquence CPU de 1.3 GHz).

On constate donc par cette expérience sur une situation proche de la réalité l'importance de la simplification pour la résolution du problème. Le temps de résolution du problème MA-TSP-PRL-DL pour ce cas précis s'est vu divisé par plus de 100 en moyenne, et la quantité de mémoire nécessaire a été divisée par plus de 25.

La section suivante présentera donc les tests pour la résolution du problème détaillé dans la section 5.2.2, avec la méthode de mise à jour

présentée dans la section 5.2.3, et à l'aide de cette simplification exacte consistant à éliminer les sous-trajets non-pertinents.

5.2.5 Résultats pour la zone d'étude du Pontet

Les expériences ont été menés sur un même jeu d'agent, tiré aléatoirement selon les proportions fixées. Jeu qui reste identique pour chacun des trois tests. Ces trois tests correspondent à l'optimisation du placement des commerces à localiser ainsi que des parkings à construire, pour un nombre de parkings maximal de 6, 5 et enfin 4. Ce nombre total de parkings, en comptant les 2 parkings fixes déjà existant, impose au problème de trouver parmi les 7 emplacements possibles l'emplacement optimal pour 4, 3 et enfin seulement 2 parkings constructibles.

Les résultats numériques boucle après boucle de ces expériences sont répertoriés dans l'annexe C.5. On y constate principalement que pour cette zone d'étude, la première solution trouvée avant bouclage reste assez performante. Ceci s'explique simplement par le fait que, malgré une forte congestion générale, artificiellement créée, il ne se créé pas sur ces solutions de fortes congestions locales qui perturberait le trafic plus que la normal en considérant le nombre d'agent et les capacités des routes. L'utilisation de ces routes est à peu près en équilibre par rapport au trafic qu'on leur demande de supporter.

Les premières boucles (à partir de la 2ème ou 3ème) donnent parfois des résultats très élevés. C'est ce que l'on pourrait appeler la phase de réglage. La première (et parfois la deuxième) résolution CPLEX donne des résultats qui vont engendrer, par simulation, de forts trafics sur les axes sensés théoriquement être optimaux (sans prendre en compte les congestions). Les temps sur tout ces axes vont donc fortement être augmentés pour les optimisations suivantes, et l'optimisation va donc les éviter, en cherchant d'autres trajets possibles. Ce n'est que petit à petit, après avoir consécutivement mis à jour et augmenté les temps d'à peu près tous les chemins possibles, que le bouclage pourra faire ressortir les solutions intéressantes. Ceci tout en évitant les déplacements aux endroits où il aura été noté les plus forts coûts prenant en compte les congestions. Il n'y a pas de boucle prédéfinie à partir de laquelle cette méthodologie est sensée finir cette phase de réglage pour aller chercher des solutions qui ont une bonne connaissance des risques de congestion. D'autant plus que la méthodologie testée prend en compte une limite d'augmentation de coût

entre deux boucles (qui a été fixée ici au double).

Enfin, après plusieurs itérations, la méthodologie converge vers une solution finalement stable qui ne mettra plus à jour les coûts de la solution trouvée. Cette dernière solution stable trouvée n'est cependant pas assurée d'être la meilleure solution. La meilleure solution sera généralement présente : soit à la première (ou deuxième) boucle, car il est possible si le trafic sur le graphe de la zone étudiée soit réparti de manière bien homogène (et donc ne nécessite pas de tenir compte du trafic finalement) ; soit après la phase de réglage, dans les dernières boucles, où une grande partie des choix à ne pas faire a laissé une emprunte au niveau de la matrice des coûts, et où donc les optimisations sont faites avec une vue d'ensemble des congestions potentielles.

Dans les trois tests réalisés (6, 5 et 4 parkings autorisés) le meilleur résultat est toujours trouvé dans les boucles proches de la fin (après la phase de réglage). On pourrait noter cependant que la première optimisation trouvée reste toujours assez proche de ce résultat. Les placements de parkings et commerces des meilleurs solutions trouvées pour chacun de ces tests sont répertoriés dans le tableau 5.1 suivant :

TABLE 5.1 – Solutions retenues en fonction du nombre de parking

	CPLEX		MatSim		
Nombre de parking	Affectations D1 à D8	Parkings utilisés	Temps Total	Temps à vide	Augmentation des temps
6	D5 :W8 D6 :W11 D7 :W7 D8 :W10	P1 :214 P2 :210, P5 :118, P6 :111 P7 :190 P8 :104	566.657	277.379	+104%
5	D5 :W8 D6 :W11 D7 :W7 D8 :W10	P1 :238, P2 :207, P5 :215 P7 :177 P8 :110	640.192	282.053	+126%
4	D5 :W13 D6 :W12 D7 :W11 D8 :W10	P1 :212, P2 :277, P :182 P :276	691.842	291.413	+137%

5.3 Conclusions

La congestion du trafic, engendrant des perturbations sur les temps de déplacement, peut donc bien être prise en compte en associant des méthodes d'optimisation et de simulation. Cette association d'outils reste cependant complexe. En particulier, les choix effectués lors des mises à jours de la matrice des coûts après simulation sont cruciaux afin que

cette association puisse être de qualité. Parmi les nombreuses possibilités, les méthodes gravitant autour d'une prise en compte des pires temps semblent donner de bons résultats. Cependant ce n'est probablement pas la seule à être efficace et la recherche de meilleurs options reste ouverte à ce niveau.

Cette méthode n'est évidement pas exacte, dans le sens où il peut exister des solutions meilleures qui ne seront jamais parcouru dans le processus. Néanmoins, elle permet d'améliorer le résultat en simulation - en prenant en compte le trafic - d'une optimisation théorique qui ne prend pas en compte le trafic. En particulier lorsque l'optimisation théorique, sans trafic, engendrerait une forte congestion très locale et non répartie sur le graphe de manière homogène.

On a pu constater aussi dans un exemple réaliste des augmentations de temps si l'on cherche à réduire le nombre de parking. Il faudra voir si ces augmentations de temps (entre 8 et 13% dans l'exemple du pontet, de 6 à 5 zones ou de 5 à 4 zones) font économiser en contre-partie une quantité de places de stationnement significative. L'estimation du nombre de places de parking nécessaires en connaissance d'une utilisation est justement l'objet du chapitre suivant.

Chapitre 6

Évaluation de la quantité de places nécessaires pour un parking

Sommaire

6.1 Explications

Il a été question sur certains de nos modèles d'optimiser les choix des positions des parkings, ainsi que d'en limiter leur utilisations. Grâce à ces modèles nous pouvons en déduire, d'après le résultat, le nombre d'utilisations de ces parkings qui ont été réalisées en simulation, pour la situation optimale. L'objectif de cette section sera de chercher à obtenir une information utile à partir de ce nombre d'utilisation : le nombre de places nécessaires.

Suivant le contexte il y aurait 2 scénarios possibles :

- Si les agents et leur horaire d'entrée dans le système ainsi que les coûts de déplacement sont très précis et fixes, c'est à dire qu'ils seront toujours fidèles aux données : dans ce cas on peut directement trouver le nombre de places nécessaires d'après la simulation MatSim en regardant combien de personnes simultanément étaient présentes sur le parking. Ce n'est pas notre cas.
- Si en revanche les données sont issues de l'estimation d'un cas moyen, qui dans la réalité est très aléatoire : les résultats d'arrivée au niveau des parkings de la simulation MatSim ne seront pas vraiment exploitables. C'est pour ce deuxième cas qu'une méthode est présentée dans ce chapitre.

Notre cas d'étude est le déplacement dans une zone commerciale de clients. Il n'est donc pas vraiment question d'optimiser le nombre de places nécessaires pour une journée précise avec des heures d'arrivée fixes sur le parking, mais plutôt de savoir combien de places seront nécessaires pour des utilisations du parkings similaires à celles obtenues.

6.2 Phénomène très aléatoire, résolution par l'aléatoire

Nous admettrons ici que le nombre d'agents, donc en fait le nombre d'utilisations du parking, est une donnée fixe, ainsi que leur temps d'utilisation du parking (propre à chaque agent). Les heures exactes d'arrivée seront ici la seule donnée que nous ferons varier. Néanmoins, la méthode qui sera vue, dans la section suivante, sera très facilement adaptable si l'on souhaite aussi pouvoir faire varier le nombre d'agents et/ou leur temps d'utilisation.

Dans l'absolue, pour que chaque agent puisse utiliser le parking la solution est simple : il faudrait autant de places de parking qu'il y a d'utilisations. Le pire cas étant celui où il existe un temps durant lequel tous les agents utilisent ce parking.

Ce qui serait plus intéressant par contre c'est de connaître le risque pris si on retire des places, et qu'il y ait alors moins de place de parkings que d'agents les utilisant. Prenons un exemple concret : sur un parking, 100 personnes vont venir se garer sur un créneau d'une après-midi (de 14h à 18h, pour l'exemple). Quelles sont les chances que tout le monde

réussisse à se garer (sans attendre) si le parking ne dispose que de 50 places ? C'est le type de question que nous nous poserons.

Pour répondre à cette question, qui porte sur une donnée (l'agent) qui sera très aléatoire dans la réalité, nous allons prendre en compte ce côté aléatoire. On génère autant de cas que possible d'après des règles aléatoires que l'on aura fixées, et on regarde combien de places sont nécessaires pour chacun de ces cas. Suite à cela nous pourrons établir la probabilité que tout le monde puisse se garer, pour chaque quantité possible de place.

La résolution de ce problème par une génération élevée de cas aléatoires pour en établir une règle moyenne pourrait sembler conséquente. Ceci étant, le calcul à réaliser pour chaque génération aléatoire est si faible que l'on peut atteindre des seuils de confiance relativement élevés. On peut générer sans souci 10.000 cas aléatoires et en traiter les probabilités en moins d'une minute même pour un grand nombre d'agents.

6.3 Méthodologie

En entrée nous aurons un fichier XML contenant les données suivantes :

- une liste de type d'agents, définissant le temps de stationnement et le nombre d'agents ayant ce temps de stationnement.
- une liste de probabilités des temps d'arrivée, probabilités qui seront définies par un temps et un poids. Le poids fera office de probabilité. La probabilité qu'un temps d'arrivée soit choisi sera son poids divisé par la somme de poids de tous les temps d'arrivée.
- les deux éléments précédemment vus seront propres à chaque parking. Il est possible de renseigner plusieurs parkings afin de tous les traiter. Néanmoins le traitement de chacun se fera de manière indépendante.

Voici un exemple de fichier d'entrée :

```
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <parkings>
```

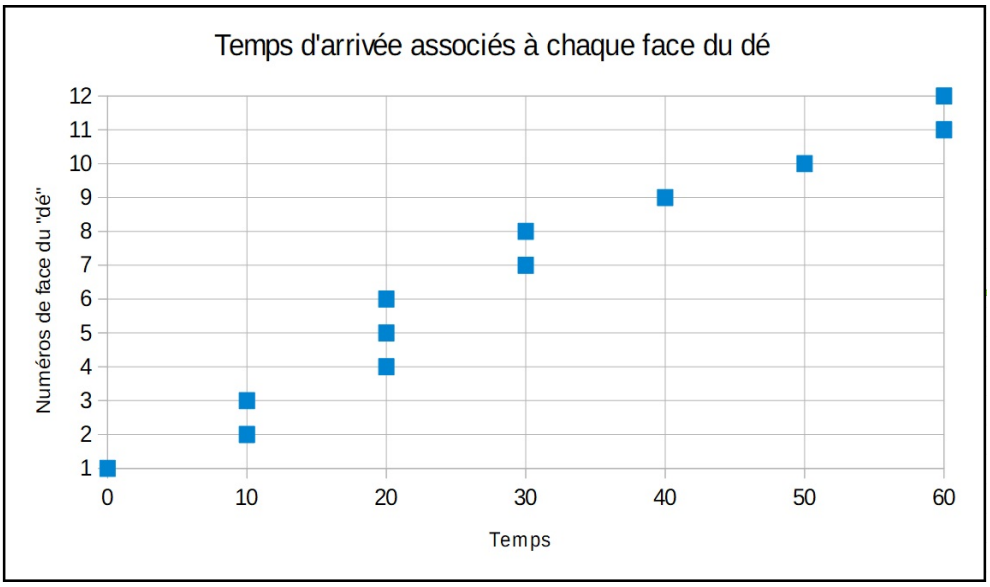


FIGURE 6.1 – Affectation des faces du dé.

```
4       <parking id="1" >
5           <user period="10" nb="15"/>
6           <user period="20" nb="32" />
7           <rule>
8               <prule time="0" probability="1" />
9               <prule time="10" probability="2" />
10              <prule time="20" probability="3" />
11              <prule time="30" probability="2" />
12              <prule time="40" probability="1" />
13              <prule time="50" probability="1" />
14              <prule time="60" probability="2" />
15          </rule>
16      </parking>
17  </parkings>
```

La méthode à réaliser étant similaire pour chaque parking on se limitera ici à regarder ce qu'il se passe pour un seul parking.

Étape 1 : Création du "dé"

On va commencer par transformer les probabilités d'arrivée en un dé. Cela facilitera les tirages aléatoires de temps d'arrivée par la suite. Un dé (liste ou tableau) composé d'autant d'éléments que la somme des poids des $< prule >$ (on peut considérer ces éléments comme les faces de notre dé).

La figure 6.1 illustre l'affectation des faces du dé d'après le fichier d'entrée que nous avons vu en exemple.

Pour les simulations qui vont suivre on utilise une classe que l'on a appelé *PlaceProba* pour stocker les résultats que l'on obtiendra au fil des

simulations. Les simulations qui vont suivre à l'étape 2, nous indiqueront un nombre de places nécessaires afin que chaque agent se gare, d'après des temps d'arrivée tirés au dé. Il y aura un *PlaceProba* pour chaque nombre de places nécessaires différents que l'on obtiendra. Cette classe a pour attributs :

- *nb* : le nombre de places de stationnement nécessaires associé, unique à chaque *PlaceProba* ;
- *count* : le nombre de fois où les simulation ont nécessité exactement ce nombre de places, c'est à dire où il y aura eu exactement *nb* agent(s) présents simultanément sur ce parking à l'instant où il y avait le plus d'agents présents lors d'une simulation ;
- *cumul* : le nombre de fois où nos simulations seraient satisfaites par ce nombre de places. C'est à dire où il y a eu un nombre inférieur ou égal à *nb* d'agent(s) présents simultanément sur ce parking à l'instant où il y avait le plus d'agents présents lors d'une simulation. Il s'agit donc du cumul des *count* de toutes les *PlaceProba* dont le *nb* est inférieur ou égal à celui-ci ;
- *cumul_pourcent* : on se servira du *cumul* et du total de simulations effectuées pour stocker ici le pourcentage de chance de succès (pourcentage de simulations où tous les agents pourraient se garer avec ce nombre de places).

Étape 2 : Les simulations

Maintenant que nous avons notre dé, ainsi que la classe dans laquelle stocker les résultats des simulations, il ne reste plus qu'à lancer les simulations proprement dites. Une liste de tous les temps comptera le nombre d'agents présents à chaque temps d'arrivée possible. On tire au sort le temps de d'arrivée des agents, un par un. On peut voir un exemple plus détaillé de ce tirage d'après la figure 6.2, ainsi qu'un exemple complet sur la figure 6.3. Pour chaque agent dont on vient de tirer le temps d'arrivée sur le parking, on va incrémenter le compteur d'agents en chaque temps compris entre son arrivée et son départ du parking (temps de départ = temps d'arrivée + durée de stationnement).

On récupère en sortie le nombre de places que nécessiterait au minimum cette simulation, à savoir le nombre maximum d'agents qui étaient présents au même moment durant la simulation. Cette récupération du maximum d'agents présents en un même instant est illustré sur la figure 6.4. On incrémentera donc ensuite le compteur *count* de la *PlaceProba* liée à ce nombre de places. C'est à dire que l'on comptabilisera cette simu-

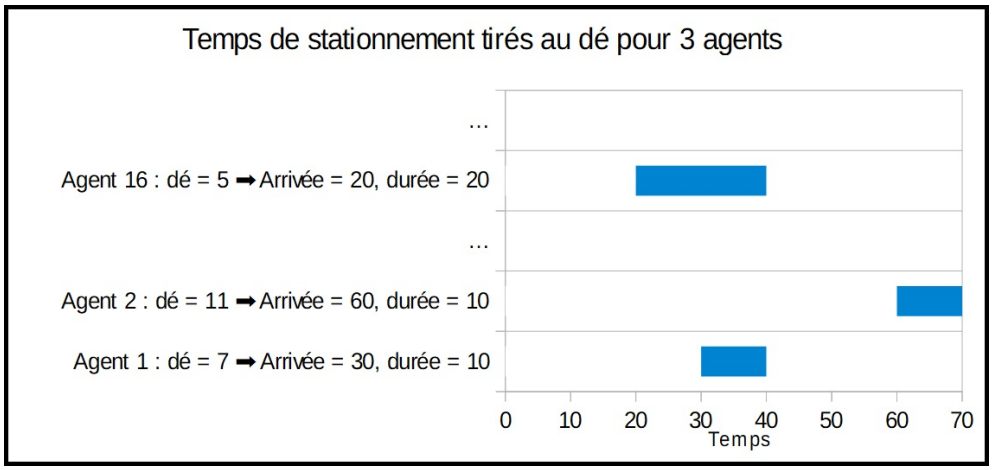


FIGURE 6.2 – Exemple de tirage aléatoire de 3 agents, avec détails de leurs informations.

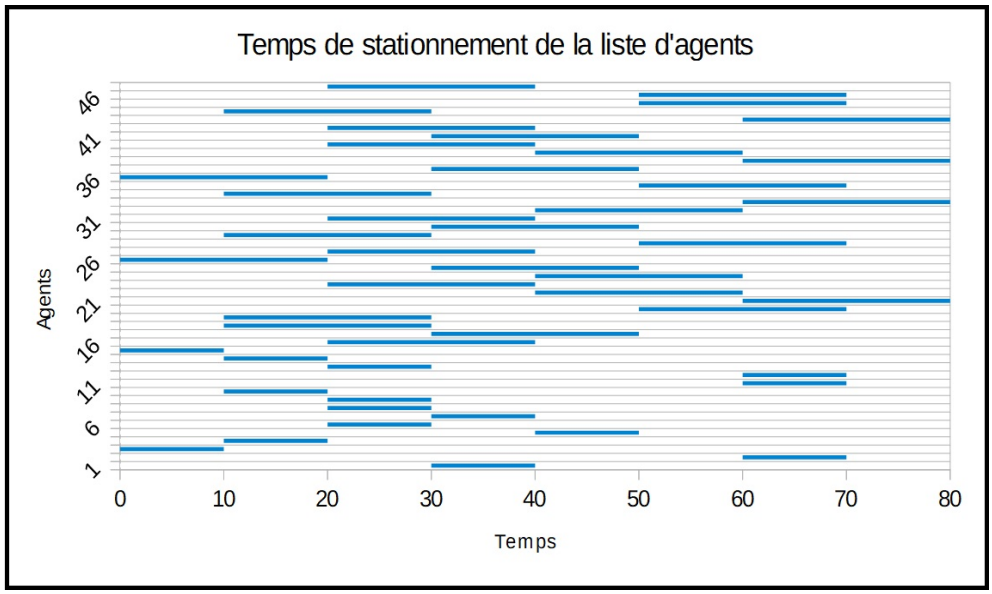


FIGURE 6.3 – Exemple de tirage aléatoire complet d'une simulation (47 agents).

