Problèmes de Tournées de Véhicules avec Synchronisation et Optimisation Multicritère avec l'Intégrale de Choquet

Fabien Lehuédé

Université de Nantes
IRCCyN, UMR CNRS 6597 (Institut de Recherche en Communications et Cybernétique de Nantes)

Dossier présenté en vue de l'obtention du diplôme d'

Habilitation à Diriger les Recherches

Proposé par

Fabien Lehuédé

Problèmes de Tournées de Véhicules avec Synchronisation et Optimisation Multicritère avec l'Intégrale de Choquet

HDR soutenue le 10 juillet 2015 devant le jury composé de :

Jean-François Cordeau	HEC Montréal	(Rapporteur)
Frédéric Semet	École Centrale de Lille	(Rapporteur)
Daniel Vanderpooten	Université Paris Dauphine	(Rapporteur)
Dominique Feillet	École des Mines de Saint Étienne	(Examinateur)
Michel Gendreau	École Polytechnique de Montréal	(Examinateur)
Pascale Kuntz	Université de Nantes	(Examinatrice)

Table des matières

Liste des figures

Chapitre 1
Introduction

Ce document présente une synthèse de l'ensemble des travaux réalisés, encadrés et en cours depuis ma soutenance de thèse en novembre 2003. Mon domaine de recherche, en recherche opérationnelle et aide à la décision, porte d'une part sur l'optimisation multicritère et d'autre part sur l'optimisation des réseaux de transport.

Mes travaux en optimisation multicritère ont été initiés durant ma thèse. Elle s'est déroulée de 2000 à 2003 en contrat CIFRE au centre de recherche du groupe Thales (TRT : Thales Research & Technology) et au Laboratoire Informatique de Paris 6 (LIP6) sous la supervision de Michel Grabisch, Christophe Labreuche et Pierre Savéant. Cette thématique a été poursuivie à TRT en tant qu'ingénieur de recherche pendant trois ans, puis à mon arrivée dans l'équipe Systèmes Logistiques et de Production (SLP) de l'IRRCyN en 2006 à travers des collaborations régionales et internationales.

Je travaille sur l'optimisation de réseaux de transport de manière croissante depuis mon arrivée dans l'équipe SLP. L'essentiel de mes contributions porte sur la résolution de problèmes de tournées de véhicules pour le transport de personnes, marchandises et de matières premières. Les problèmes sur lesquels j'ai travaillé partagent pour la plupart une caractéristique d'actualité dans le domaine : la présence de contraintes de synchronisation entre véhicules. Une seconde problématique en développement concerne la conception de réseaux de transport. Cette dernière est principalement étudiée sous l'angle collaboratif entre fournisseurs cherchant à mutualiser leurs achats de transports vers des clients communs.

Ces activités de recherche sont synthétisées ci-dessous et plus amplement détaillées dans ce manuscrit.

1.1 Optimisation combinatoire multicritère avec l'intégrale de Choquet

Le paradigme de l'optimisation combinatoire considère en général une unique fonction objectif à maximiser ou minimiser. Néanmoins dans de nombreux cas pratiques, on cherche une solution qui offre un bon compromis entre plusieurs fonctions objectif. En présence d'un nombre restreint d'objectifs, l'optimisation multiobjectif permet de construire un ensemble de solutions non dominées (front de Pareto) dans lequel un décideur pourra choisir. Au delà de deux critères, la création et l'analyse du front de Pareto peut être extrèmement complexe. Une solution consiste alors à tenir compte des préférences du décideur pendant la résolution du problème combinatoire. On peut alors accélérer la recherche de solutions et se concentrer sur les solutions qui intéressent le décideur. Ceci implique de modéliser les préférences du décideur sur les différentes valeurs pouvant être prises par les fonctions objectif. Cette problématique relève

du champs de l'Aide MultiCritère à la Décision (AMCD). Dans ce domaine, les chercheurs ont développé de nombreux modèles mathématiques, des procédures et des outils qui permettent la modélisation des préférences d'un humain sur un ensemble d'alternatives représentées par leurs valeurs selon différents attributs, quantitatifs et qualitatifs.

Mes travaux on principalement porté sur l'intégration d'un des modèles d'AMCD les plus généraux dans des outils et algorithmes d'optimisation combinatoire. Ce modèle suit le cadre général de la théorie de l'utilité multiattribut et peut être décomposé en deux parties : en premier lieu, à chaque attribut (fonctions objectif) est associé une fonction d'utilité monocritère. Pour une solution donnée, cette fonction établi son niveau de satisfaction selon le critère associé et permet d'exprimer l'ensemble des critère sur une même échelle. Dans la seconde partie, les critères sont agrégés pour établir la valeur de la solution. Je me suis intéressé à une fonction d'agrégation multicritère particulièrement flexible : l'intégrale de Choquet. De nombreux travaux ont montré que les capacité de modélisation de cette fonction en faisaient un outil adapté à la modélisation des préférences. Néanmoins l'intégrale de Choquet est un outil complexe pour l'optimisation : une importance est associée à chaque coalition de critère et le poids sélectionné pour un critère dans l'agrégation dépend de son classement vis-à-vis des autres.

Ma thèse a porté sur l'intégration de l'intégrale de Choquet en optimisation combinatoire dans un paradigme de programmation par contraintes. J'ai travaillé en premier lieu sur le développement d'une contrainte globale pour l'intégrale de Choquet en développant des algorithmes de filtrage incrémentaux permettant une propagation efficace de la contrainte dans le solveur (Lehuédé et al. 2002; 2006a) J'ai ensuite conçu un framework permettant de guider la recherche de solutions dans les problèmes étudiés (Lehuédé et al. 2003a; 2006b). Il est basé sur l'alternance de recherches suivant des stratégies mono-critères avec un choix intelligent de la stratégie à utiliser en fonction de la solution courante. Les différents algorithmes développés ont été utilisés pour des recherches de solutions complètes et incomplètes et mis en œuvre sur des problèmes académiques (planification d'examens) et industriels (placement d'applications de traitement du signal sur machines parallèles). Le framework de recherche a fait l'objet d'un brevet national Lehuédé et al. (2003b) étendu à l'international en 2006.

Ces travaux ont été poursuivis à Nantes dans le cadre du projet régional MILES de la fédération Atlanstic. Ils ont fait l'objet du stage de master et de la thèse de Hugo Fouchal (2008–2011) (Fouchal 2011), co-encadrée avec Xavier Gandibleux au Laboratoire Informatique de Nantes Atlantique (LINA). La thèse porte sur la recherche de plus-courts chemins Choquet-optimaux. Plusieurs algorithmes d'étiquetages ont été conçus pour permettre la recherche d'une solution optimale selon l'intégrale de Choquet : dans des problèmes de plus courts chemins d'un sommet source à un sommet puits (Fouchal et al. 2011b), en premier lieu ; puis pour des problèmes de plus courts chemins d'un sommet source à tous les autres (Fouchal et al. 2011a). Ces travaux présentent des bornes et relations de dominance pouvant être étendues à des problèmes autres que le plus court chemin.

Ma contribution la plus récente sur ce thème porte sur l'intégration de l'intégrale de Choquet dans une métaheuristique de type Adaptive Large Neighborhood Search (ALNS) pour la résolution d'un problème d'optimisation de tournées de transport de personnes handicapées (Lehuédé et al. 2013). Ces travaux ont été réalisés avec Olivier Péton de l'équipe SLP dans le cadre d'un projet Hubert Curien Amadeus en collaboration avec Sophie Parragh et Fabien Tricoire de l'université de Vienne. Ils présentent la modélisation de préférences multicritères pour l'agrégation de cinq objectifs liés au coût et à la qualité de service. Différents opérateurs ont été proposés pour

faciliter la recherche de solutions préférées et surtout augmenter la robustesse des algorithmes. Ces derniers travaux font le lien avec un autre thème de recherche développé au sein de l'équipe SLP : l'optimisation de tournées de véhicules.