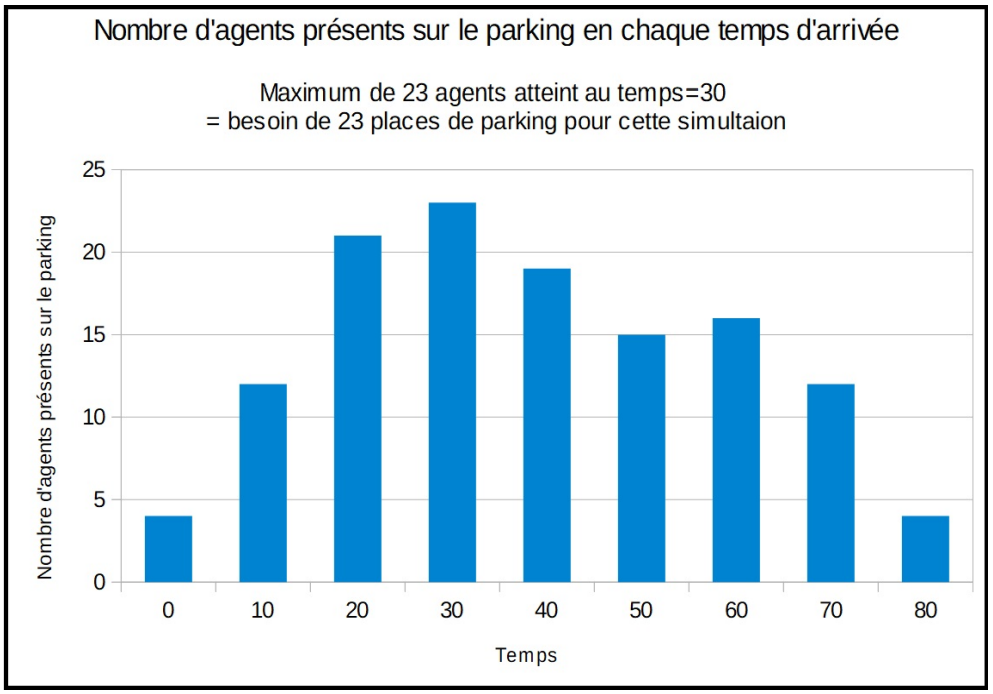


FIGURE 6.4 – Sommes des agents présents en chaque temps, pour une seule simulation.

lation dans le total des simulations ayant donné exactement ce résultat en nombre de places de stationnement nécessaires.

Étape 3 : Les résultats

Ce type de simulation doit s'effectuer le plus de fois possible pour gagner en précision. Une fois suffisamment de simulations effectuées (le nombre de simulation à effectuer est fixé à l'avance), il ne reste qu'à calculer les *cumul* et *cumul_percent* des *PlaceProba*. Nous allons traiter les *count* des *PlaceProba* (triés par nombre de places nécessaires) par ordre croissant afin de calculer pour chacune le total cumulé. Par exemple, le *cumul* du *PlaceProba* pour un nombre de 27 places de parkings sera la somme de toutes les simulations ayant donné un résultat de nombre de places nécessaires inférieur ou égale à 27. Donc, le cumul de tous les *count* des *PlaceProba* que l'on aura vus lorsque l'on traitera la *PlaceProba* associée au nombre de place 27. Le résultat le plus important à récupérer est les *cumul_percent*. Ce dernier est simplement calculé en divisant le *cumul* par le nombre total de simulations.

100.000 simulations sur le fichier d'entrées du début de cette section ont été effectuées. Le résultat, sorti initialement au format XML, est synthétisé ici dans le tableau 6.1 pour plus de lisibilité.

Nb place	Compteur de cas trouvé	Cumul	Cumul/Total (%)
25	12834	50840	50,84
26	12757	63597	63,597
27	10978	74575	74,575
28	8988	83563	83,563
29	6608	90171	90,171
30	4401	94572	94,572
31	2783	97355	97,355
32	1415	98770	98,77
33	738	99508	99,508
34	310	99818	99,818
35	112	99930	99,93
36	48	99978	99,978
37	15	99993	99,993
38	5	99998	99,998
39	1	99999	99,999
40	1	100000	100

TABLE 6.1 – Résultats de la simulation des utilisations d'un parking

Définition 13. *Le* ***taux de succès*** *est le pourcentage de chance de réussite d'un nombre de places de parkings, pour un jeu de données d'agents. Ce nombre de places admettra une réussite si tous les agents peuvent s'y garer, sans attente. Inversement, le* ***taux d'échec*** *sera le pourcentage de chance que le nombre de places ne suffise pas a admettre une réussite (au moins un agent n'a pas pu se garer).*

La figure 6.5 montre la courbe de progression du taux de succès en fonction de l'évolution du nombre de place. La figure 6.6 insiste sur les nombres de places donnant un résultat proche de 100% afin de pouvoir en discerner aussi la progression.

Les résultats dont le taux d'échec serait supérieur à 50% ont été volontairement ignorés (peu utile) On peut voir, par ces résultats, que pour les 47 utilisations de ce parking fictif (ici sur une période de temps très courte) 35 places de parkings suffiraient pour permettre à chacun de se garer sans aucune attente avec 0.1% de taux d'échec. Bien évidement, cet exemple n'est que fictif, ces résultats ne servent qu'à mettre en image le type d'information que l'on pourrait retirer par cette méthode. Le temps d'exécution quant à lui est relativement négligeable, dès lors que l'on reste dans des conditions réalistes en nombre d'utilisation, même pour un grand nombre de simulation. La plus grande partie du processus est bien évidement le tirage aléatoire d'un nombre et l'addition donnant le

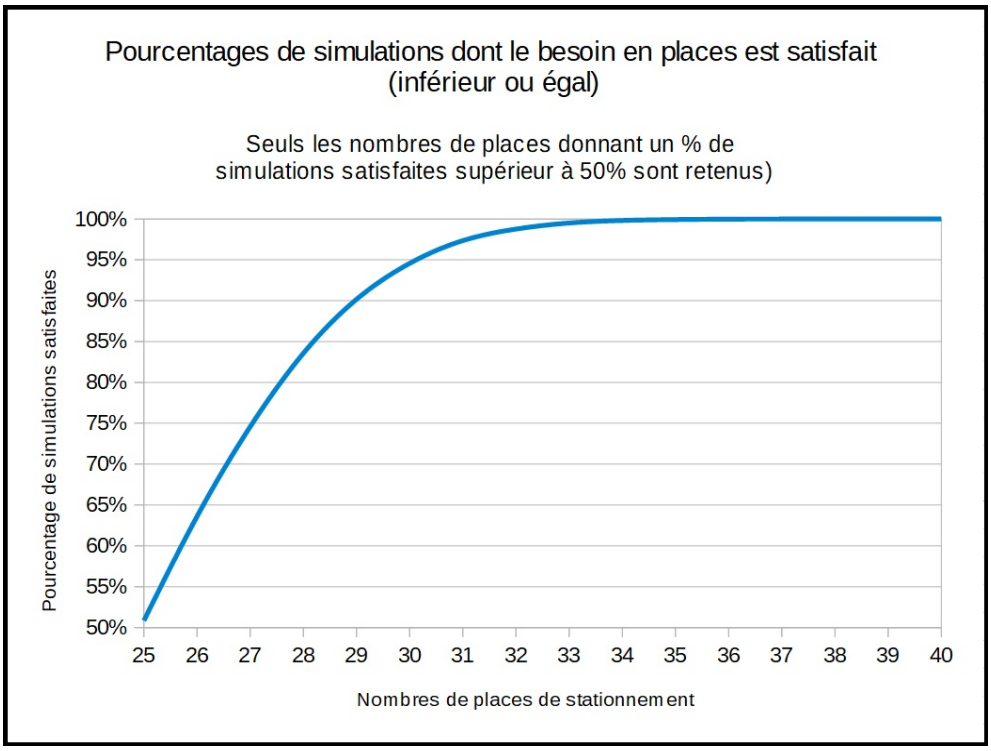


FIGURE 6.5 – Taux de succès en fonction du nombre de places de stationnement.

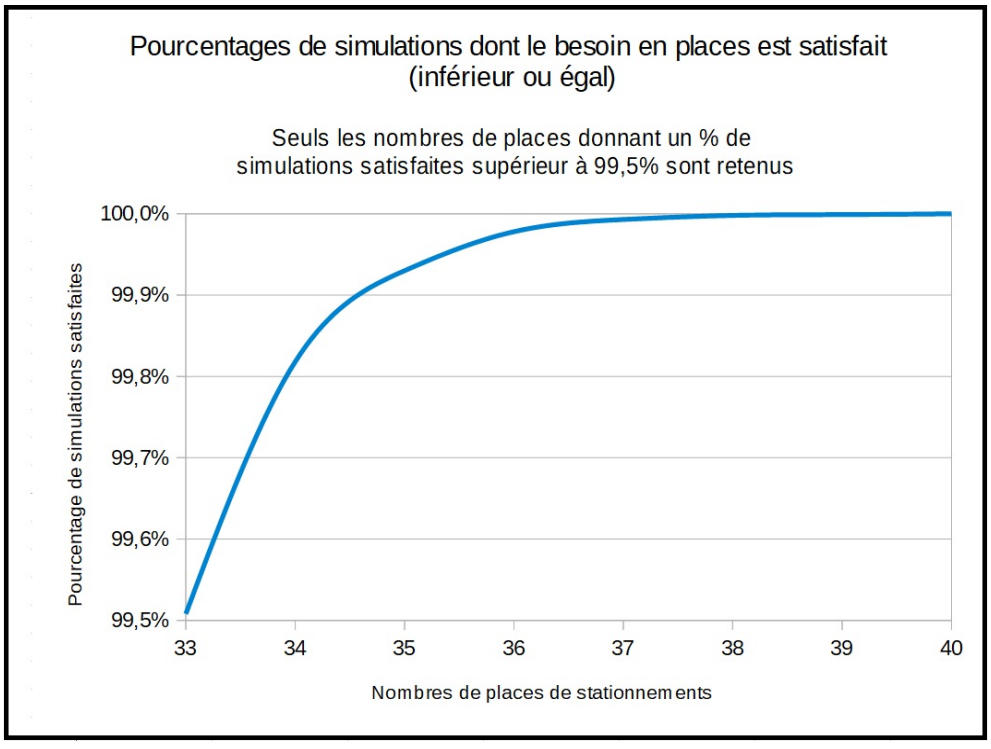


FIGURE 6.6 – Taux de succès en fonction du nombre de places de stationnement (réduit aux taux de succès supérieurs à 99.5%).

créneau de stationnement, réalisé une très grand nombre de fois (nombre de simulations * nombre d'agents).

6.4 Exploitation des résultats

Nous nous retrouvons donc avec, pour résultat, une liste de nombre de places de parkings, et le taux de succès pour chaque quantité de places. Deux utilisations intéressantes peuvent être faites de ces données.

La première, la plus évidente : connaître combien de places de parking sont nécessaires au minimum pour que chaque utilisateur trouve une place, à un taux d'échec près. On décide de considérer un certain pourcentage d'échec comme négligeable, et ainsi regarder à partir de quelle quantité de places on admet un pourcentage d'échec qui lui est inférieur. Par exemple, si avec 50 places je permets sur un parking 100 utilisations (par rapport aux autres données fournies) je n'ai que 0.049% d'échec (situations où au moins une personne n'a pas pu se garer), ce pourcentage peut être considéré comme négligeable. On peut alors préférer ne construire que 50 places pour ce parking au lieu des 100 places, seule solution qui ne peut réellement pas avoir d'échec.

Outre le fait de pouvoir se garer ou non, on regarde parfois aussi la difficulté à se garer. En effet, dès lors qu'un parking -en particulier pour les grands parkings- arrive proche de la saturation, le temps pour un individu pour trouver une place augmente sensiblement. Il peut être utile que certains parkings soient le moins souvent possible pleins à plus de 80% (par exemple) afin de limiter le cruising. Les résultats obtenus par les simulations peuvent être utilisés pour répondre à ce type de demande. Par exemple, si l'on souhaite qu'un parking ne soit jamais plein à plus de 80%, avec 10% d'échecs admis, il suffira de regarder à partir de combien de places on admet moins de 10% de taux d'échec, puis de diviser ce nombre par 80% pour connaître la quantité de places nécessaires pour satisfaire cette contrainte. Atteindre une saturation trop élevée ayant moins de conséquences que d'avoir un client qui na pas pu se garer du tout. On pourra ici s'intéresser à des taux d'échecs bien supérieurs en comparaison du premier cas.

Bien évidement, ces deux notions peuvent être utilisées simultanément. On peut décider de ne pas avoir plus de 0.5% d'échec et de n'entrer en saturation de 80% d'utilisation que dans 10% des cas au maximum. Il

suffit de regarder le nombre de places nécessaires pour satisfaire la première contrainte, puis la deuxième, et de prendre le plus grand nombre de places parmi ces deux. Ce sera donc le plus petit nombre de places qui satisfait à la fois les deux contraintes.

6.5 Évaluation du nombre d'utilisations possibles pour un parking en connaissance de la quantité de place

La transformation inverse est aussi possible. Dans le cas des problèmes MA-TSP-PRC ou variantes (c'est à dire les problèmes MA-TSP-PR qui prennent en compte une capacité maximale d'utilisations de parking), il n'est pas chose aisée de déterminer qu'elle est la capacité d'utilisation maximale d'un parking sur une période donnée. On dispose plutôt généralement d'une capacité physique maximale en nombre de places de parking. La méthode utilisée pour obtenir un nombre idéal de places connaissant l'utilisation théorique est aussi possible dans le sens contraire. Cette version ne sera pas détaillée aussi finement que la précédente car elle repose intégralement sur la même méthodologie.

Méthode :

- Initialisation, on dispose toujours de règle de probabilité permettant de définir les chance d'arriver à chaque instant discrétisé de notre période. Les précédents agents sont remplacés par des probabilités de durée de stationnement. Une dernière donnée supplémentaire est demandée, la limite en places de parking.
- Étape 1 : création du dé (identique à la méthode précédente) pour les temps d'arrivée sur le parking, ainsi que d'un nouveau dé qui fournira aléatoirement une durée de stationnement.
- Étape 2 : Simulation par création d'agents un par un, dont le temps d'arrivée et la durée sont choisis aléatoirement à l'aide des deux dés. À chaque ajout d'agents, on vérifie le nombre de places utilisées. Le processus est arrêté dès lors qu'un agent ajouté engendrerait un dépassement de la limite en places de parking. Le nombre d'agents, sans compter ce dernier est alors enregistré.
- Étape 3 : Plusieurs simulations de l'étape 2 se succèdent jusqu'à un nombre décidé au départ (plus il y a de simulations, plus les résultats sont pertinents). Il ne reste qu'à répertorier les statistiques

résultantes pour obtenir le nombre d'utilisation que l'on peut réaliser avec ces places de parkings. Ceci de la même manière que la méthode réciproque. Nous obtiendrons donc une liste de probabilité d'utilisations possibles à partir de ce nombre de places de parking (exemple de donnée résultante possible : avec seulement 0.5% de chance d'échec, nos 40 places de parking peuvent satisfaire 120 utilisations).

6.6 Impact d'une mutualisation de parking

Un des leviers sur lequel ces travaux s'appuient est le nombre de parkings présents, que l'on cherche à limiter pour favoriser une certaine mutualisation entre les commerces. Il est sous-entendu de diverses manières que diminuer ce nombre de parkings, et donc forcer d'une certaine manière cette mutualisation, fera diminuer le nombre de places de stationnement pour satisfaire le besoin (et donc par conséquent, diminuera la surface). Cela peut se comprendre assez facilement en imaginant deux parkings qui fusionnerait en un seul.

Imaginons donc que seuls deux parkings sont présents sur une zone, et que nous devons estimer leur nombre de places nécessaires. Ils vont être utilisés respectivement un nombre n et m de fois. Afin de simplifier l'impact de la mutualisation nous considérerons qu'aucune utilisation n'a été fait par le même agent (donc le nombre d'utilisation d'un parking résultant de la fusion des deux parkings aura un nombre d'utilisation de $n + m$). Cette simplification nous permet de traiter le pire cas (celui ou potentiellement nous gagnerons le moins de place, s'il y a en effet un gain) étant donné que le nombre d'utilisation de ce parking mutualisé est sa borne maximale.

Si l'on traitait le premier parking (avant mutualisation), afin d'attribuer ses utilisations, on obtiendrait l'agencement des différents véhicules répartis temporellement sur un certain nombre de places, comme nous avons pu le voir lors de l'explication de la méthodologie à la section 6.3 sur la figure 6.4. Sur cet exemple de simulation, on voit que le maximum de place nécessaires était atteint au temps $t = 30$ pour un besoin de 39 places. Si une nouvelle utilisation supplémentaire était ajouté sans concerner ce temps $t = 30$ sur toute sa durée, elle pourrait se faire sur ce même parking sans en augmenter le nombre de places nécessaire. Dans

cette optique donc, lorsque l'on traitera le deuxième parking, de manière séparé, certaines utilisations le concernant pourraient utiliser des espaces encore disponibles du premier parking. Si l'on pouvait réaliser ce déplacement (permettre à une utilisation du deuxième parking de s'effectuer sur le premier quand cela est possible sans en augmenter le nombre de place nécessaire) le deuxième parking se retrouverait donc avec moins d'utilisations, et en conséquent nécessitera certainement de moins de places de parking. Cette optimisation des utilisation de place de parking entre deux zone de stationnement sera rendu possible par la mutualisation. Cela pourra réduire la quantité de places de parking nécessaires permettant à chaque utilisateur de trouver une place. Suivant le cas ce gain n'est pas obligatoire, mais le nombre de places nécessaires pour la mutualisation des deux ne dépassera jamais le nombre de places nécessaires pour les deux parkings séparés.

Mutualisation entre 2 parkings équivalents

La méthodologie décrite dans la section 6.3 va pouvoir appuyer cet avis théorique par une simulation. La mutualisation testée est la suivante :

- 2 parkings vont fusionner pour n'en former plus qu'un (ou l'un des deux est supprimé). Chaque utilisation des 2 parkings se répercute directement et intégralement sur cette fusion des deux. On ne considérera pas que des agents utilisaient les deux avant fusion, afin de ne pas intégrer de biais qui iraient encore plus dans le sens de ce que l'on souhaite démontrer (c'est à dire que l'on se situera dans le cas ou la fusion serait la moins bénéfique).
- Chacun des deux parkings admet 500 utilisations, 100 de 10 minutes, 100 de 20 minutes, 100 de 30 minutes, 100 de 40 minutes et enfin 100 de 50 minutes.
- Les probabilités d'heures d'arrivées seront réparties équitablement sur 4 heures, avec une discrétisation du temps par palier de 5 minutes.

Le test réalisé se compose donc en 2 parties :

- Simulation du nombre de places nécessaires pour un parking avant la fusion. Les deux parkings ont des données identiques donc il est inutiles de tester les deux. Les nombres de places nécessaires trouvés en fonction des pourcentages d'échecs accordés seront simplement multipliés par 2.