1.2 Optimisation de tournées de véhicules

Depuis mon intégration dans l'équipe SLP en 2006, la plus grande partie de mon activité de recherche est réalisée en transport, avec un accent plus particulier sur l'optimisation de tournées de véhicules. J'ai débuté cette thématique avec la conception du logiciel Marika (2007-2008) qui réalise l'optimisation des tournées de transport des usagers des établissements (Intituts Médico-Educatifs et Etablissements et Services d'Aide par le Travail) de l'association ADAPEI 44 (Lehuédé et al. 2008; 2009). Ce logiciel a été développé dans l'équipe SLP en collaboration avec Olivier Péton, Claire Pavageau et Piere Dejax. Sa spécification, en relation avec les directeurs des établissements de l'ADAPEI 44, nous a ouvert à deux problématiques de recherche qui ont été traitées par la suite : la construction de tournées avec transferts ; et la conception de tournées régulières.

Lors de la spécification de Marika, un cas particulier, non traité, a été présenté à l'équipe projet : un directeur de deux établissements proches réalise une forme de massification en planifiant le transport de ses usagers à l'établissement le plus proche. Il planifie ensuite des «navettes» (en utilisant les mêmes véhicules), qui conduisent les passagers transportés à un établissement intermédiaire à leur destination finale. En collaboration avec Olivier Péton, nous avons généralisé ce mode de fonctionnement et formalisé le problème de transport à la demande avec transferts, qui a fait l'objet de la thèse de Renaud Masson (2009–2012). Cette thèse a permis la réalisation de nombreux travaux intéressants. Une première étude fait abstraction de la contrainte de temps de trajet maximum inhérente au transport de personnes : le problème de tournées de collectes et livraisons avec transferts (PDPT : Pickup and Delivery Problem with Transfers) a été résolu avec une métaheuristique ALNS (Masson et al. 2013a). Plusieurs opérateurs de retraits et d'insertion des requêtes de transport dans une solution on été proposés. Le sous-problème de la réalisabilité temporelle d'une insertion dans un ensemble de tournées comportant des transferts et des fenêtres de temps a été plus particulièrement étudié, avec la proposition d'un algorithme de réalisabilité en temps constant (Masson et al. 2013b). Le cas particulier du problème avec navettes, où aucune collecte n'est réalisée entre deux livraisons, a été résolu avec un algorithme de Branch & Cut & Price (Masson et al. 2014b). Celui-ci a été conçu en collaboration avec Stefan Ropke lors d'un séjour de Renaud Masson à l'université technologique du Danemark. Le problème de transport à la demande a fait l'objet du développement d'une recherche Tabou relativement simple en première étude (Masson et al. 2011). Le sous-problème de réalisabilité des contraintes temporelles a été modélisé comme un problème de planification temporelle et résolu comme un problème de détection de cycle de longueur négative dans un graphe. Sa résolution est accélérée par la vérification de conditions nécessaires et de conditions suffisantes de réalisabilité (Masson et al. 2012). Une métaheuristique ALNS a été développée (Masson et al. 2014a). Son utilisation pour la résolution de problèmes de transport à la demande de la littérature montre qu'elle est légèrement plus efficace que les algorithmes de l'état de l'art, à la fois sur des temps de résolution courts et d'autres moins contraints. L'apport des transferts a été évalué pour des instances de la littérature et des instances réelles. Dans les deux cas, celui-ci est très dépendant de l'instance. Les réductions de cout maximales sont de l'ordre de 9%.

La seconde problématique découlant du logiciel Marika porte sur la conception de tournées de transport régulières. En effet, le logiciel réalise des tournées sur une semaine (5 jours), le planning hebdomadaire étant répété toute l'année. Les usagers ne sont pas tous présents tous les jours. Un souci lié à cette variabilité est de ne pas présenter un nombre élevé d'horaires significativement différents à chaque personne. Ce problème a été étudié dans le cadre du projet CTouVer supporté par le GdR RO et réalisé en collaboration avec Dominique Feillet et Thierry Garaix de l'École des Mines de Saint Étienne, mon collègue Olivier Péton et Dominique Quadri de l'Université d'Avignon. On introduit un modèle original pour la modélisation de la régularité basé sur des classes temporelles. Un algorithme de recherche à voisinage large est introduit : à chaque itération, les tournées d'une journée sont détruites et reconstruites. Pour la reconstruction, une heuristique basée sur un Branch & Price résout un problème complexe de tournées de véhicules avec des fenêtres de temps multiples et sans temps d'attente. La méthode permet d'exhiber un front Pareto des solutions non dominées selon les critères coût et régularité. Des tests réalisés sur des instances réelles et des instances générées montrent l'efficacité de la méthode et l'impact de la régularité sur le coût (Feillet et al. 2014).

Les algorithmes de Renaud Masson ont par ailleurs été étendus pendant sa thèse dans le cadre d'une étude en logistique urbaine en collaboration avec Anna Trentini, Nicolas Malhéné et Houda Tlahig de l'EIGSI à la Rochelle. Cette étude concerne la livraison de petits colis en centre ville de La Rochelle à l'aide de containers spécialisés, transportés via une ligne de bus et des tricycles électriques (Trentini et al. 2012, Masson et al. 2015). Dans la suite de ces travaux et plus généralement dans la suite de la thèse de Renaud Masson, je co-encadre la thèse de Philippe Grangier, débutée en 2012, en collaboration avec Michel Gendreau et Louis-Martin Rousseau du CIRRELT. Cette thèse est focalisée sur les systèmes de distribution faisant intervenir une synchronisation entre plusieurs tournées. A ce jour, deux problématiques ont été étudiées. La première est un problème de tournées de véhicules à deux échelons, routes multiples et contraintes de synchronisation issu d'un modèle innovant de distribution en logistique urbaine. Un algorithme ALNS a été proposé pour résoudre ce problème. Les principales contributions méthodologiques portent sur la création de nouveaux opérateurs de recherche et sur l'extension des algorithmes de réalisabilité conçus pour le PDPT (Grangier et al. 2014). A cette occasion il a été montré que le test de réalisabilité en temps constant permettait de réduire d'un facteur 12, en moyenne, le temps de résolution de l'ALNS par rapport à une approche plus classique. La seconde problématique actuellement à l'étude par Philippe Grangier est le problème de tournées de véhicules avec cross-dock et contraintes de capacité. On y modélise la capacité du cross-dock comme un nombre maximum de véhicules pouvant être simultanément traités, d'une part pour le déchargement des marchandises, et d'autre part pour leur rechargement dans les véhicules qui réalisent la distribution.

La notion de synchronisation est également présente dans la thèse d'Axel Grimault, que je co-encadre depuis 2012 en collaboration avec Nathalie Bostel. Cette thèse fait partie du projet FUI ORLoGES. Elle a pour but la construction de tournées de collectes et livraisons en camions pleins entre différents sites (plateformes de stockage, recyclage et/ou production, chantiers, carrières, décharges,...). Sur certains sites, les véhicules ne peuvent être servis simultanément en raison de contraintes matérielles. Celles-ci peuvent être inhérentes au déchargement, qui doit être synchronisée avec l'application pour les enrobés, ou même au chargement (par la pelleteuse présente sur le site par exemple). Les travaux ont fait l'objet d'une heuristique en deux phases permettant le découpage des requêtes en camions pleins et la construction d'un premier en-

semble de tournées (Grimault et al. 2014). Un algorithme ALNS et une nouvelle extension des algorithmes de réalisabilité du PDPT ont été réalisés. Un article est en cours de rédaction.