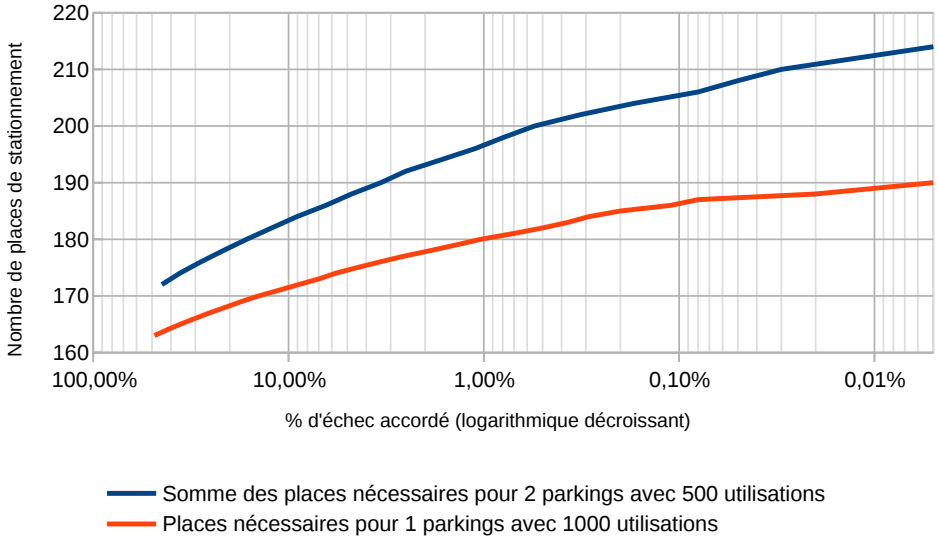


FIGURE 6.7 – Test du nombre de places de stationnement nécessaires en fonction du taux d'échec accordé, avant et après mutualisation de 2 parkings avec 500 utilisations.

- Simulation du nombre de places nécessaires pour la fusion des deux parkings.

Pour finir, on compare le nombre de places nécessaires total pour satisfaire les besoins des 2 parkings séparément, et le nombre de places nécessaires si l'on fusionne les deux. La figure 6.7 met en comparaison les courbes des nombres de places en fonction du taux d'échec accordé pour ces deux cas.

On constate pouvoir économiser environ 9% de places de stationnement en mutualisant ces deux parkings si on admet un très faible taux d'échec. Si on admet un taux d'échec maximum de 0.1% par exemple (c'est à dire que les 1000 utilisations pourront être effectuées sans qu'il ne manque de place de parking, avec au moins 99.9% de chance), le total de places nécessaires si les 2 parkings ne sont pas mutualisés est de 206 places (103 pour chacun des deux parkings). Après mutualisation, le parking résultant n'a besoin que de 187 places pour admettre un taux d'échec inférieur à 0.1%, soit environ une réduction d'environ 9.2% de places de stationnement par rapport à la situation avant mutualisation. On ne peut cependant admettre de règle précise de par ces tests quant au gain gagné de manière général. Beaucoup de paramètres pourront faire varier ce gain. À fortiori les suppression de parking seront plus complexe qu'une simple mutualisation de deux parkings : si le principe reste le même, les utilisations d'un parking supprimé pourront se répercuter sur plusieurs

autres parkings. Ce gain pourra apparaître dès lors que la période de temps étudiée permet de disposer de moins places de stationnement qu'il n'y a d'utilisations : c'est à dire qu'une même place de stationnement pourrait, sur la même période étudiée, être utilisée au moins 2 fois. Ce qui est parfaitement compatible avec le cas d'une zone commerciale, où il y a de fortes rotations d'agents pour une même place de parking, et au contraire plutôt incompatible dans le cas d'une zone de stationnement liées à des habitations en période nocturne, où il y aura très peu de rotations. Dans ce deuxième contexte il pourrait être préférable de considérer une équivalence entre le nombre d'utilisations et le nombre de places de parkings nécessaires pour satisfaire ces utilisations.

Mutualisation : exemple du Pontet (section 5.2.2)

Les résultats de l'exemple du Pontet de la section 5.2.2 est repris pour estimer quel serait le gain d'une réduction du nombre de zone de stationnement, en terme de nombre de place. Les quantités d'utilisations trouvées en résultat étaient en nombre d'utilisation effectué par un agent. Dans cet exemple nous avions spécifiés que afin de simplifier le problème un agent représentait en réalité 10 personnes ayant les même propriétés. On multiplie donc les utilisations des parkings résultantes par 10 afin de connaître le vrai nombre d'utilisations.

TABLE 6.2 – Nombre de places nécessaires, en fonction du nombre de zone autorisées dans le cas du Pontet, en satisfaisant les contraintes : jamais plus de 90% de saturation, 5% d'échec autorisé.

Nombre de parkings	Nombre d'utilisation des parkings	Nombre de places conseillé	Total
6	P1 :2140 P2 :2100 P5 :1180 P6 :1110 P7 :1900 P8 :1040	P1 :276 P2 :270 P5 :161 P6 :152 P7 :247 P8 :144	1250
5	P1 :2380 P2 :2070 P5 :2150 P7 :1770 P8 :1100	P1 :303 P2 :267 P5 :277 P7 :231 P8 :151	1229
4	P1 :2120 P2 :2770 P7 :1820 P8 :2760	P1 :273 P2 :349 P7 :238 P8 :348	1206

On constate que le gain en nombre de place de parking lorsque l'on réduit le nombre de parking reste très faible, en comparaison à l'augmentation de temps de déplacement sur ces zones. Passer de 6 à 5 zones ne génère qu'un économie d'environ 2% de places de stationnement pour une augmentation de 13% du temps de déplacement. Passer de 5 à 4

zones de stationnement réitère cet économie de place de stationnement d'environ 2% supplémentaire, cette fois-ci pour une hausse du temps de déplacement de l'ordre de 8%.

Bien évidement, cet exemple ne peut faire office de généralité. Si ce gain de l'ordre de 2% pourrait se retrouver lors de mutualisations généralisées pour passer de 6 à 5 ou de 5 à 4 zones de stationnement devraient se retrouver dans d'autre situation, cela ne sera pas le cas de la hausse de temps de déplacement. En effet, si le temps de déplacement augmente autant dans cet exemple, il ne faut pas oublier que cela est en partie dû aux congestions extrêmes désirées pour le test de nos méthodologies, même pour la situation avec 6 parkings. De manière évidente, le fait de supprimer un parking sur une situation qui est déjà très instable prend beaucoup plus d'ampleur que si le trafic étaient initialement fluide

Dans tous les cas, la rentabilité ou non de cet échange entre temps de déplacement et places de parking ne sera traitée dans notre étude. Notre objectif à ce niveau n'étant que de donner la possibilité aux personnes chargées de l'aménagement d'avoir les informations nécessaire leurs permettant d'effectuer ce choix de manière éclairée.

Chapitre 7

Conclusions générales et perspectives

Conclusion générale en cours de rédaction.

Annexes

Annexe A

Modélisations OPL - CPLEX

N.B. : Dans les parties où du code est représenté, comme il est souvent d'usage la totalité des accentuations ont été supprimés (même dans les commentaires).

A.1 MA-TSP-PR-DL

A.1.1 Modélisation en arcs avec voitures

Données d'entrée

La liste des identifiants de chaque entité du graphe, ainsi que les commerces non-localisés, classée en type d'entité (chaque identifiant est unique pour toutes ces entités) :

```
{int} Entrees = ...;
{int} Sorties = ...;
{int} Parkings = ...;
{int} Commerces = ...;
{int} AeraComm = ...;
// "..." signifie que les valeurs seront a recuperer dans le fichier de
    édonnes
```

La liste des affectations possibles :

```
tuple Affectation{
  int ID;
  int p;
}
//(ID=numero du commerce, p=numero de la localisation pour commerce)
{Affectation} PossComm=...;
```

La liste des arcs ainsi que le coût associé à chacun :

```
tuple Edge
{
  int orig;
  int dest;
}
// liste de tous les arcs
{Edge} Edges = ...;
// matrice des couts associes a chaque arc
float C[Edges]=...;
```

Les agents :

```
tuple Agent
{
  key int num;
  int e; // Identifiant éEntre
  int s; // Identifiant Sortie
}
// Liste agents
{Agent} Agents = ...;
// Liste des listes d'objectifs des agents
{int} m[Agents]=...;

//Calcul du nombre d'etape maximal de chaque agent
int Ka[a in Agents]=3*sum(c in m[a])1 + 1;
int Kmax=(max(a in Agents)Ka[a]);
// Creation de la liste K pour les etapes des agents
range K=1..Kmax;
```

Variables décisionnelles

```
// y les arcs empruntes par les agents :
// Agents = numero de l'agent
// K = numero de deplacement
// Edges = numero d'arrete
dvar boolean y[Agents][K][Edges];
// x les arcs empruntes par les voitures d'agent :
dvar boolean x[Agents][K][Edges];
// les affectations Commerce->Localisation :
dvar boolean affComm[PossComm] ;
```

Fonction d'optimisation

```
minimize sum(a in Agents,  ed in Edges, k in K) C[ed]*y[a][k][ed];
```

Contraintes

Contraintes d'affectations :

```
  forall(c in Commerces)
    ct1: sum(poss in PossComm : poss.ID == c) affComm[poss]==1;
  forall(p in Points)
    ct2: sum(poss in PossComm : poss.p == p) affComm[poss]<=1;
```

Contraintes d'accès aux objectifs :

```
 forall (a in Agents, c in AeraComm){
      ct3: sum(e in Edges:e.dest==c, k in K) y[a][k][e] ==
           sum(poss in PossComm : poss.ID in m[a] &&
               poss.p==c)affComm[poss];
   }
```

Contraintes de trajets :

```
//ct4/4b : chaque agent, chaque etape, un seul deplacement
forall(a in Agents, k in K)
    ct4 : sum(ed in Edges)y[a][k][ed]<=1;
forall(a in Agents, k in K)
    ct4b : sum(ed in Edges)x[a][k][ed]<=1;


// ct5/5b : chaque agent demarre à son origine
// et fini a sa sortie
forall (a in Agents)
    ct5: sum(e in Edges:e.orig==a.e) y[a][1][e] == 1;
forall (a in Agents)
   ct5b: sum( e in Edges:e.dest==a.s, k in K) y[a][k][e] == 1;


// ct6 : chaque agent suis un chemin continu
forall (a in Agents,j in Points:j not in Sorties, k in K : k<etapeMax)
    ct6: sum(e in Edges:e.dest==j) y[a][k][e] == sum(e in
        Edges:e.orig==j)y[a][k+1][e];
// ct6b : chaque voiture suis un chemin continu sur la route
forall (a in Agents,j in Points:j not in Sorties , k in K: k<etapeMax)
    ct6b : sum(e in Edges:e.dest==j) x[a][k][e] == sum(e in
        Edges:e.orig==j)x[a][k+1][e];
```

```
22
23
24 //ct7, chaque voiture ne circule jamais sur le érseau non-routier
25 forall (a in Agents, e in Edges:e.orig in AeraComm || e.dest in
       AeraComm, k in K)
26    ct7: x[a][k][e] == 0;
27
28 // ct8 : chaque agent circule sur le reseau routier avec sa voiture
29 forall (a in Agents, e in Edges:e.orig not in AeraComm && e.dest not in
       AeraComm)
30    ct8: sum(k in K) x[a][k][e] == sum(k in K) y[a][k][e];
```

A.1.2 Modélisation en arcs sans voiture

Données d'entrée

Les données d'entrées sont exactement les mêmes que pour le modèle avec voitures.

Variables décisionnelles

Les arcs empruntés par les voitures sont supprimés

```
1 // y les arcs empruntes par les agents :
2 dvar boolean y[Agents][K][Edges];
3 // les affectations Commerce->Localisation :
4 dvar boolean affComm[PossComm] ;
```

Fonction d'optimisation

Identique au modèle avec voiture.

Contraintes

Les contraintes d'affectations et d'accès aux objectifs sont identiques au modèles considérant les voitures.

Contraintes de trajets :

```
1 //ct4 : chaque agent, chaque etape, un seul deplacement
2 forall(a in Agents, k in K)
3     ct4 : sum(ed in Edges)y[a][k][ed]<=1;
4
5
6 // ct5/5b : chaque agent demarre a son origine
```

```
// et fini a sa sortie
forall (a in Agents)
    ct5: sum(e in Edges:e.orig==a.e) y[a][1][e] == 1;
forall (a in Agents)
   ct5b: sum( e in Edges:e.dest==a.s, k in K) y[a][k][e] == 1;


// ct6 : chaque agent suis un chemin continu
forall (a in Agents,j in Points:j not in Sorties, k in K : k<etapeMax)
    ct6: sum(e in Edges:e.dest==j) y[a][k][e] == sum(e in
        Edges:e.orig==j)y[a][k+1][e];


// ct7 : contrainte de retour au parking
 forall (a in Agents,p in Parkings, k in K : k <= Ka[a] )
    ct7: sum(e in Edges: e.dest==p &&  e.orig in AeraComm ,l in K :
        l>=k && l <= Ka[a]) y[a][l][e]
    >= sum(e in Edges: e.orig==p && e.dest in AeraComm ,l in K : l >=k
        && l <= Ka[a])y[a][l][e];
```

A.1.3 Modélisation en sous-trajets

Données d'entrée

La liste des identifiants de chaque entité du graphe, la liste des affectations des commerces sur les localisations, ainsi que la liste des agents sont conservées telles que présentées dans la modélisation en arc qui considère la voiture.

Le seul changement au niveau des données d'entrée est au niveau des arcs. Les arcs ne sont plus demandés, les listes de chaque type de sous-trajets viennent les remplacer :

```
// Entree -> Parking -> Commerce
tuple tripleIPC
{
  int I;
  int P;
  int C;
}
{tripleIPC} IPC = ...;
float IPC_Cost[IPC]=...; // cout de chaque "ipc"

// Commerce -> Parking -> Sortie
tuple tripleCPO
{
  int C;
  int P;
  int O;
}
{tripleCPO} CPO = ...;
float CPO_Cost[CPO]=...; // cout de chaque "cpo"
```

```
20 // meme si les types tripleCPO et tripleIPC sont identiques, les
       distinguer permettra plus de éclart, par les noms de leur
       èparamtre, au niveau des contraintes.
21
22 // Commerce -> Parking -> Parking -> Commerce
23 tuple quadraCPPC
24 {
25   int C1;
26   int P1;
27   int P2;
28   int C2;
29 }
30 {quadraCPPC} CPPC = ...;
31 float CPPC_Cost[CPPC]=...; // cout de chaque "cppc"
```

Variables décisionnelles

```
1  dvar boolean F[Agents][IPC]; // premier deplacement
2  dvar boolean L[Agents][CPO]; // dernier deplacement
3  dvar boolean N[Agents][CPPC][K]; // deplacement inter-commerce
4
5  dvar boolean affComm[PossComm] ;
```

Fonction d'optimisation

```
1  minimize
2   sum(a in Agents,  ipc in IPC) IPC_Cost[ipc] * F[a][ipc] +
3   sum(a in Agents,  cpo in CPO) CPO_Cost[cpo] * L[a][cpo] +
4   sum(a in Agents,  cppc in CPPC, k in K) CPPC_Cost[cppc] *
        N[a][cppc][k];
```

Contraintes

Les contraintes d'affectations (ct1/ct2) restent identiques à la modélisation en arcs avec les voitures. Ce sont les seuls contraintes conservées en l'état.

Contraintes d'accès aux objectifs :

```
1 // ct3 : si une localisation est éaffecte par un objectif de l'agent
```

```
// alors cet objectif doit être attein (en "F" ou en "N")
forall(a in Agents, aera in AeraComm)
     ct3:
     (sum(ipc in IPC : ipc.C==aera)
        F[a][ipc])
     +
     (sum(cppc in CPPC : cppc.C2 == aera, k in 1..Ka[a])
         N[a][cppc][k])
     ==
     sum(poss in PossComm : poss.p == aera && poss.ID in m[a])
        affComm[poss];
```

Contraintes de trajets :

```
// ct4/ct4b/ct4t/ct4q : chaque agent, chaque etape <  Ka[a], un seul
    deplacement
forall(a in Agents)
    ct4: sum(ipc in IPC) F[a][ipc] == 1;
forall(a in Agents)
    ct4b: sum(cpo in CPO) L[a][cpo] == 1;
forall(a in Agents, k in K)
    if(k <= Ka[a])
      ct4t: sum(cppc in CPPC) N[a][cppc][k] == 1;
    else
      ct4q: sum(cppc in CPPC) N[a][cppc][k] == 0;


// ct5/ct5b : chaque agent demarre a son entree
// et fini a sa sortie
forall(a in Agents){
     ct5:
        sum(ipc in IPC: ipc.I==a.i) F[a][ipc]==1;
     ct5b:
        sum(cpo in CPO: cpo.O==a.o) L[a][cpo]==1;
}

// ct6_1 : si 1 seul objectif, continuite entre "F" et "L"
forall(a in Agents, p in Parkings, c in AeraComm)
    if(Ka[a]==0)
        ct6_1:
        (sum(ipc in IPC : ipc.I==a.i && ipc.C==c && ipc.P==p)F[a][ipc])
        ==
        (sum (cpo in CPO : cpo.P==p && cpo.C==c) L[a][cpo]);


// ct6_2/ct6_2b : si l'agent a 2 objectifs ou +,
// continuite entre "F" et "N", ainsi qu'entre "N" et "L"
forall(a in Agents, p in Parkings, c in AeraComm){
    if(Ka[a]>0){
        ct6_2:
            (sum(ipc in IPC : ipc.I==a.i && ipc.C==c &&
                ipc.P==p)F[a][ipc])
            ==
            (sum (cppc in CPPC : cppc.P1==p && cppc.C1==c)
                N[a][cppc][1]);
        ct6_2b:
            (sum(cpo in CPO : cpo.O==a.o && cpo.C==c &&
```

```
                cpo.P==p)L[a][cpo])
41              ==
42              (sum (cppc in CPPC : cppc.P2==p && cppc.C2==c)
                    N[a][cppc][Ka[a]]);
43      }
44  }
45
46  // ct6_3 : si l'agent a 3 objectifs ou +,
47  // continuite entre les etapes de "N"
48  forall(a in Agents, p in Parkings, c in AeraComm, k in K : k<Ka[a])
49      if(Ka[a]>0)
50          ct6_3:
51          (sum (cppc in CPPC : cppc.P2==p && cppc.C2==c) N[a][cppc][k])
52          ==
53          (sum (cppc in CPPC : cppc.P1==p && cppc.C1==c) N[a][cppc][k+1]);
```

A.2 MA-TSP-PRCL-DL (en sous-trajets)

Données d'entrée

Sont rajoutées au précédent modèle les données d'entrées suivante :

```
1  // S : force l'existence d'un parking
2  // 0= peut exister ou non, 1 = contraint d'exister
3  int S[Parkings]=...;
4  // nombre total de parking éautoris
5  int TotalParking=...;
6
7  // écapacit maximal d'utilisation de parking
8  int T[Parkings]=...;
```

Variables décisionnelles

Sont rajoutées au précèdent modèle les variables décisionnelles (artificielles) suivantes :

```
1  // s : existence d'un parking
2   dvar boolean s[Parkings];
3  // t : nombre d'utilisation d'un parking
4   dvar int t[Parkings];
```

Fonction d'optimisation

Identiques au modèle précédent.

Contraintes

Deux types de contraintes sont rajoutés. Une partie liée aux capacités d'utilisation de parking et l'autre liée à leur existence (limite du nombre de parkings existant).

Contraintes d'existence de parking :

```
// s[p] force a 1 si S[p]==1
forall(p in Parkings)
   ct_prl1:
      s[p] >= S[p];

// s[p]= 1 si le parking est utilise lors d'un "F"
forall(ipc in IPC, a in Agents, k in K)
    ct_prl2a:
        s[ipc.P] >= F[a][ipc];

// s[p]= 1 si le parking est utilise lors d'un "N"
// pour atteindre le commerce suivant
forall(cppc in CPPC, a in Agents, k in K)
    ct_prl2b:
        s[cppc.P2] >= N[a][cppc][k];

// s[p]=0 si aucun element ne le force a valoir 1
forall(p in Parkings)
    ct_prl3:
        s[p] <= sum(a in Agents, ipc in IPC : ipc.P==p)F[a][ipc] +
            sum(a in Agents, cppc in CPPC : cppc.P2==p, k in
            K)N[a][cppc][k];

// total de parking existant <= total de parking autorise en donnee
ct_prl4:
    sum(p in Parkings)s[p] <= TotalParking;
```

Contraintes de capacité d'utilisation de parking :

```
// calcul du nombre d'utilisation de chaque parking
forall(p in Parkings)
    ct_prc1:
        t[p] ==
            sum(a in Agents, ipc in IPC : ipc.P==p)
                F[a][ipc]
            +
            sum(a in Agents, cppc in CPPC : cppc.P2==p, k in K)
                N[a][cppc][k];

// nombre d'utilisation d'un parking doit etre inferieur
// a sa capacite d'utilisation donnee
forall(p in Parkings)
    ct_prc2:
```

```
15      t[p] <=T[p];
```

À noter : le choix de compter comme 2 utilisations le fait qu'un même agent reste sur le même parking n'est pas obligatoire. Si l'on préfère ne compter que pour une seule utilisation le fait qu'un agent reste sur un même parking pour voir plusieurs commerce il suffit de modifier la contrainte ct_prc1 comme telle (on ne compte plus les déplacements inter-objectifs "N" pédestres, où $P1 == P2$) :

```
1  // ct_prc1 modifiee
2  forall(p in Parkings)
3      ct_prc1_modifiee:
4          t[p] ==
5              sum(a in Agents, ipc in IPC : ipc.P==p)
6                  F[a][ipc]
7              +
8              sum(a in Agents, cppc in CPPC : cppc.P2==p &&
                   cppc.P1!=cppc.P2, k in K)
9                  N[a][cppc][k];
```

A.3 Problème d'affectation, issu d'un MA-TSP-PR-DL après pré-traitement des trajets complets (section 4.1.4)

Données d'entrée

La liste des identifiants de chaque entité du graphe, ainsi que les commerces non-localisés, classée en type d'entité (chaque identifiant est unique pour toutes ces entités) :

```
1  {int} entrees=...;
2  {int} sorties=...;
3  {int} parkings=...;
4  {int} localisations=...;
5  {int} commerces=...;
```

La liste des affectations possibles :

```
1  tuple Affectation{
2    int ID;
3    int p;
```

```
}
//(ID=numero du commerce, p=numero de la localisation pour commerce)
{Affectation} PossComm=...;
```

La liste des ensembles de localisation possibles :

```
tuple ListInt
{
  {int} list;
  int nb_entity;
  // nb_entity est le nombre d'element
  // de la liste, pour ne pas avoir a compter
}
// ensemble possible des listes de localisations pouvant être
// affectees par la liste d'objectif d'un agent
// ( Il s'agit d'une liste de liste )
{ListInt} ensembles_localisations=...;
```

On regroupe les entrées et sortie possible pour chaque agent, pour plus de simplicite :

```
tuple IO
{
  int i;//entree
  int o;//sortie
}
{IO} io=...;
```

Les agents deviennent tels :

```
tuple Agent
{
  int ID;
  IO io; // entree+sortie
  ListInt objectifs; // liste des objectifs
}
{Agent} agents = ...;
```

Pour gagner en rapidité (afin d'éviter de compter plusieurs fois plusieurs agents ayant les mêmes objectifs dans certaines variables décisionnelles), on crée la liste des ensembles d'objectifs possibles (que possède au moins l'un des agents). Cela pourrait néanmoins être facultatif (on peut se servir directement de la liste présente dans les données de chaque agent) :

```
{ListInt} ensembles_objectifs=...;
```

Enfin, la nouvelle matrice de coût, qui dépend d'un ensemble de localisations à atteindre (localisations des objectifs), de l'entrée et de la sortie (le coût total est pré-calculé en pré-traitement par résolution d'un TSP-PR) :

```
tuple Cout
{
  int cout; // le cout
  ListInt loc; // ensemble de localisation a atteindre
  IO io; // entree+sortie
}
int couts[ensembles_localisations][io]=...;
```

Variables décisionnelles

```
// x : affectation 1-1 d'un objectif sur une localisation
 dvar boolean x[affectations] ;
// y : affectation *-* d'un ensemble d'objectifs
// que possede l'un des agents
// sur un ensemble de localisations
 dvar boolean y[ensembles_objectifs,ensembles_localisations] ; //y
```

Fonction d'optimisation

La nouvelle fonction d'optimisation va minimiser le produit de chaque affectation multiple (c'est à dire d'ensemble d'objectifs vers un ensemble de localisation) par la somme des coûts qu'engendrerais cette ensemble de localisation avec l'entrée et la sortie de chaque agent ayant cette liste d'objectifs. Cette fonction est bien évidement linéaire comme lors des précédentes modélisations :

```
 minimize
  sum(
    ens_obj in ensembles_objectifs,
    ens_loc in ensembles_localisations :
        ens_loc.nb_entity==ens_obj.nb_entity
  )
     y[ens_obj,ens_loc]
```

```
8       *
9       ( // * le cout, qui ne depend pas des VD
10         sum(a in agents : a.objectifs==ens_obj)
11             c[ens_loc][a.io]
12       )
```

Contraintes

On retrouve les contraintes d'affectations 1-1 des objectifs sur les localisations :

```
1  ct1:
2  // tout les commerces affectent exactement 1 localisation
3   forall(c in commerces.list)
4     sum(aff in affectations : aff.comm == c)
5       x[aff] == 1;
6
7
8  ct2: // les localisations sont éaffectes au maximum par 1 commerce.
9   forall(l in localisations.list)
10    sum(aff in affectations : aff.loc == l)
11      x[aff] <= 1;
```

On rajoute une contrainte d'unicité pour les affectations multiples *-* et on élimine les cas ou le nombre d'entités diffère (pour fixer leur valeur, qui n'influencera pas la fonction objectif) :

```
1
2   ct3: // Chaque ens_obj n'affecte qu'un seul ens_loc
3   forall(ens_obj in ensembles_objectifs)
4     sum(ens_loc in ensembles_localisations :
          ens_loc.nb_entity==ens_obj.nb_entity)
5      y[ens_obj,ens_loc] == 1;
6
7   ct3bis:
8   // si nb d'objectif est different de nb de localisation, y == 0
9   forall(ens_obj in ensembles_objectifs)
10    sum(ens_loc in ensembles_localisations :
          ens_loc.nb_entity!=ens_obj.nb_entity)
11     y[ens_obj,ens_loc] == 0;
12 }
```

Pour finir, la contrainte la plus importante forcera indirectement ce que l'on attend de ces affectations multiples *-* : que ces variables artificielles y prennent la valeur de 1 si chaque élément de l'ensemble des objectifs affecte un élément de l'ensemble de localisation. Nous avons d'une part la contrainte ct3 qui oblige chaque y à prendre une fois la valeur de 1 pour chaque ensemble d'objectif possible (c'est à dire que

chaque ensemble d'objectif doit affecter un ensemble de localisation). Il ne reste donc qu'à forcer avec cette contrainte y à prendre la valeur de 0 si au moins l'un des objectifs n'affecte pas l'une des localisations pour les deux ensembles concernés. Le seul choix restant que pourra, et devra (ct3), affecter une liste d'objectifs sera la liste unique (l'unicité peut se vérifier grâce à la contrainte ct1) des localisations affectée 1 à 1 par ces objectifs (choix de la variable décisionnelle x) :

```
1  ct4:
2  // si une localisation loc n'est affectee par
3  // aucun des objectifs de la liste d'objectifs ens_obj
4  // alors aucune  liste de localisations contenant loc
5  // ne peut etre affectee par la liste ens_obj
6   forall(ens_obj in ensembles_objectifs, loc in localisations)
7          sum(ens_loc in ensembles_localisations :
               ens_loc.nb_entity==a.comm.nb_entity && loc in ens_loc.list)
8            y[a,ens_loc]
9        <=
10       sum(aff in affectations : aff.comm in ens_obj.list && aff.loc ==
             loc)
11           x[aff];
12 }
```

Annexe B

Génération de test

Sommaire

B.1 Introdution

De nombreux tests ont été réalisés, en ayant pour but l'évaluation de différents modèles ou heuristiques. Au vu de la quantité assez conséquente de tests à réaliser il n'est pas concevable d'obtenir autant d'ensembles de jeu de données réels. Il est donc nécessaire de pouvoir générer des jeux de données aléatoires en grand nombre : donc d'avoir un outil permettant de générer rapidement des jeux de données aléatoire.

Dans cet objectif, un logiciel a été créé pour générer, dans un premier temps, des "situations". Le terme **situation** qui revient souvent ici définit simplement un jeu de données pour l'un de nos tests, représentant un de nos problèmes à résoudre. Il s'agit donc de la réunion d'un ensemble de points d'entrée, de sortie, de stationnement, de commerce, éventuellement de localisation si les commerces ne sont pas localisés. À cela se rajoute toutes les autres données d'entrée qui peuvent rentrer en considération pour l'un de nos problèmes, comme les ensembles d'affectations

possibles ou les capacités d'utilisations des parkings. Un deuxième terme similaire qui revient souvent est le **type de situation**. Sa définition est plus vague : il s'agit d'un groupement similaire de situations qui est défini par certaines propriétés. Au niveau de la génération de situations aléatoires, un type de situation sera défini par toutes les propriétés qui ne seront pas aléatoires lors de la génération. Parmi ces propriétés par exemple le fait d'avoir 10 agents. Un type de situation qui a parmi ses nombreuses propriétés le fait de disposer de 10 agents ne définira pas qu'elle sont les propriétés exactes de ces agents, qui seront, à la génération, en grande partie aléatoires. Un très grand nombre de situation peuvent donc appartenir à un même type de situation.

Cette annexe présentera les divers choix qui ont pu être effectués liés à ces situations. La deuxième section détaille comment sont générées les situations. La troisième section explique comment sont générés, à partir de ces données, les fichiers de données destinés à CPLEX ou MatSim. Enfin, la quatrième section parle de la lecture des résultats donné par MatSim permettant de mettre à jour les données des situations.

B.2 Génération de situation

Une situation sera donc générée aléatoirement en répondant aux critères d'un type de situation demandé par un test. Voici présentée une liste non-exhaustive des principaux critères d'un type de situation, avec pour chacun leur rôle ainsi que la partie aléatoire qui en découlera, quand il y en a une :

- Xmax / Ymax : définis le cadre de la zone générée. Pour plus de simplicité il est considéré que les limites de la zone générée est un rectangle, qui servira de repère orthogonal. Ce rectangle peut être défini par sa diagonale partant du point d'origine (0,0) à (Xmax,Ymax). Pour la majorité de nos tests ces critères sont fixés à 1000 (zone carré de 1000 par 1000, sans unité).
- Distance de séparation : une distance de séparation minimale est définie pour empêcher deux entités localisées d'être trop proches. Lorsqu'une entité localisée et randomisée, si elle est trop proche d'un point déjà existant, elle est randomisée de nouveau jusqu'à obtenir un emplacement possible.
- Le nombre d'entrée et sortie : il n'est pas vraiment utile pour nos test de faire en sorte que ces nombres puissent être différents.

Une valeur quantitative commune pour les deux est donc présente. Considérant que les entrées et sorties d'une zone commerciale sont, la plupart du temps, à sa périphérie, leur localisation sera aléatoirement générée en bordure de la zone.

— Le nombre de parking : Défini la quantité de zone de stationnement à placer. Ces derniers sont placés aléatoirement dans la zone.
— Le nombre de localisation d'objectifs : Défini le nombre de localisation d'objectifs à placer. Ces derniers sont placés aléatoirement dans la zone. Il est possible de renseigner aussi une distance maximale qu'un agent sera à même de parcourir à pied. Si celle-ci est renseignée, et si la randomisation d'une localisation d'objectif atteint un lieu inaccessible depuis les parkings, on effectue de nouveau sa randomisation jusqu'à obtenir une localisation accessible (tous les parkings sont générés en amont).
— Le nombre d'objectifs : Défini le nombre d'objectifs non-localisés à générer.
— Le nombre de localisations par objectif : Défini le nombre de localisations que peuvent affecter chaque objectif. Les affectations possibles ne sont pas aléatoires. Si le nombre de localisations par objectif est X, le premier objectif pourra affecter les X premières localisations, le suivant les X premières en partant de la seconde localisation, etc...
— Le nombre d'agent : Défini le nombre d'agent.
— Le nombre d'objectifs par agent : Défini le nombre d'objectifs qu'auront chaque agent indépendamment. Pour chaque agent, ses objectifs seront choisis aléatoirement parmi la liste d'objectif définie. Durant ces randomisations, si l'objectif trouvé est déjà dans la liste des objectifs précédemment sélectionnés pour l'agent, il est randomisé de nouveau jusqu'à obtenir un objectif non-sélectionné (il n'y a pas deux fois le même objectif dans la liste d'objectif d'un agent).
— Cette liste n'est pas exhaustive, d'autre paramètre comme les capacités de parkings le nombre total de parking à utiliser ou la liste des parkings contraints d'exister sont aussi modélisé. Néanmoins, les autres données sont dans la plupart des cas fixés manuellement et n'ont pas d'intérêt à être randomisée.

B.3 Écriture de fichiers de données

B.3.1 Écriture de fichiers de données pour CPLEX (ILOG OPL)

À partir des situations complètes générées sous forme de classe en Java, une fonction permet de directement générer le fichier de données pour les différents modèles vus au cours des chapitres 3 et 4.

Voici les principaux éléments à renseigner sur le fichier de données pour lancer les optimisations à l'aide d'OPL :

- Les coordonnées de chaque points localisés : ces coordonnées seront nécessaires afin d'écrire le fichier de données pour MatSim en fin d'optimisation.
- La liste des identifiants de chaque type d'entités localisées ou non : c'est à dire la liste des entrées, des sorties, des localisations d'objectifs, des objectifs (non-localisés) et des parkings.
- Les capacités et contraintes d'existences pour chaque parking.
- La liste complète des possibilités d'affectations (objectifs → localisations d'objectif)
- La liste des agents : leur identifiant et entrées/sorties.
- la liste des objectifs de chaque agent.
- Les liaisons existantes (de 2,3 ou 4 points suivant le modèle) ainsi que leurs coûts :

  * dans le cas d'une modélisation en simple liaisons (deux points), il faut écrire chaque liaison définie dans le chapitre 3. À savoir, d'entrée vers parking, entre parkings, de parking vers localisation d'objectif, entre localisations d'objectif, de localisation d'objectif vers parking, et pour finir de parking vers sortie.

  * dans le cas d'une modélisation en sous-trajet, la liste de chaque type de sous-trajet doit être écrite, avec les coûts associés à chaque sous-trajet possible. Les sous-trajets de type "First" et "Last sont représentés par 3 points : une entrée un parking et une localisation d'objectif pour le premier ; une localisation d'objectif, un parking et une sortie pour le deuxième. Les sous-trajets "Next" sont composés eux, de 4 points : deux localisations d'objectif et deux parkings. Les coûts associés sont la somme des coûts des liaisons correspondantes à ce sous-trajet. Attention toutefois, les sous-trajets de type "Next" utilisant deux fois le même parking représentent en réalité une liaison simple entre les deux localisations

d'objectif concernées.

— Des noms de fichiers : dans lesquels écrire les résultats trouvés par OPL et éventuellement pour générer le fichier de données des agents pour MatSim (lors d'un bouclage entre les deux outils).

B.3.2 Écriture de fichiers de données pour MatSim

Pour le bouclage entre OPL (résolution par CPLEX) et MatSim, le fichier de données des agents pour MatSim est directement généré depuis le résultat obtenu par CPLEX. Ce fichier est au format XML. Chaque agent y est représenté par une balise $< person >$ dans laquelle sont renseignés toutes les informations de son trajet.

Les objectifs sont absents de ce fichier, seuls comptes les destinations réelles (localisées) du graphe atteintes par les agents. Même si MatSim et OPL ne dispose pas du même graphe (en particulier au niveau des liaisons) ils ont pour points communs les différents points d'intérêts localisables (entrées, sorties, localisations d'objectifs et parkings). L'optimisation OPL, qui ne connaît que le réseau simplifié entre points d'intérêt, choisi dans un premier temps les déplacements à effectuer de point d'intérêt à point d'intérêt pour chaque agent. Ces choix seront écrit dans ce fichier XML. MatSim qui lui dispose du réseau routier réel choisira les axes routiers pour les déplacements entre ces points d'intérêt qui ont été demandés sur le fichier XML.

Le langage utilisé, OPL, utilisé par le logiciel ILOG d'IBM, permet en plus de formuler plus facilement la modélisation linéaire (qui va être traitée par CPLEX) d'effectuer divers traitement informatique après résolution. Par exemple de générer des fichiers avec la solution trouvée. Il est donc idéal pour directement écrire après en résolution le fichier nécessaire à MatSim contenant le trajet des agents. Pour que cela soit effectué il faut bien évidement renseigner aussi dans les informations du fichier de données (pour ILOG) le chemin où écrire ce fichier d'agents pour MatSim.

Voici, à titre d'exemple, le plan d'un agent (<*person*>) que MatSim doit lire :

```
<person id="0">
  <plan>
    <act type="h" link="I1R" x="454" y="712" end_time="13:26" />
      <leg mode="car" />
    <act type="p" link="P13" x="504" y="627" dur="0:01" />
      <leg mode="walk" />
    <act type="c" link="C25" x="496" y="580" dur="0:30" />
      <leg mode="walk" />
    <act type="p" link="P13R" x="504" y="627" dur="0:01" />
      <leg mode="car" />
    <act type="h" link="04" x="454" y="712" dur="0:01" />
  </plan>
</person>
```

On peut voir que les points d'intérêt sont renseignés en temps que coordonnée. MatSim permet aussi de spécifier par quel axe arrive l'agent. Ici, le réseau complet pour MatSim, qui ne sera pas vu, est construit de manière à connaître l'identifiant et le type du point atteint par ce lien d'accès. On peut donc dans cette exemple savoir que cet agent par de l'entrée ayant pour identifiant 1 (link="I1R"), se gare au parking 13 (link="P13") pour atteindre le commerce 25 (link="C25") à pied. Ensuite il repart à pied de ce commerce pour récupérer son véhicule depuis le parking 13 et sort enfin par la sortie 4. Entre chaque point d'intérêt (représenté ici par une activité *<act>* qui a un lieu et une durée) il faut impérativement spécifier le mode de déplacement, "*walk*" pour "à pied" ou "*car*" si l'agent utilise sa voiture. Quand il s'agira du mode "*car*", c'est à la simulation MatSim de choisir le chemin exact à réaliser sur cette partie du trajet, d'après le réseau routier réel dont MatSim dispose.

B.4 Lecture des résultats MatSim

Plusieurs méthodes permettent de récupérer les temps entres les noeuds en sortie de la simulation MatSim, comme par exemple les résultats présents au niveau des plans des agents (fichier output_plans.xml.gz). Néanmoins, si l'on souhaite récupérer les vrais temps qui prennent en compte la congestion qui a été engendrée, c'est à dire les temps qu'on réellement mis les agents pour leur déplacement, il est nécessaire de récupérer ces derniers au niveau des évènements (fichier output_events.xml.gz). Il a été constaté que les autres méthodes renvoient des temps ne prenant pas en compte la congestion. La récupération des temps se fait directement par lecture du fichier xml généré dans les résultats, après décompression. Chaque évènement y est répertorié. Les évènement qui nous intéressent sont ceux de type "departure" et "arrival" qui correspondent aux arrivés

et départ sur ou depuis un noeud sur lequel l'agent réalisera une activité (se garer est aussi considéré ici comme une activité). À chaque départ, on note pour l'agent concerné le temps ainsi que le point de départ. Ce point de départ n'apparaît pas sur le fichier d'évènement mais la manière dont est construit notre graphe nous permet de le connaître en fonction du nom de la liaison empruntée, qui elle y est présente. Le temps et l'agent concerné sont eux présents à chaque évènement. À chaque arrivée traitée, on calcul à partir de temps de départ de l'agent concerné le temps de trajet, et on enregistre ce temps pour la liaison entre le point de départ et le point d'arrivée. Tous les temps trouvés sont conservés pour chaque liaison, afin de calculer ensuite le temps moyen pour chaque liaison.