1.3 Conception de réseaux de transport mutualisés

Mes travaux sur la conception de réseaux de transport ont été initiés en collaboration avec Narendra Jussien dans le cadre du projet Logistique Mutualisée Durable (LMD), financé de 2008 à 2009 par l'ADEME dans le cadre du PREDIT 3. L'étude se base sur l'expérience de mutualisation des entreprises Bénédicta, Nutrimaine (Banania) et Pastacorp (Lustucru). Le problème d'optimisation abordé est celui de la conception d'un réseau de transport mutualisé permettant à des producteurs géographiquement proches d'organiser ensemble la distribution de leurs produits vers leurs clients communs. Cette distribution est réalisée par l'intermédiaire de hubs logistiques et de tournées comprenant plusieurs points de collecte et/ou de distribution. On étudie ici un flux annuel, la mutualisation des transports permettant à la fois de réduire les coûts et d'augmenter la fréquence de réalisation des tournées. La sous-traitance du transport dans ce réseau fait apparaître des structures de coût (grilles tarifaires) et des contraintes originales en optimisation (détour maximum et nombre de clients maximum par tournées). Le problème a été résolu par une extension de l'heuristique de localisation DROP (Daskin 1995). A chaque itération de l'heuristique, pour chaque ensemble de hubs considérés, les producteurs sont affectés au plus proche. L'ensemble des tournées est considéré en extension. La fréquence et le coût de chaque tournée sont déterminés. Un Programme Linéaire en Nombres Entiers (PLNE) permet de déterminer les tournées à utiliser. La réduction de l'ensemble des tournées à l'aide de règles de dominance a été étudié dans le cadre du stage de master d'Arthur Brunel en collaboration avec Olivier Péton. L'étude LMD devait être poursuivie par une seconde phase qui n'a pas été déposée. Ce travail nous a néanmoins ouvert d'intéressantes perspectives.

En premier lieu, un contrat industriel a été réalisé pour le compte de sept producteurs de végétaux de la région angevine en collaboration avec Olivier Péton et la société 4S-Network (co-fondée par Xavier Perraudin, ex-directeur logistique Bénédicta et participant à LMD). Le contrat a permis la simulation de la distribution mutualisée de plus d'un an de commandes en provenance d'environs 5 000 clients répartis sur le territoire français. Celle-ci est réalisée via un Centre de Distribution et Consolidation (CDC) et six Centres de Distribution Régionaux (CDR). Des tournées de navettes appartenant aux producteurs sont conçues pour l'approvisionnement du CDC. Des tournées de camion complet pouvant réaliser jusqu'à deux collectes au CDC ou chez les producteurs permettent d'approvisionner les CDR. La distribution depuis le CDC et les CDR est ensuite sous-traitée avec des coûts qui suivent une grille tarifaire. Pour chaque période de l'historique de commandes, le problème de conception des tournées et plans de transport est résolu par une décomposition en trois PLNE (Lehuédé et al. 2014). Cette approche permet de résoudre le problème avec des temps très réduits et une distance très faible à l'optimum. L'étude a permis d'estimer les réductions de coût inhérentes à la mutualisation. Quatre des producteurs ont lancé sa mise en œuvre, supportée par le projet FUI VEGESUPPLY (2013–2016). Ce projet finance SLP pour deux ans de post-doctorat réalisé par Xin Tang. Les recherches en cours portent sur la localisation des CDR et la réalisation d'un module opérationnel. Ce dernier doit réaliser quotidiennement les tournées de navettes et de camions complet à destination des CDR.

Parallèlement au FUI VEGESUPPLY et dans la suite de LMD, la société 4S–Network développe le concept de CRC (Centre de Routage Collaboratif). Ce service a pour but de faciliter

la mutualisation des transports entre fournisseurs de la grande distribution. Dans ce cadre, un contrat industriel a été passé avec 4S-Network pour supporter la thèse de Juliette Médina, débutée en octobre 2013. L'objet de cette thèse, co-encadrée avec Olivier Péton, porte sur la conception d'outils d'aide à la décision pour la conception du service et l'optimisation de son fonctionnement, ainsi que la réalisation de simulations à l'intention des fournisseurs. Un premier modèle a été proposé pour optimiser le parcours de commandes depuis la sortie des usines jusqu'au distributeur. Le réseau considéré comporte également des entrepôts de préparation et des plateformes de cross-dock. Le modèle intègre notamment des coûts de transport de différentes natures, ainsi que les coûts de transferts sur les plateformes. Ce travail est en cours de publication.

1.4 Plan du document

Le manuscrit comporte deux principaux chapitres : le chapitre 2 synthétise les travaux réalisés en optimisation multicritère ; le chapitre 3 synthétise ceux en tournées de véhicules. La synchronisation entre tournées constitue un aspect original qui se retrouve dans la majeure partie des problèmes traités. J'ai donc pris le parti de présenter ces travaux sous cet angle avec un effort d'analyse et d'unification des algorithmes proposés.

Mes contributions en conception de réseaux de transport mutualisés sont plus récentes (la première publication vient d'être soumise). Le Chapitre 4, présente donc une synthèse, mais surtout les perspectives, de ces travaux.

Chaque chapitre situe les travaux réalisés dans les différents domaines, comporte sa propre conclusion et ses perspectives. Une conclusion générale est donnée en Chapitre 5.

Chapitre 2

Modélisation des Préférences et Optimisation Combinatoire Multicritère avec l'Intégrale de Choquet

Ce chapitre présente l'essentiel des travaux réalisés et supervisés dans la continuité de ma thèse sur l'intégration d'un modèle d'Aide MultiCritère à la Décision (AMCD) en programmation par contraintes. Dans une première partie, je présente les motivations inhérentes aux travaux, puis quelques méthodes d'aide à la décision multicritère. Suivent ensuite les travaux issus de la thèse d'Hugo Fouchal en Section 2.3 sur la recherche de chemins Choquet–optimaux dans les graphes. La Section 2.4 présente l'intégration d'un modèle multicritère dans une métaheuristique pour la résolution d'un problème de transport de personnes.

Par souci de simplification, la présentation de l'AMCD se veut réductrice dans ce chapitre, en se focalisant sur mes principales contributions, mais également en se plaçant du point de vue des problématiques abordées en optimisation combinatoire. Cette présentation est également le fruit de réflexions liées à sept années de découverte de la communauté « Logistique et Transport » et des approches de collègues, chercheurs et industriels dans ce domaine (ou quelques autres) pour la résolution de problèmes comportant plusieurs objectifs.

2.1 Motivations

De nombreux problèmes d'optimisation combinatoires peuvent se résumer à l'optimisation d'un unique critère, souvent économique, sous réserve de satisfaire un ensemble de contraintes. Ces dernières peuvent aussi bien concerner des limites matérielles ou structurelles, liées à l'exploitation de ressources par exemple, que des limites en terme d'acceptabilité de la solution sur des aspects qualité de service, gestion des ressources humaines, ... Néanmoins dans de nombreuses situation, on cherche une solution de bon compromis entre différents objectifs de natures variées et contradictoires : la solution idéale, optimale pour l'ensemble des objectifs, n'existe pas.

Ces vingt dernières années, un effort important a été fourni pour résoudre des problèmes à deux ou trois objectifs, en trouvant un ensemble de solutions non dominées, dites également Pareto-optimales ou efficaces (Ehrgott 2005). La visualisation du front des solutions efficaces est tout à fait parlante dans le cas bi-objectif. Néanmoins, dès que le nombre est supérieur, le choix d'une solution par un décideur devient complexe (Branke et al. 2008). Dans le cas de l'optimisation de tournées de véhicules, on peut souligner quelques méthodes récentes permettant d'aborder ces problèmes (Jozefowiez et al. 2009, Parragh et al. 2009, Paquette et al. 2013). Néanmoins les temps de calculs inhérents à la construction d'un front Pareto sont conséquents comparativement à une approche mono-objectif.

Les objectifs qui ont guidés ma recherche en optimisation multicritère sont donc :

1. **rendre possible la recherche d'une solution dans des problèmes combinatoires ou des préférences sont exprimées sur un nombre important p d'objectifs ($p \geq 3$) ;**
2. **exploiter ces préférences pour rendre la recherche d'une solution préférée plus performante**.

De nombreuses approches et en particulier les méthodes interactives (Vanderpooten and Vincke 1989, Branke et al. 2008) poursuivent des buts similaires. L'approche suivie dans les travaux que j'ai menés ou encadrés consiste à exploiter les résultats obtenus en Aide MultiCritère à la Décision (AMCD) sur la modélisation mathématique des préférences d'une personne (appelé décideur). Cette approche est dite « *a priori* » : les préférences du décideur sont tout d'abord modélisées ; puis un algorithme d'optimisation combinatoire recherche une solution préférée selon ce modèle.