Annexe C

Tests

Sommaire

C.1 Tests de comparaison entre les 3 modèles pour le MA-TSP-PR

Nous avons vu dans les sections précédentes 3 méthodes de modélisation d'un même problème, le MA-TSP-PR. Ces 3 modélisations sont exactes, par rapport à notre approche de la problématique. C'est à dire que toutes trois trouveront la solution optimale pour minimiser le même coût. Ce n'est donc bien évidement pas sur leur performance de minimisation que nous allons les comparer mais en termes de temps d'exécution en mobilisant CPLEX pour résoudre les modèles codés en OPL (Open Programming Langage). Tous les tests que nous allons voir ont été réalisés sur un ordinateur quad-core de 3.2 GHz de fréquence disposant de 8 Go de mémoire vive. Le but ne sera pas vraiment de savoir quel est le temps d'exécution, mais de comparer les 3 modèles pour savoir lequel sera le plus rapide.

Hormis quelques exceptions qui seront précisées, les tests ont tous été réalisés de la manière suivante :

1. Génération d'un nombre prédéfini de situations aléatoires appartenant au même type / à la même complexité.
2. Écriture des fichiers de données de ces situations pour les 3 modélisations.
3. Résolutions des 3 modélisations pour chaque situation. On garde les temps de résolution et on vérifie la cohérence des résultats (même valeur optimale).
4. On modifie ces situations pour en changer un paramètre. Par exemple, on ajoute 1 ou plusieurs parkings.
5. On résout les modélisations avec les nouvelles données générées. On garde les temps de résolution, on vérifie la cohérence, et on revient à l'étape 4 jusqu'à l'arrêt des itérations.

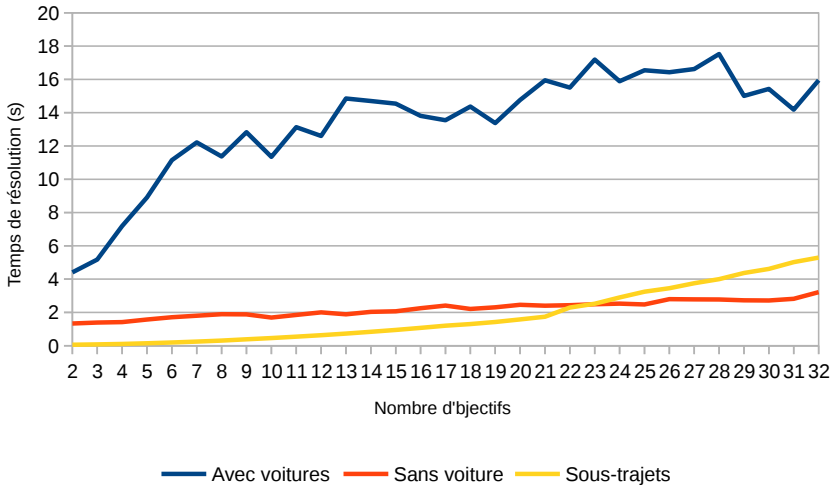
La condition d'arrêt (le nombre d'itérations) sera fixée suivant les cas pour différentes raisons :

- Le nombre d'itérations effectuées est suffisant pour avoir un visuel correct de l'impact ou non du facteur étudié.
- Les itérations suivantes pourraient engendrer une saturation en mémoire vive, ce qui fausserait nos estimations en temps.
- On évitera, sauf quand cela pourrait avoir de l'intérêt, de monter à plus de 100 secondes pour la résolution d'un seul problème. Cela évite d'une part le risque de saturation en mémoire d'un test isolé. D'autre part chaque figure résulte d'un très grand nombre de résolutions.

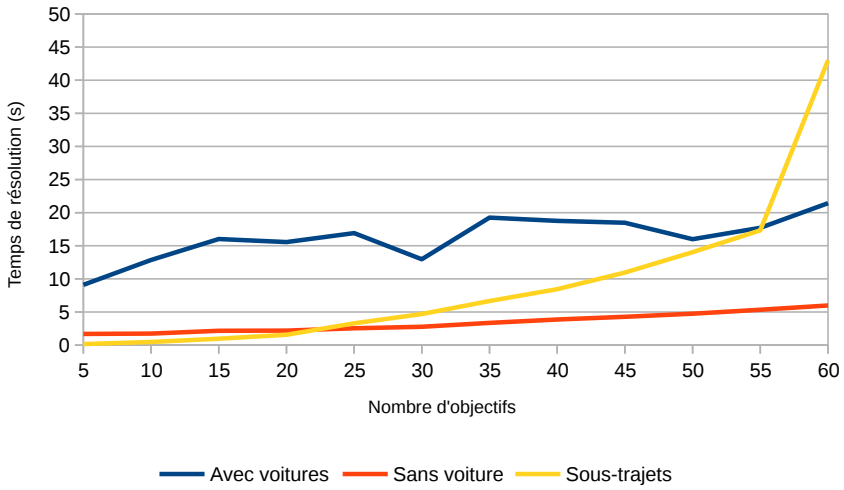
Augmentation du nombre d'objectifs

Nous allons essayer premièrement de voir les réactions de chacun de ces 3 modèles si l'on augmente petit à petit le nombre d'objectifs. La figure C.1a nous montre l'évolution en temps des 3 modélisations sur un type de situation de base prédéfinie. On voit peu à peu le modèle en "sous-trajet" qui initialement était le plus performant, dépasser le modèle en liaison sans voiture. Les courbes de la figure C.1b continu un peu plus loin le processus pour nous laisser apercevoir le moment où ce modèle va même devenir moins performant que le modèle en liaison avec voiture, sensé être le plus complexe.

Explication : Nous savons que les problèmes MA-TSP-PR, même si "multi-agents" ne sont en réalité qu'une multitude de petits TSP-PR



(a) 2 à 32 objectifs.



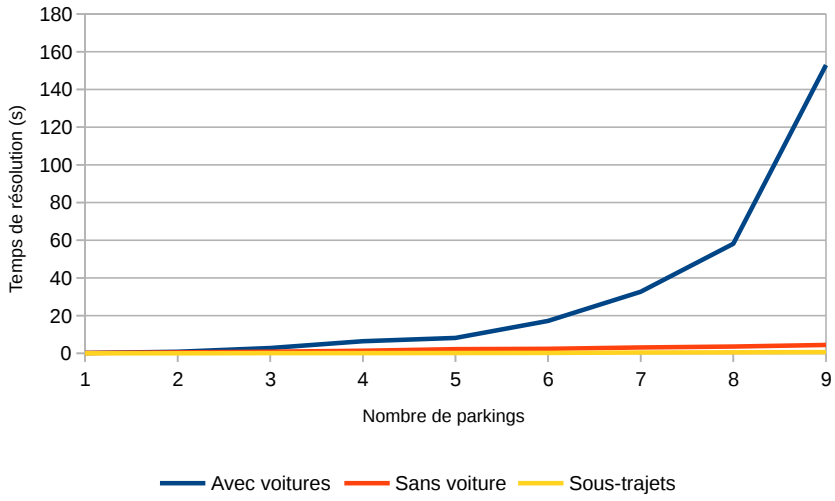
(b) 5 à 60 objectifs (pas de 5).

FIGURE C.1 – Augmentation du temps de résolution en fonction du nombre d'objectifs en MA-TSP-PR, pour 3 entrées, 3 sorties, 5 parkings, 100 agents visitant chacun 2 objectifs.

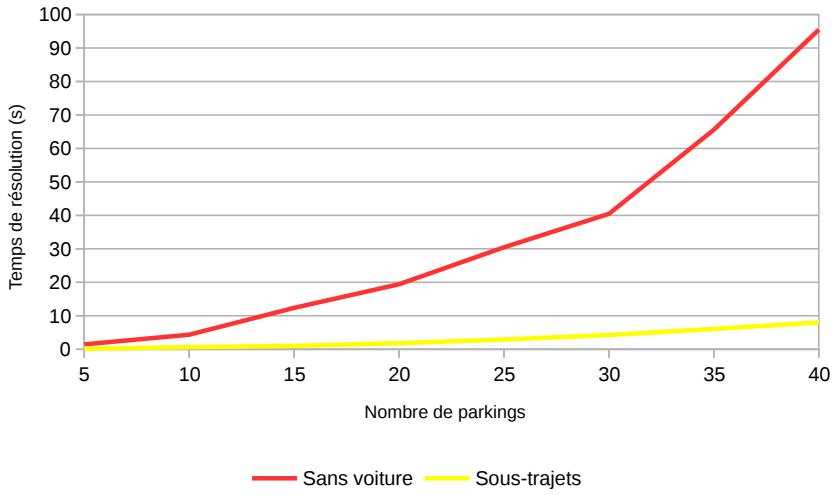
concaténés et n'ayant aucune incidence les uns sur les autres. En ajoutant un à un de nouveaux objectifs, le problème ne sera pas réellement plus complexe. Même si l'on a bien évidement veiller à redéfinir aléatoirement les objectifs ciblés par les agents après chaque ajout d'objectif, il n'en demeure pas moins qu'itération après itération les agents conservent le même nombre d'objectifs. Chaque petit TSP-PR donc va être composé du même nombre d'objectifs ciblés. Cela explique qu'il n'y ait pas une grosse variation de temps de résolution pour les deux premiers modèles. Pour le modèle en sous-trajet, le problème va venir de la quantité excessive de "sous-trajets" qui vont être générés par un ajout d'objectif. En effet, pour 5 parkings et 60 objectifs c'est pas moins de 90.000 sous-trajets de type "Next" qui vont être générés, parmi lesquels chacun des 100 agents devra choisir pour leurs transports inter-objectifs. Ce nombre augmentera linéairement en fonction du carré du nombre de parkings (ou plutôt de $|D| * (|D| - 1)$ plus exactement). Si l'on rajoute un seul objectif en plus on augmente le nombre de sous-trajets de type "Next" de $|D| * |P|^2$ pendant que le nombre de liaisons liées à des objectifs pour les 2 autres modèles augmenteront beaucoup plus lentement (à chaque nouvel ajout d'objectif on rajoute $|D \cup P|$ liaisons).

Augmentation du nombre de parkings

Nous reprenons le même principe que pour le précédent test, mais cette fois-ci en augmentant le nombre de parkings. Tout ce dont nous avons parlé lors de l'analyse des précédents résultats n'est plus valable ici. En effet, même si le MA-TSP-PR n'est que la concaténation d'une multitude de TSP-PR, l'augmentation du nombre de parkings va réellement avoir un impact sur chacun de ces petits problèmes. Nous pouvons donc constater maintenant les réactions face à une vraie augmentation de la complexité du problème de nos différentes modélisations. La première figure, C.2a, nous montre que, assez vite, dès le neuvième parking le modèle utilisant des liaisons avec voitures demande un temps bien supérieur aux deux autres, atteignant les 150 secondes sur l'ordinateur des tests, alors que les deux autres sont toujours sur des temps inférieurs à 10 secondes. D'ailleurs ce temps de 150 secondes ne sera toujours pas atteint par les autres modèles même pour 40 parkings, comme on le voit sur la figure C.2b. Enfin, celui des 3 modèles qui réagit le mieux à cette montée en complexité est celui en sous-trajets, qui surpasse de loin celui en liaison sans voiture (plus de dix fois plus rapide pour 40 parkings).



(a) De 1 à 9 parkings.



(b) De 5 à 40 parkings (pas de 5).

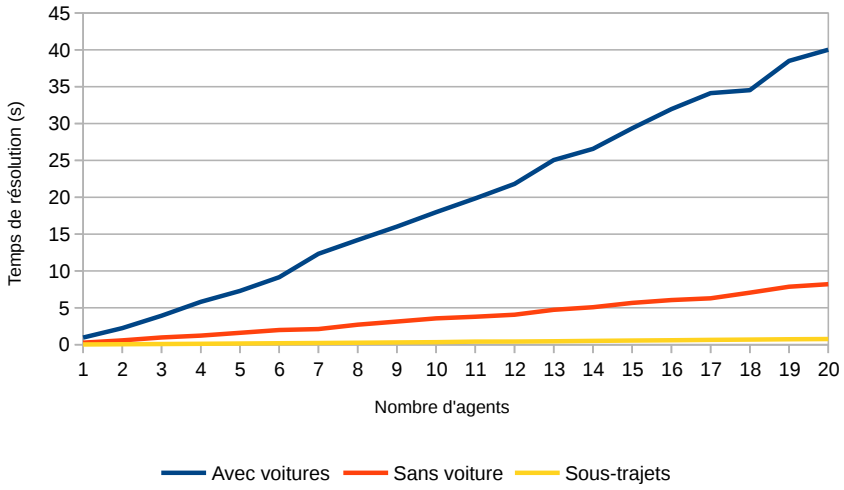
FIGURE C.2 – Augmentation du temps de résolution en fonction du nombre de parkings en MA-TSP-PR, pour 3 entrées, 3 sorties, 5 objectifs, 100 agents visitant chacun 2 objectifs.

Augmentation du nombre d'agents

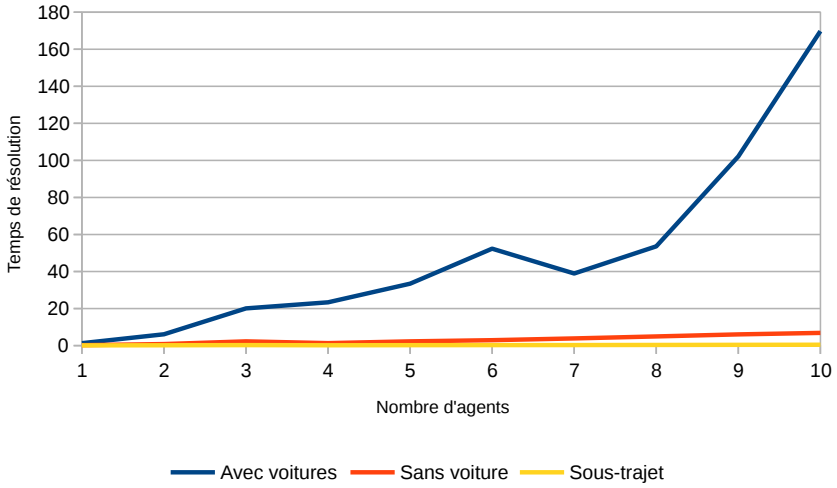
On a rappelé pour le premier test que le problème MA-TSP-PR pourrait, en réalité, n'être considéré que comme plusieurs TSP-PR indépendants. En effet, le trajet de chaque agent dans ce type de problème n'a pas d'impact sur le trajet des autres agents. De manière logique, les temps de résolution des MA-TSP-PR en augmentant le nombre d'agents vont être à peu près linéaires (à quelques bruits près) pour nos 3 méthodes. Des premiers résultats l'attestant sont présentés sur la courbe de la figure C.3a. Cependant, parfois, suivant la complexité des situations de base, surtout pour la méthode par liaison avec voiture, et en particulier s'il y a au moins 3 objectifs ciblés par agent ou plus, des résultats comme ceux présentés en figure C.3a attestent du contraire. Cela peut être dû parfois à du bruit excessif, dû à une instabilité en temps de résolution de ce modèle (très grande variance de temps de résolution) ou tout simplement aussi au fonctionnement interne de CPLEX, qui ne sera peut être pas toujours linéaire pour traiter un grand nombre de problèmes détachés les uns des autres simultanément. Néanmoins, nous resterons toujours, au moins pour les deux autres modèles, et à condition de ne pas faire saturer la mémoire vive, sur des évolutions de temps de résolution qui restent proches d'une fonction linéaire.

Augmentation du nombre d'objectifs par agent

Pour finir, nous constatons sur les figures C.4a et C.4b l'augmentation du temps de résolution en faisant varier le nombre d'objectifs par agent. Il s'agit de la caractéristique la plus influente sur la complexité du problème. En effet, c'est ce nombre d'objectifs qui va établir la longueur des chemins pour chaque agent (la longueur des solutions de leur TSP-PR). Le nombre de chemins possibles à tracer pour chaque agent va augmenter de manière factorielle en fonction de ce nombre d'objectifs. Si bien qu'avec seulement 4 objectifs par agent la modélisation en liaisons simples avec voiture admet déjà des temps de résolutions largement supérieurs à 100 secondes. Parmi les 3 modélisations, la modélisation en sous-trajets admet très significativement les meilleurs résultats.

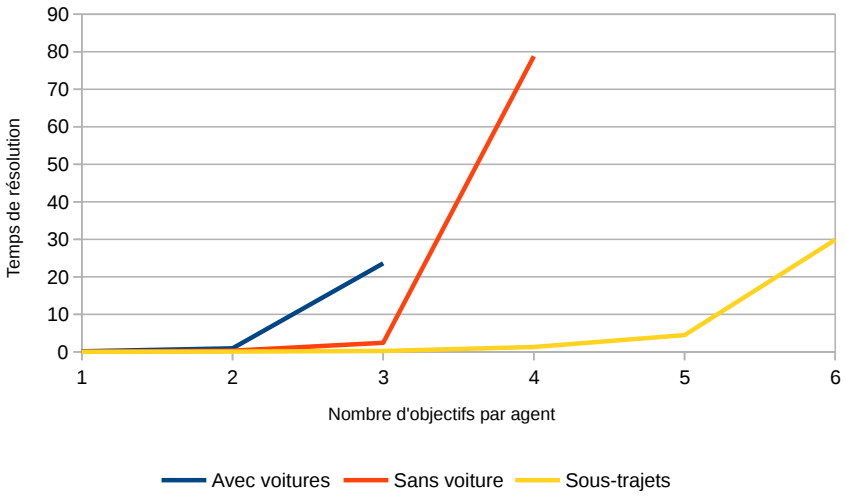


(a) Pour 3 entrées, 3 sorties, 5 objectifs, de 20 à 400 agents visitant chacun 2 objectifs.

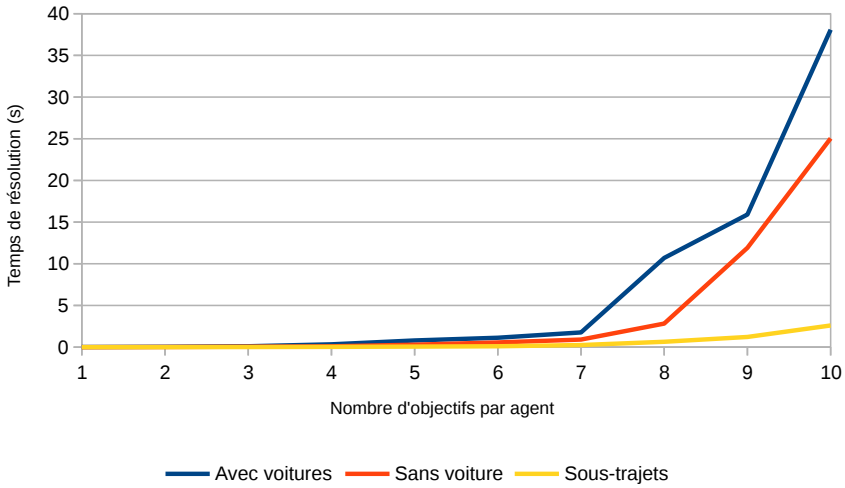


(b) Pour 10 entrées, 10 sorties, 10 objectifs, de 1 à 10 agents visitant chacun 3 objectifs.

FIGURE C.3 – Augmentation du temps de résolution en fonction du nombre d'agent en MA-TSP-PR.



(a) 20 agents visitant de 1 à 6 objectifs.



(b) 1 seul agent visitant de 1 à 10 (tous) objectifs.

FIGURE C.4 – Augmentation du temps de résolution en fonction du nombre d'objectifs par agent, en MA-TSP-PR, pour 3 entrées, 3 sorties, 5 parkings, 10 objectifs.

C.2 Tests pour le MA-TSP-PR-DL et le MA-TSP-PRCL-DL

Dans ces expérimentations numériques nous cherchons à avoir 3 informations :

- Quel modèle entre "Sans voiture" ou "en sous-trajets" est le plus performant en terme de temps de résolution lorsque l'on intègre les possibilités de relocalisation d'objectifs ?
- Comment augmente la complexité (temps de résolution) lorsque l'on intègre les possibilités de relocalisation d'objectifs ?
- Comment augmente la complexité (temps de résolution) lorsque l'on augmente le nombre d'agents dans une situation ou tous les objectifs sont entièrement relocalisables (chaque objectif peut affecter toutes les localisations) ?

Augmentation des possibilités d'affectation

Ces tests consistent à regarder l'impact de l'augmentation d'affectations possibles pour les objectifs sur le temps de résolution. Aussi, de savoir si avec l'ajout des affectations le modèle en "sous-trajet" est toujours plus performant que le modèle "sans voiture". Pour cela nous partons d'une dizaine de situations avec les objectifs à localisation fixée. Nous rajoutons pour chaque objectif une affectation supplémentaire possible à chaque itération. La figure C.5a présente ce test avec les deux modélisations ("sans voiture" et en "sous-trajet"). La différence de temps de résolution entre les deux méthodes s'est amplifiée. Le temps de résolution de la modélisation "sans voiture" commencera déjà à monter à plus de 2 minutes pour l'itération 9 (soit le double de la précédente itération) alors que le modèle en "sous-trajet" est maintenu aux alentours de 0.1 seconde de temps de résolution (0.04 pour l'itération 1 sans relocalisation, 0.14 pour les itérations 8 et 9). Le test suivant, présenté en figure C.5b réalise la même chose pour des situations un peu plus complexes et uniquement en modélisation par "sous-trajet". Il serait difficile d'établir la moindre règle valable quant à cette augmentation, car en fonction du type de situation exacte la courbe constatée pourra être très différente. Cependant, on peut au moins constater au vu de cette courbe que le temps de résolution augmente de manière nettement moins rapide que la complexité du problème. En effet, dans cet exemple le nombre de jeux d'affectations possibles varie entre ces itérations de 1 pour la première

(les localisations sont totalement fixées pour les objectifs) à plus de 3 millions pour la dernière (chaque objectif peut se positionner sur chaque localisation, plus de 3 millions de possibilités). Or sur la dernière itération le temps de résolution n'a été multiplié que par 10 par rapport à la première.

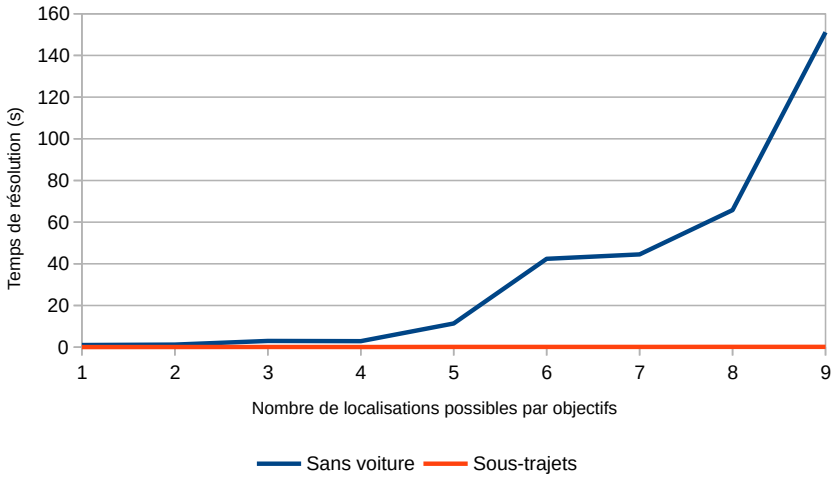
À partir de maintenant, tous nos tests se concentreront uniquement sur le modèle en "sous-trajet", qui est de loin le plus performant.

Augmentation du nombre d'objectifs et de localisations avec possibilités d'affectations complètes

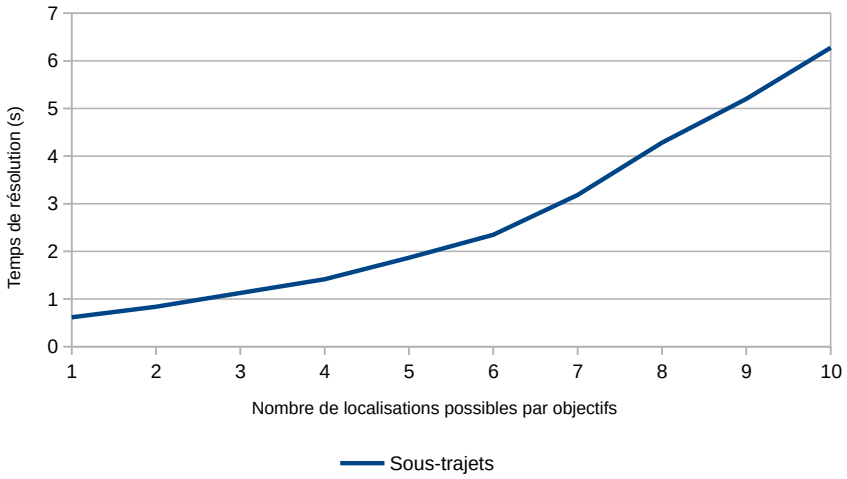
Ici il s'agira simplement de constater l'augmentation conséquente du temps de résolution si l'on augmente de 1 le nombre de localisations et d'objectifs. Nous partons, sur la figure C.6 d'un nombre de localisations et objectifs de 2, et en rajoutons un de chaque sur plusieurs itérations jusqu'à obtenir 12 localisations et objectifs. Les possibilités d'affectations sont complètes, chaque objectif peut affecter chaque localisation. L'augmentation constatée semble au minimum exponentielle, chaque nouvelle ajout multiplie le précédent temps par une valeur comprise entre 2 et 4 (c'est le cas aussi pour les valeurs des premières itérations, même si non visible sur la courbe). Cela nous montre en particulier qu'un nombre conséquent d'objectifs pourra assez facilement rendre notre résolution optimale non-réalisable, si l'on permet à chaque objectifs d'affecter chaque localisation. Si la courbe continue de la même manière pour les itérations suivantes sur ce jeu de tests, il faudrait par exemple attendre plus de 10.000 ans pour optimiser ce type de situation avec 40 localisations et objectifs. Pourtant, il s'agit d'un chiffre qui semble faible si l'on compare au nombre de commerces qu'il pourrait y avoir à localiser dans une zone commerciale. Ces optimisations linéaires ne seront donc pas adaptées à des combinatoires trop complexes où l'on souhaiterait entièrement relocaliser une zone commerciale. Il faudra plutôt dans ce cas avoir une partie des commerces qui sont fixes et de permettre la relocalisation d'un nombre réduit de commerces.

Augmentation du nombre de localisations

Cette fois-ci nous considérons une relocalisation complète, c'est à dire que chaque objectif peut affecter chaque localisation. Nous partons d'un type de situation ayant autant d'objectifs que de localisation, et nous



(a) Pour 5 parkings et 10 agents. De 1 à 9 affectations possibles par objectifs.



(b) Pour 10 parkings et 50 agents. De 1 à 10 affectations possibles par objectifs (10 = toutes les affectations sont possibles).

FIGURE C.5 – Augmentation du temps de résolution en fonction du nombre de d'affectations possibles par objectifs en MA-TSP-PR-DL, pour 3 entrées, 3 sorties, 10 objectifs, chaque agents visitant 2 objectifs.

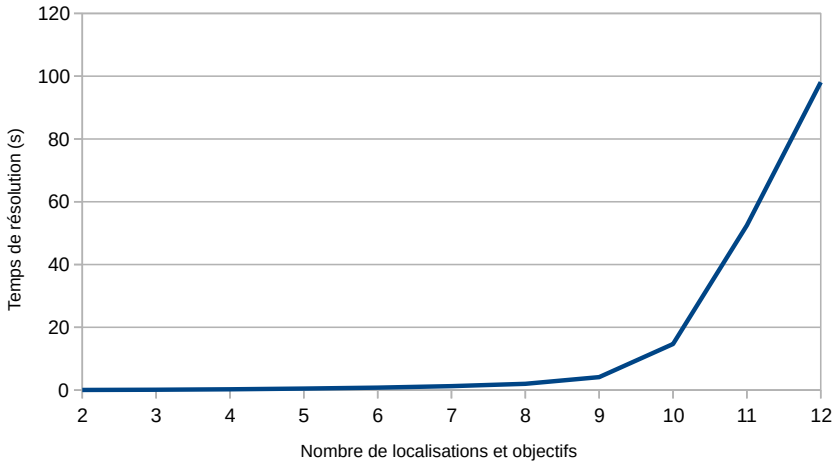


FIGURE C.6 – Augmentation du temps de résolution en fonction du nombre de d'objectifs et localisations en MA-TSP-PR-DL. Pour 3 entrées, 3 sorties, 10 parkings, de 2 à 12 objectifs et localisations, toutes les affectations possibles, 50 agents visitant 2 objectifs.

augmentons itération après itération le nombre de localisation (sans toucher aux objectifs). Chaque ajout d'une localisation est accompagnée de l'ajout des possibilités d'affectations de chaque objectif pour cette dernière. Les résultats de la figure C.7a. On observe pour les premières itérations, jusqu'à 13 localisations pour les 7 objectifs, un comportement plutôt linéaire du temps de résolution. À la 14ème localisation ajoutée, une brusque montée de ce dernier, puis au deux suivants un ralentissement (augmentation moins importante). Ces données sont issues d'une moyenne, sur la résolution d'une dizaine de situation du même type, sur lesquelles pour chacune l'on à ajouté une localisation à chaque itération. Il aurait pu être possible qu'une de ces dizaines de situation, après ajout de la 14ème localisation serait devenue un cas particulier très compliqué à résoudre pour CPLEX ce qui aurait créé un bruit dans les données, mais ce n'est pas le cas ici : chacune des 10 situations à vu son temps de résolution particulièrement augmenté lors cette itération et un peu moins pour les suivantes. Le même test a ensuite été réalisé sur serveur avec plus d'espace disponibles pour la mémoire vive (mais une fréquence un peu moins élevée) et avec 10 nouvelles autres situations du même type. Les résultats de ce deuxième test sont présentés sur la figure C.7b. On retrouve approximativement le même schémas qui s'opère. Cela n'est pas vraiment explicable à notre niveau de connaissance du fonctionnement exacte et complet, interne à CPLEX, pour la résolution, mais cela nous montre au moins justement qu'il est très difficile de pouvoir établir de vraies règles pour pouvoir estimer le temps de résolution d'une de nos

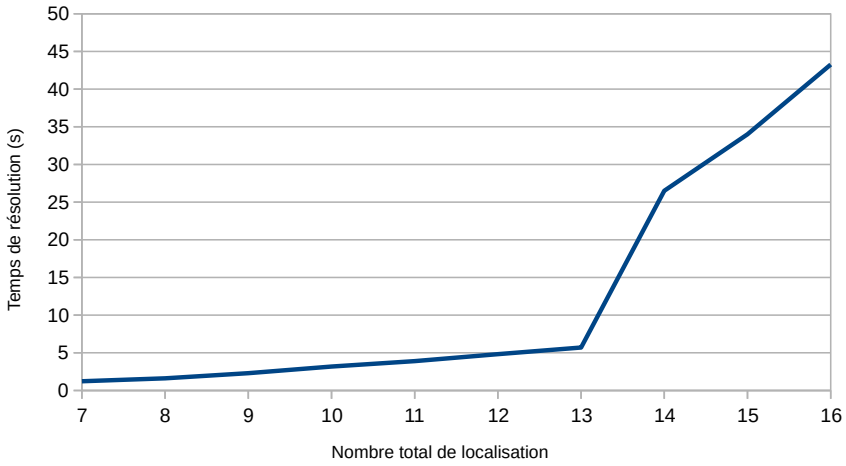
situations. À priori, il semblerait tout de même que l'ajout de localisation seule soit moins problématique que l'ajout d'objectifs précédemment étudié. L'augmentation ne semble pas exponentielle.

Augmentation du nombre d'agent

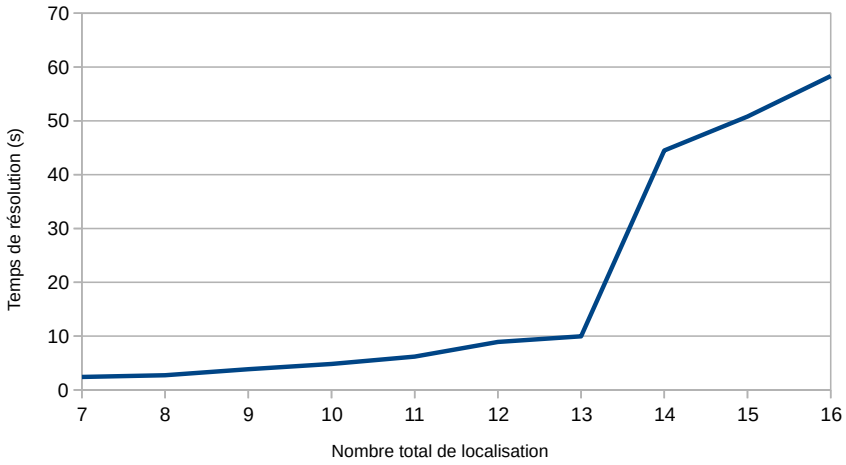
Nous avons déjà réalisé des tests sur l'augmentation du nombre d'agents lors de la section liée au MA-TSP-PR. Le nombre en terme de variables décisionnelles est bien encore proportionnel au nombre d'agents (si on ne compte pas les variables d'affectation, négligeables comparées aux variables liées aux trajets des agents en terme de quantité). Par contre cette fois-ci, les trajets des agents ne sont plus indépendants les uns des autres. Ceci car les affectations des objectifs vont être choisies pour minimiser le coût total des trajets des agents. La figure C.8 montre une courbe de résultats pour un test d'augmentation d'agents. Au premier coup d'oeil, il semblerait que les temps de résolutions soient linéaires en fonction du nombre d'agents.Néanmoins ce n'est pas exact. On pourrait trouver sur cette courbe, plus facilement visible sur les données numériques, deux phases. Jusqu'à 50 agents les temps de résolutions progressent environ de $0.01 \sim 0.015$ secondes par agent supplémentaire, passé les 50 premiers agents ce temps supplémentaire par agent passe en moyenne, au double, ce qui est significatif et non pas simplement du bruit dans les données. Il n'y aura pas de règle précise, cette accélération de temps de résolution sera différente suivant la complexité de notre situation. Par contre, elle restera toujours légère : plus précisément, si l'on double le nombre d'agent, le temps de résolution du nouveau problème devrait dans la plupart des cas être compris entre le double et le triple du temps précédent. Cela semble restait de l'ordre du linéaire.

Nombre maximal de parkings utilisés

On va s'intéresser maintenant au problème MA-TSP-PRL-DL, introduisant donc une notion de capacité et de limite du nombre total de parking. Dans ce premier test on veut savoir si la limite du nombre de parking a un impact sur le temps de résolution. On partira donc cette fois-ci de situations ayant une limite du nombre de parkings utilisés égal au nombre total de parkings, et nous ferons décroître cette limite jusqu'à n'autoriser qu'un seul parking utilisé. Deux exemples sont présentés, la figure C.9a pour des situations très simples et disposant de peu de parkings, et la figure C.9b où nous avons sensiblement augmenté le



(a)



(b) Autre test identique, avec nouveau jeu de données et sur serveur (moins de fréquence mais plus de coeur et de mémoire vive).

FIGURE C.7 – Augmentation du temps de résolution en fonction du nombre de de localisations en MA-TSP-PR-DL, pour 3 entrées, 3 sorties, 10 parkings, 7 objectifs, de 7 à 16 localisations, toutes les affectations possibles, 50 agents visitant 2 objectifs.

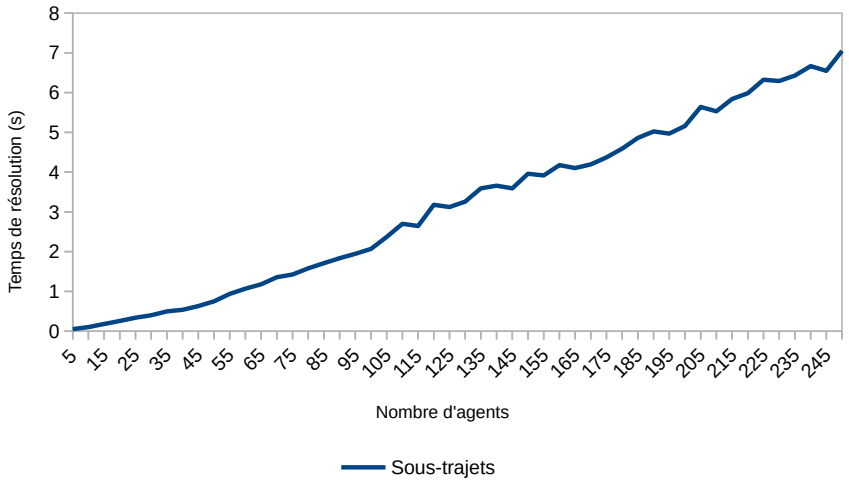


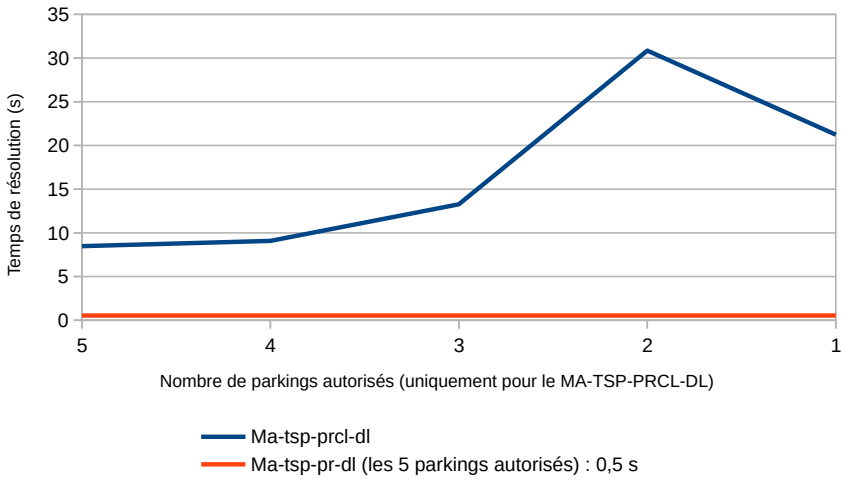
FIGURE C.8 – Augmentation du temps de résolution en fonction du nombre d'agents en MA-TSP-PR-DL. Pour 3 entrées, 3 sorties, 7 parkings, 7 objectifs, 7 localisations, toutes les affectations possibles, de 5 à 250 agents, par pas de 5, visitant 2 objectifs.

nombre de parkings. Sur la deuxième nous avons utilisé l'échelle logarithmique, pour faire ressortir les valeurs importantes. On remarque que l'ajout de ces contraintes, qui pourtant diminue le nombre de solutions non-optimales possibles, augmente sensiblement le temps de résolution. Cela peut varier suivant les complexités de situation, nous trouvons un rapport de presque 20 entre la résolution du MA-TSP-PR-DL de la première figure et son équivalent non-contraignant (limite de parkings utilisés maximum égal au total de parking) en MA-TSP-PRCL-DL, contre un rapport d'environ 3 seulement pour le deuxième exemple (alors qu'il est plus complexe). La limite que l'on fixera aura elle aussi un impact sur le temps de résolution, qui pourra être assez conséquent. Il semblerait d'après nos tests que plus la limite est faible, plus le temps de résolution sera conséquent, à l'exception de "1" même si cette valeur fait parti des plus longues à optimiser. La pire des limites en terme de temps pour la résolution de l'optimisation est donc de 2 parkings.

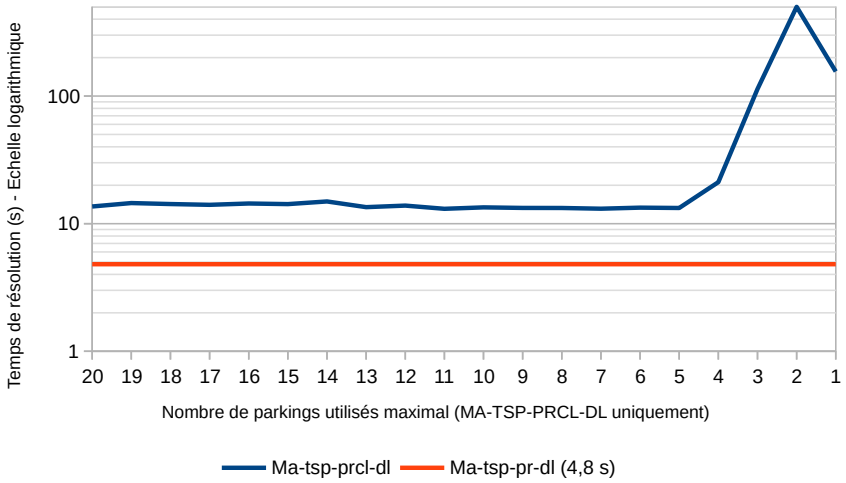
C.3 Tests des simplifications

Les tests de nos simplifications ont été réalisés de la manière suivante :

1. On génère plusieurs situations aléatoires de même complexité (voir Annexe 2 - Génération de tests), avec pour chacune leur fichier de données respectif pour CPLEX, associé à notre modélisation MA-TSP-PR-DL.



(a) Pour 5 parkings, de 5 à 1 parkings utilisés au maximum.



(b) Pour 20 parkings, de 20 à 1 parkings utilisés au maximum.

FIGURE C.9 – Diminution du nombre de parkings utilisés maximal, pour 3entrées, 3 sorties, 7 objectifs, 50 agents visitant 2 objectifs.

2. On effectue pour chaque situation un pré-traitement sur les données pour réduire la liste de liaisons à celles pertinentes uniquement (première simplification). On génère le fichier de données pour CPLEX résultant pour notre modélisation MA-TSP-PR-DL en sous-trajet.
3. On effectue une recherche de la matrice de coût en fonction des trajets individuels optimaux pour chaque situation. On génère les fichiers de données pour notre modélisation d'affectations sans trajet pour le MA-TSP-PR-DL.
4. Pour chaque situation, on résout les 3 modèles.
5. Enfin, on compare les trois méthodes, en fonction du temps nécessaire, pour la résolution du problème.

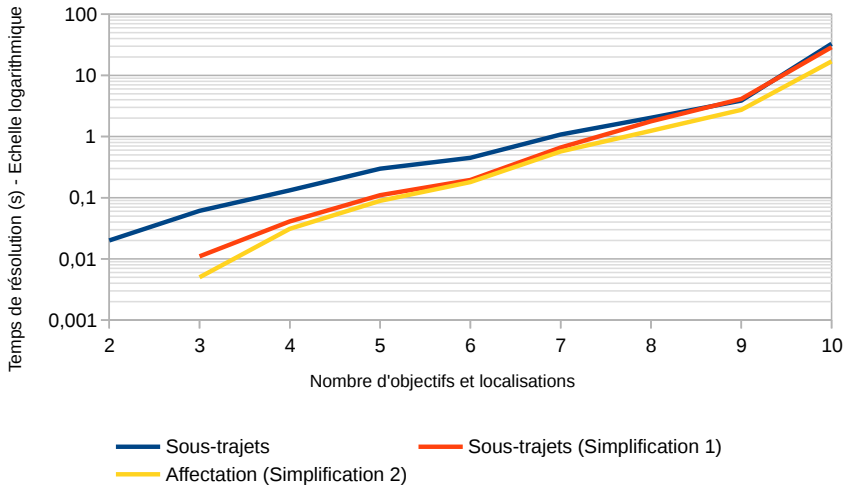
Comparaison avec une augmentation du nombre d'objectifs et localisations

Ce premier test consiste à comparer les temps obtenus lorsque l'on augmente la complexité en nombre d'objectif et localisation (en affectations complètes). La première figure C.10a nous montre des résultats avec un type de situation de base disposant de 2 objectifs par agent, tandis que la suivante, la figure C.10b représente les résultats pour un autre type de situation de base, en particulier où les agents doivent atteindre 3 objectifs.

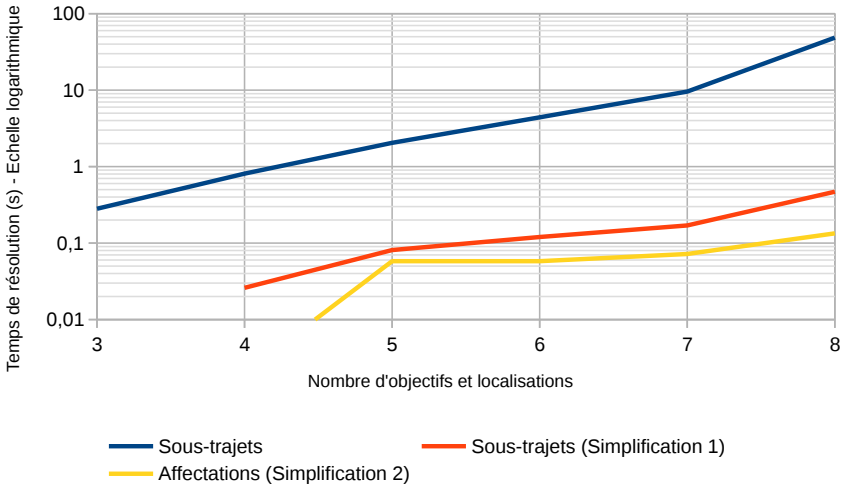
Les résultats de la première courbe sont assez étonnant. Elle nous montre que si nous augmentons fortement le nombre d'objectifs sur une situation de base assez simple, les 3 méthodes semblent à peu près équivalentes. Les simplifications effectuées semblent plutôt inutiles malgré une très forte diminution du nombre de sous-trajets pour la simplification 1 par rapport au modèle en sous-trajets sans pré-traitement. Cependant, dès lors que l'on passe à au moins 3 objectifs, et l'on double le nombre de sous-trajets "Next" à utiliser pour chaque agent (2 au lieu d'1), les simplifications s'avèrent plus que bénéfiques en temps de calcul.

Comparaison avec une augmentation du nombre de parkings

Même test que précédemment, cette fois-ci en jouant sur le nombre de parkings. Résultats présentés sur la figure C.11



(a)



(b)

FIGURE C.10

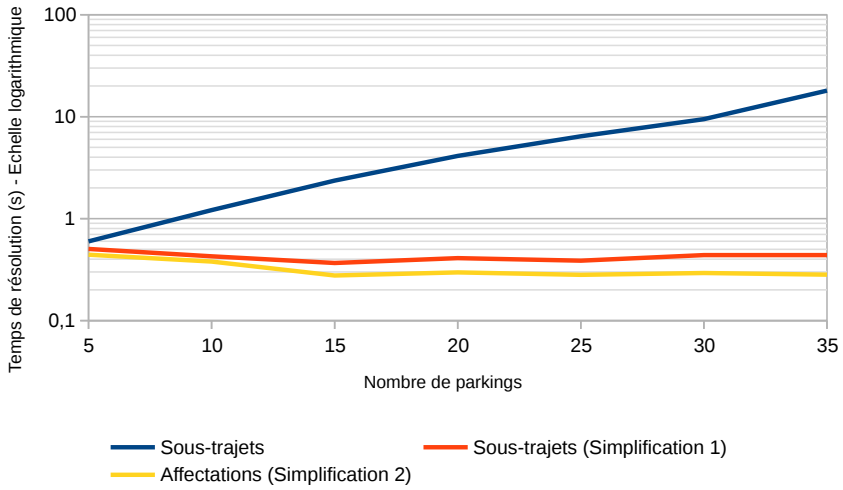


FIGURE C.11

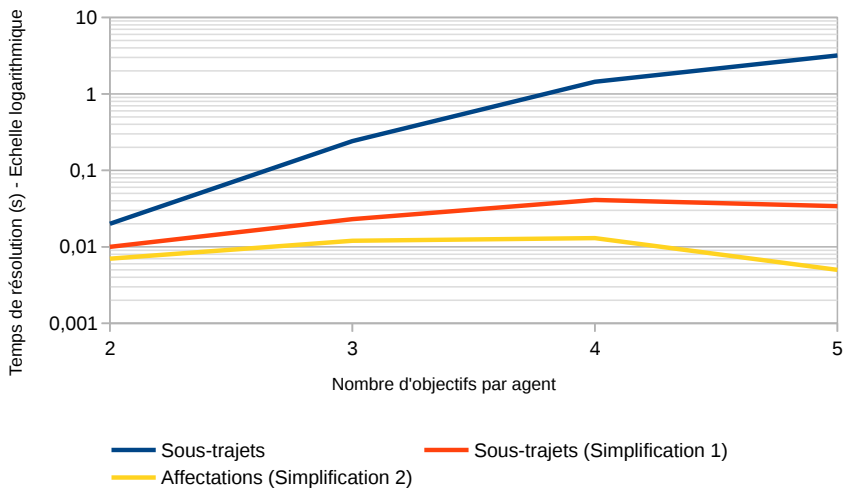


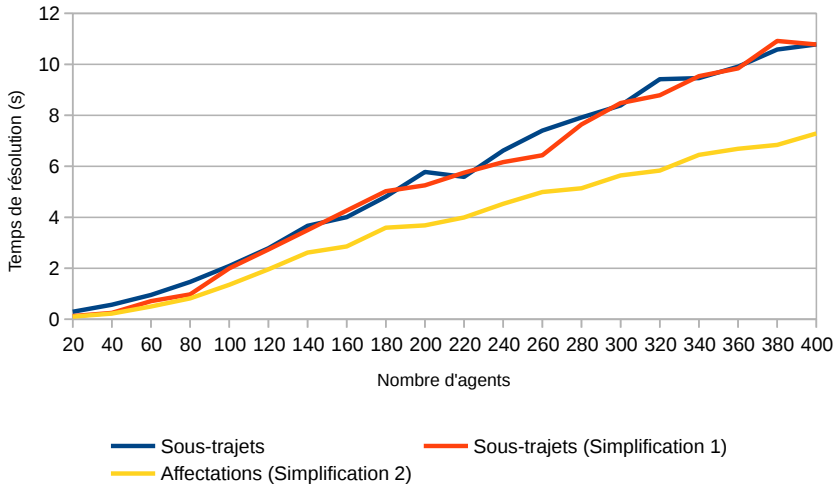
FIGURE C.12

Cette fois-ci il n'y a plus le moindre doute quant à l'utilité des simplifications. La méthode initiale augmente en temps de résolution exponentiellement, dû à la forte augmentation de sous-trajets en fonction du nombre de parkings. Les méthodes avec pré-traitement, que ce soit celle qui élimine les sous-trajets non-pertinents ou celle qui décompose le problème pour ressortir uniquement le problème d'affectation, ont tendance à l'inverse à ne pas modifier leur temps de calcul voir même les réduire pour les premières itérations. Même si la méthode de décomposition (simplification 2) semble plus efficace les 2 méthodes restent d'un même ordre de grandeur.

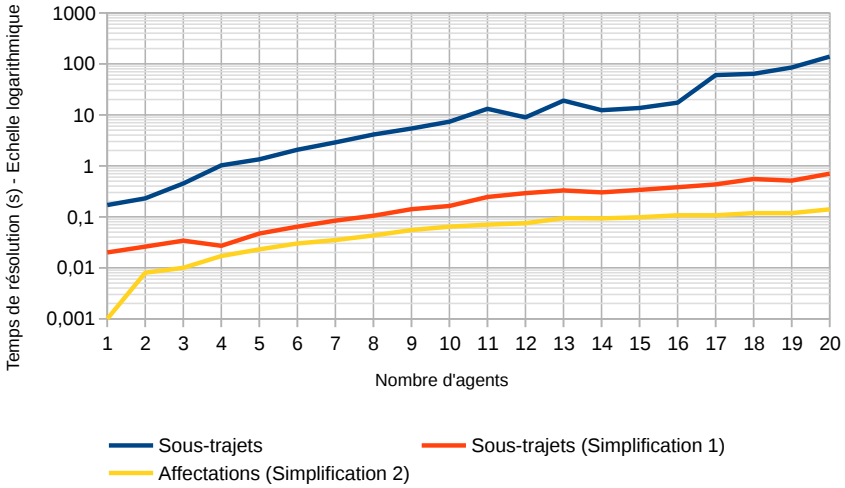
Comparaison avec une augmentation du nombre d'objectifs par agent

Sur la figure C.12 on présente les résultats d'un test où l'on cherche à augmenter le nombre d'objectifs visés par chaque agent.

Sans trop de surprise, en ayant déjà eu un avant goût de l'effet du nombre d'objectifs par agent sur les figures C.10a et C.10b on retrouve une nette différence entre la méthode sans pré-traitement et les deux autres. On notera en particulier sur ce test que la méthode de décomposition amenant vers un problème d'affectation semble se détacher assez nettement. Elle diminue même assez fortement en temps de calcul sur la dernière itération.



(a)



(b)

FIGURE C.13

Comparaison avec une augmentation du nombre d'agents

Pour finir, nous retrouvons dans les figures C.13a et C.13b un test en augmentant le nombre d'agents.

On retrouve des résultats similaires en certains points à nos tests sur les objectifs. C'est à dire que tant que la situation est très simple avec seulement deux objectifs par agents, nos simplifications ne sont pas très efficaces face à l'augmentation d'agents. Par contre dès lors que l'on passe à au moins 3 objectifs par agents, d'une part les simplifications l'une comme l'autre sont largement plus rentables en temps de calcul mais aussi le deviennent de plus en plus au fil de l'augmentation des agents. On notera une petite valeur étrange à la première itération pour la méthode d'affectation. Cette valeur à 0.001 secondes est un cas particulier de ce

modèle, qui, s'il n'y a qu'un agent, n'a simplement qu'à trouver le coût le plus petit parmi un ensemble de valeurs. Si on ignore uniquement cette valeur particulière, la simplification 2 sur la deuxième figure est à la fois celle qui est la plus rapide et à la fois celle qui semble augmenter le plus lentement au fur et à mesure que l'on rajoute des agents.

Comparaison du nombre d'arètes

Ce dernier test ne nous apportera pas vraiment de nouvelles informations utiles mais nous permettra de comprendre pourquoi il y a une différence aussi élevée au niveau du temps de résolution du modèle en sous-trajets avec ou sans pré-traitement de suppression des sous-trajets non-optimaux. En particulier lors d'une augmentation en nombre de parkings. Il s'agira ici simplement, sans aucune résolution de modèle, de comparer le nombre de chaque type de sous-trajets avant et après avoir effectué le pré-traitement. Pour une dizaine de situations nous fixons 50 agents, 2 objectifs par agent, 3 entrées et 3 sorties, 10 parkings, 10 objectifs et 10 localisations (affectations complètes). Sur chacune on simulera un pré-traitement (sans toucher réellement à la situation) ayant pour but de supprimer les sous-trajets non-pertinents. Nous comptons le nombre de chaque type de sous-trajets si le pré-traitement était effectué. Ensuite nous ajoutons 10 parkings à chaque situation et nous effectuons de nouveau la même simulation de pré-traitement. Nous continuons ainsi jusqu'à 200 parkings. Les résultats sont présentés dans le tableau C.1. Les quantités de chaque sous-trajets sans pré-traitement sont fixes qu'importe la génération exacte d'un même type de situation (et sont facilement calculable). Les quantités de sous-trajets après traitement par contre peuvent varier, ce pourquoi on calculera la moyenne pour chaque type de sous-trajets entre les 10 situations d'une itération.

Nb de parkings	Sous-trajets			Sous-trajets pertinents		
	"First"	"Next"	"Last"	"First"	"Next"	"Last"
10	300	9.000	300	44	94,1	43
20	600	38.000	600	44	104	43
30	900	87.000	900	45	106	44
50	1.500	245.000	1.500	44	107	43
100	3.000	990.000	3.000	41	109	40
200	6.000	3.980.000	6.000	38	105	37

TABLE C.1 – Comparaison du nombre de sous-trajets, avant et après suppression des sous-trajets non-pertinents.

Comme on peut le voir, déjà pour seulement 10 parkings à l'origine le nombre de sous-trajets de type "Next" est presque divisé par 100 par le pré-traitement. Ce nombre va très peu varier même en augmentant fortement le nombre de parkings, pour les sous-trajets après le pré-traitement, tandis que le graphe d'origine contient exactement $|P|^2 * |D| * (|D| - 1)$, et donc augmente à la vitesse de $|P|^2$ si l'on augmente $|P|$.

C.4 Tests des heuristiques

Nous générons une centaine de situations à étudier pour chacun des tests pour réduire le bruit qui pourrait être généré à la fois par les heuristiques qui ne sont plus exactes et surtout par celui généré par le placement aléatoire qui fera office de moyenne. Comme précisé dans la précédente section, on commencera tout d'abord pour chaque situation, à lui associer un placement aléatoire (jeu d'affectations) et un placement résultant de l'heuristique. On résout ensuite le MA-TSP-PR-DL de cette situation pour trouver son coût minimal, puis les 2 MA-TSP-PR avec les deux différents placements (aléatoire et heuristique). Une fois les 100 situations traitées nous comparons la somme des coûts trouvés en MA-TSP-PR-DL (valeur optimale dont on aimerait se rapprocher avec les heuristiques) avec la sommes des coûts trouvés en MA-TSP-PR avec placement aléatoire ainsi que celle avec le placement choisi par notre heuristique.

Afin de faciliter ce processus nous testerons en même temps toutes les heuristiques. Donc en réalité nous ne comparerons pas uniquement 3 résultats totaux mais beaucoup plus. Pour chaque situation il y aura donc, en jeux de données : un premier, pour le MA-TSP-PR-DL, un deuxième, celui du MA-TSP-PR en placement aléatoire, et autant en plus que l'on testera d'heuristiques. Les heuristiques 1 et 2 que l'on a vues dans ce chapitre engendreront donc 2 jeux de données à optimiser. Pour la 3ème heuristique, il y a un paramètre α à faire varier, dont on ne connaît pas la valeur idéale. Justement un des buts des tests sur cette troisième heuristique sera, en plus de savoir si elle est efficace, de connaître le paramètre α pour lequel elle est la plus efficace. Nous ferons varier ce paramètre entre 0.2 et 1.9 par palier de 0.1. Ce qui nous rajoutera 18 jeux de données supplémentaires à obtenir pour pouvoir tracer la courbe de performance et de LOT de la troisième heuristique en fonction de α. Pour chaque situation testée il y aura donc en tout 1 MA-TSP-PR-DL et 20 MA-TSP-PR à résoudre.

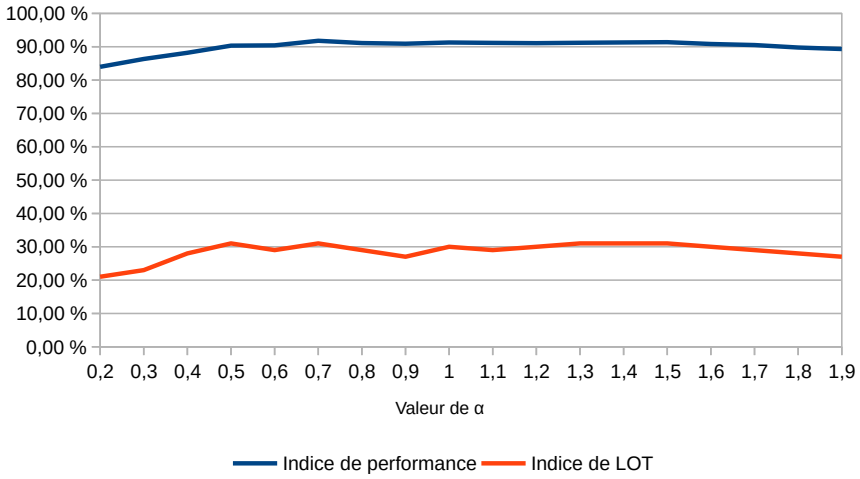


FIGURE C.14 – Indice de performance et de LOT en fonction de α - Test A.

• Test A : 5 Entrées et Sorties - 10 Parkings - 6 Objectifs et Localisations - 100 agents - 2 Objectifs par agents

Heuristique 1 :

Indice de performance : 12.3%

Indice de LOT : 0

Heuristique 2 :

Indice de performance : 77.4%

Indice de LOT : 14%

Heuristique 3 :

Meilleur α pour la performance : 0.7

Indice de performance pour $\alpha = 0.7$: 91.8%

Indice de LOT pour $\alpha = 0.7$: 31%

Meilleurs α pour l'indice de LOT : 0.5, 0.7, 1.3, 1.4, 1.5, pour un indice de LOT à 31%.

Courbe complète des résultats en figure C.14.

• Test B : 3 Entrées et Sorties - 5 Parkings - 5 Objectifs et Localisations - 10 agents - 2 Objectifs par agents

Heuristique 1 :

Indice de performance : 16.5%

Indice de LOT : 2%

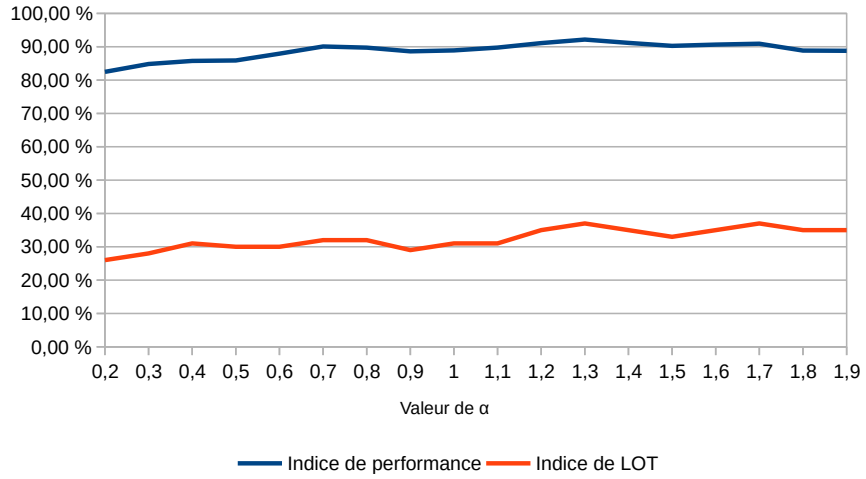


FIGURE C.15 – Indice de performance et de LOT en fonction de α - Test B.

Heuristique 2 :

Indice de performance : 69.6%

Indice de LOT : 16%

Heuristique 3 :

Meilleur α pour la performance : 1.3

Indice de performance pour $\alpha = 1.3$: 92.2%

Indice de LOT pour $\alpha = 1.3$: 37%

Meilleurs α pour l'indice de LOT : 1.3 et 1.7, pour un indice de LOT à 31%.

Courbe complète des résultats en figure C.15.

• Test C : 3 Entrées et Sorties - 5 Parkings - 5 Objectifs et Localisations - 10 agents - 3 Objectifs par agents

Heuristique 1 :

Indice de performance : 13.1%

Indice de LOT : 0

Heuristique 2 :

Indice de performance : 59.9%

Indice de LOT : 14%

Heuristique 3 :

Meilleur α pour la performance : 1.2

Indice de performance pour $\alpha = 1.2$: 91.2%

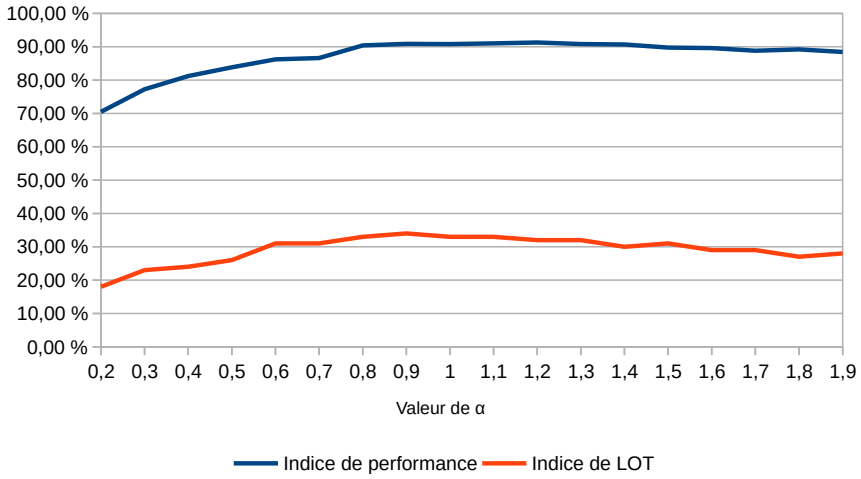


FIGURE C.16 – Indice de performance et de LOT en fonction de α - Test C.

Indice de LOT pour $\alpha = 1.2$: 32%
Meilleur α pour l'indice de LOT : 0.9 pour un indice de LOT à 34%.
Courbe complète des résultats en figure C.16.

- **Moyenne pour les 3 tests**

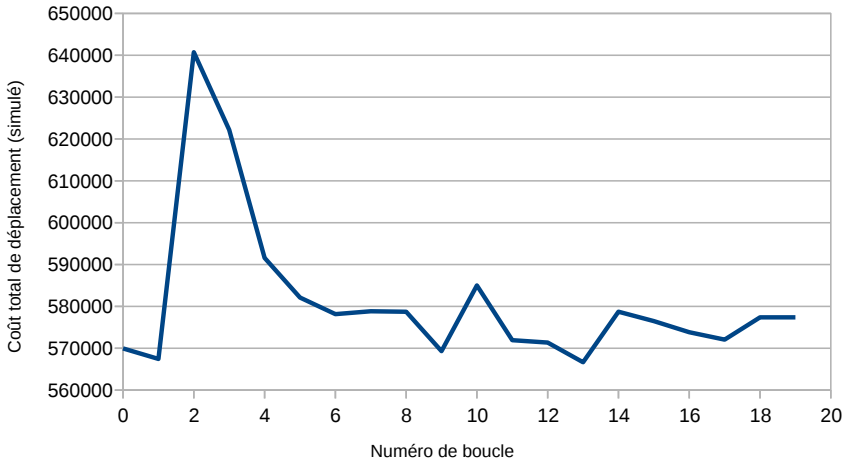
Heuristique 1 :
Indice de performance : 13.97%
Indice de LOT : 0.67%

Heuristique 2 :
Indice de performance : 68.97%
Indice de LOT : 14.67%

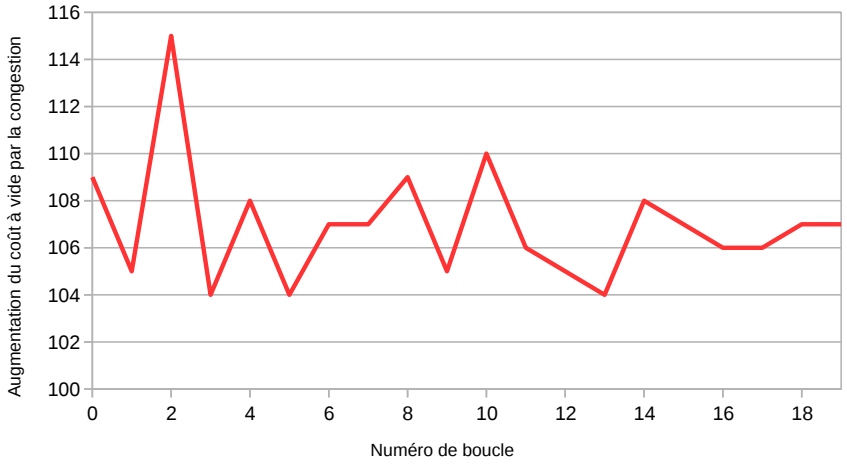
Heuristique 3 :
Meilleur α pour la performance : 1.3
Indice de performance pour $\alpha = 1.3$: 91.38%
Indice de LOT pour $\alpha = 1.3$: 33.33%
Meilleur α pour l'indice de LOT : 1.3 pour un indice de LOT à 33.33%.

C.5 Tests du bouclage CPLEX-MatSim sur la zone étudiée du Pontet

C.5.1 6 Parkings



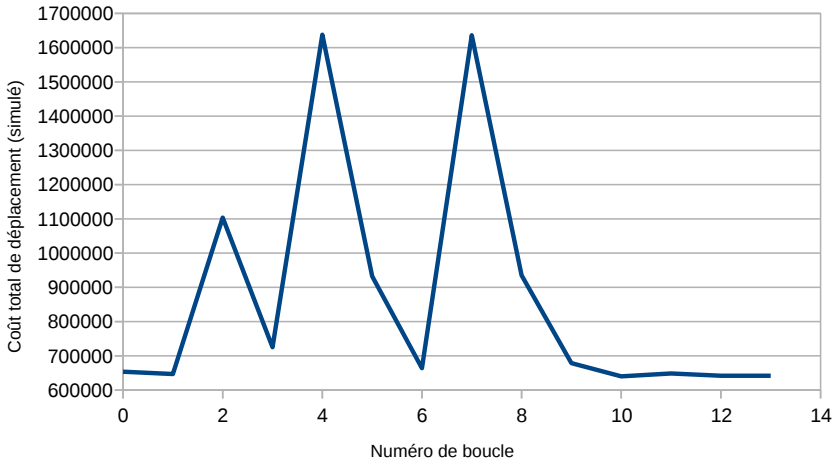
(a) Évolution du coût total



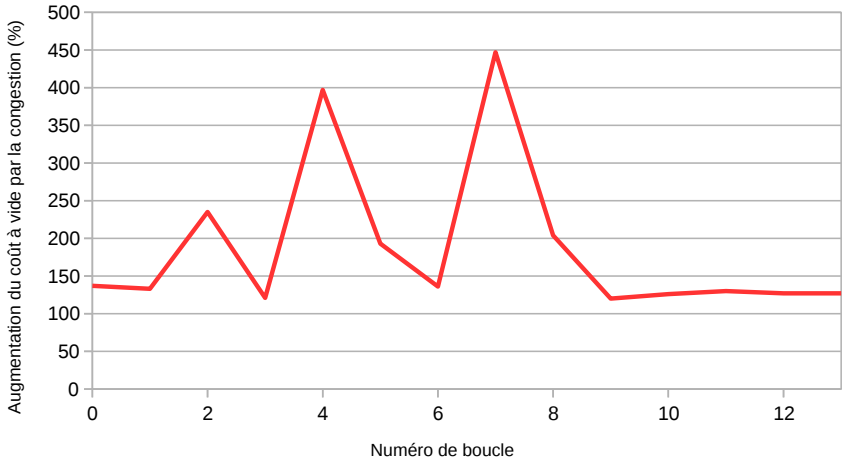
(b) Évolution de la congestion

FIGURE C.17 – Évolution du coût total trouvé par la simulation d'un agencement du problème du Pontet au fil des boucles, pour 6 parking.

C.5.2 5 Parkings



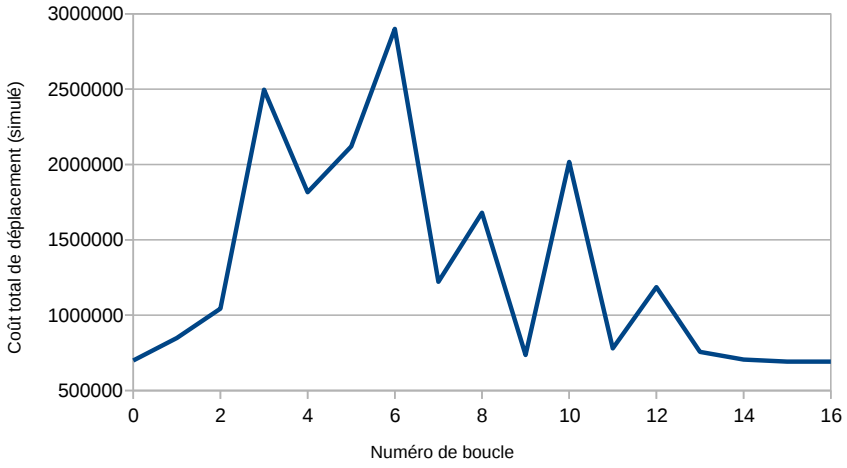
(a) Évolution du coût total



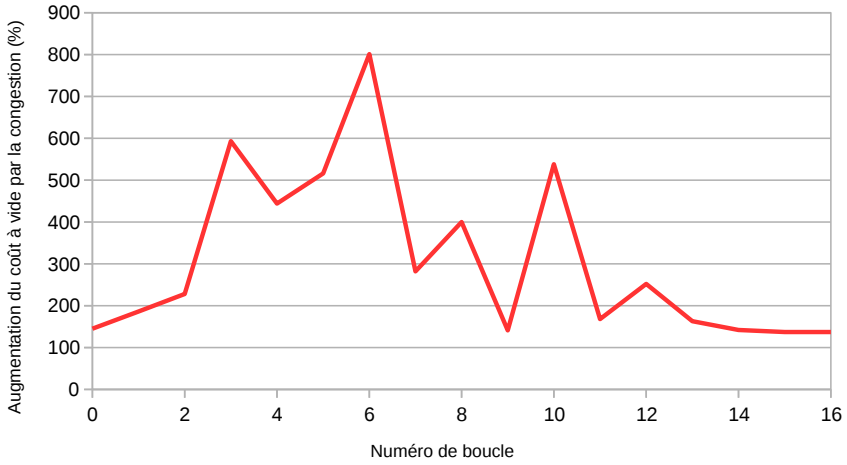
(b) Évolution de la congestion

FIGURE C.18 – Évolution du coût total trouvé par la simulation d'un agencement du problème du Pontet au fil des boucles, pour 5 parking.

C.5.3 4 Parkings



(a) Évolution du coût total



(b) Évolution de la congestion

FIGURE C.19 – Évolution du coût total trouvé par la simulation d'un agencement du problème du Pontet au fil des boucles, pour 4 parking.

Liste des illustrations

Liste des tableaux

Bibliographie

Ouvrages de référence

Baray, Jérôme, Qian Ding et Abdelkader Abdellaoui (jan. 2013). « Intégration d'un Modèle d'Implantation Commerciale Stratégique au sein d'un SIG ». In :

Benenson, Itzhak, Karel Martens et Slava Birfir (2008). « PARKAGENT : An agent-based model of parking in the city ». In : *Computers, Environment and Urban Systems* 32.6. GeoComputation : Modeling with spatial agents, p. 431-439. issn : 0198-9715. doi : https://doi.org/10.1016/j.compenvurbsys.2008.09.011. url : http://www.sciencedirect.com/science/article/pii/S0198971508000689.

Communauté métropolitaine de Montréal (2013). *Recueil d'exemples de bonnes pratiques en aménagement de stationnement.*

Darbéra, Richard (1999). « Du mauvais usage des politiques de stationnement pour réguler la circulation ». In : *Transports* 395, p. 166-171.

Ferber, Jacques (1997). « Les systèmes multi-agents : un aperçu général ». In : *Techniques et sciences informatiques* 16.8.

Garcez, Cristina et David Mangin (2014). « Du Far West à la ville ». In : *L'urbanisme commercial en questions, Parenthèses Éditions.*

Glasnapp, James et al. (2014). « Understanding Dynamic Pricing for Parking in Los Angeles : Survey and Ethnographic Results ». In : sous la dir. de Fiona Fui-Hoon Nah, p. 316-327.

Gutin, Gregory et Abraham P Punnen (2006). *The traveling salesman problem and its variations.* T. 12. Springer Science & Business Media.

Horni, Andreas, Kai Nagel et Kay W Axhausen (2016). *The multi-agent transport simulation MATSim.* Ubiquity Press London.

Horni, Andreas, Darren M. Scott et al. (2009). « Location Choice Modeling for Shopping and Leisure Activities with MATSim : Combining Microsimulation and Time Geography ». In : *Transportation*

Research Record 2135.1, p. 87-95. DOI : 10.3141/2135-11. URL : https://doi.org/10.3141/2135-11.

LA Express Park (2012). URL : http://www.laexpresspark.org/.

« Les crises de la grande distribution » (2016). In : *Revue française de socio-Economie*.

Levy, Nadav et Itzhak Benenson (2015). « GIS-based method for assessing city parking patterns ». In : *Journal of Transport Geography* 46, p. 220-231. ISSN : 0966-6923. DOI : https://doi.org/10.1016/j.jtrangeo.2015.06.015. URL : http://www.sciencedirect.com/science/article/pii/S0966692315001052.

Mladenović, Nenad et al. (2007). « The p-median problem : A survey of metaheuristic approaches ». In : *European Journal of Operational Research* 179.3, p. 927-939. ISSN : 0377-2217. DOI : https://doi.org/10.1016/j.ejor.2005.05.034. URL : http://www.sciencedirect.com/science/article/pii/S0377221706000750.

Moati, Philippe (2011). *Nouvelle Révolution commerciale (La)*. Odile Jacob.

Pierce, Gregory et Donald Shoup (2013). « Getting the Prices Right ». In : *Journal of the American Planning Association* 79.1, p. 67-81. DOI : 10.1080/01944363.2013.787307. URL : https://doi.org/10.1080/01944363.2013.787307.

Porter, Richard et al. (2013). « Optimisation of Car Park Designs ». In : *Research report. University of Bristol*, p. 47. URL : http://www.maths-in-industry.org/miis/726/.

SF Park (2010). URL : http://www.sfpark.org/.

SOeS, Union Européenne (2012-2018). *Données des Répartitions des superficies de changements CLC 2012-2018*. URL : https://www.statistiques.developpement-durable.gouv.fr/corine-land-cover-0.

— (1990-2018). *Données des Répartitions des superficies : CLC 2000, CLC 2006 révisé, CLC 2012 révisé et CLC 2018*. URL : https://www.statistiques.developpement-durable.gouv.fr/corine-land-cover-0.

Waraich, Rashid et Kay Axhausen (déc. 2012). « Agent-Based Parking Choice Model ». In : *Transportation Research Record Journal of the Transportation Research Board* 2319, p. 39-46. DOI : 10.3141/2319-05.

AVIGNON UNIVERSITÉ

Approches mathématiques pour l'aménagement de zones commerciales : modèles linéaires, algorythmes et systèmes multi-agents

par
SAHUC Cyril

Résumé : Abstract à écrire

Mots-clés : Localisation, Problème du Voyageur de Commerces, Zone Commerciale, Parking, Commerce, Matsim

Keywords : Location, Travelling Salesman Problem, Commercial Area, Parking, Commerce, Matsim

LABORATOIRE INFORMATIQUE D'AVIGNON