2.1.1 Les limites de la somme pondérée

L'approche la plus intuitive et probablement encore la plus utilisée, lorsqu'on désire optimiser simultanément plusieurs objectifs, consiste à les agréger dans une somme pondérée. La popularité de cette approche résulte probablement de deux importants facteurs d'attractivité pour le concepteur d'un outil :

Tout d'abord, la somme pondérée est très bien comprise par le commanditaire du logiciel, voire même parfois suggérée ou imposée par celui-ci. En gardant la main sur les valeurs des pondérations, le futur utilisateur a le sentiment qu'il pourra facilement obtenir la solution de compromis désirée. Il évite également de cette manière d'avoir à discuter au cours d'un projet déjà complexe, de préférences plus ou moins rationnelles, à la fois en interne et avec le concepteur de l'outil. La compréhension d'un modèle mathématique plus complexe constitue de même un frein, rendant potentiellement plus difficile la justification du choix de la solution de compromis.

En second lieu, la somme pondérée est une solution de facilité évidente pour le concepteur de l'outil. Le modèle est simple et il s'intègre facilement dans la plupart des algorithmes d'optimisation. Mais surtout, en laissant l'utilisateur fixer les poids des objectifs, on s'affranchit de la mise en place d'un processus d'interaction complexe qui lui permettrait d'exprimer ses préférences.

Néanmoins, les travers de la somme pondérée sont multiples et son utilisation pose souvent de nombreux problèmes :

1. Les poids de la somme pondérée permettent à la fois de compenser les différences d'échelle des critères et d'exprimer leurs importances relatives.
2. Il n'existe pas nécessairement un jeu de poids permettant de trouver une solution de bon compromis, qui peut se trouver dans une zone concave du front de Pareto.
3. Le modèle peut être très sensible à de petites variations des poids.
4. La somme pondérée suppose que la satisfaction du décideur augmente de manière linéaire lorsqu'on améliore un critère.
5. Elle suppose que chaque critère contribue indépendamment de la valeur des autres à la qualité de la décision.

Dans le cadre d'un logiciel d'optimisation on peut noter de plus qu'il n'y a aucune garantie qu'un jeu de poids satisfaisant pour un jeu de données le soit pour un autre. Il y a notamment une grande sensibilité à la taille du problème traité. Ce problème reste néanmoins sous-jacent à toute fonction d'agrégation multicritère. Le choix de la somme pondérée, qui a semblé pertinent au départ, se transforme alors souvent en un casse-tête au moment de régler les poids du modèle.

2.1.2 Quelques mots sur l'aide multicritère à la décision

L'AMCD a pour objectif d'aider un ou plusieurs décideurs à prendre une décision et cherche donc à réaliser une recommandation en accord avec les préférences de celui ou ceux-ci. Les méthodes placent le décideur au centre de l'approche et doivent faire face à de nombreuses difficultés, contribuant autant à aider le décideur à formuler ses préférences, qu'à les modéliser.

L'AMCD considère en général un ensemble connu et fini X d'objets ou *alternatives* sur lesquels doivent porter la décision. Roy (1985) distingue trois types de problématiques : le *choix* d'une alternative considérée comme la meilleure ; le *tri* des alternatives dans un ensemble de classes prédéfinies ; le *classement* des alternatives de la meilleure à la plus mauvaise.

Pour réaliser cette recommandation, les méthodes les se basent sur l'identification et la structuration d'un ensemble de *points de vue*, aussi appelés *attributs* ou *métriques*, pris en compte par le décideur pour comparer les alternatives les unes aux autres. Ceux-ci servent de base pour formuler des *critères*, qui établissent une valeur attribuée à un objet selon un point de vue donné. Un critère est donc caractérisé par une fonction g telle que $g(x) > g(y)$ si l'alternative x est préférée à y selon le point de vue considéré.

La recommandation est ensuite basée sur une *agrégation* des critères qui permet de modéliser la *relation de préférence*, notée $\succeq$ entre les solutions. Pour celle-ci, on distingue deux écoles :

- Les méthodes de *surclassement* (Roy 1968), qui reposent sur une comparaison par paires de l'ensemble des alternatives selon chaque critère, suivi d'une agrégation de ces comparaisons.

- Les méthodes issues de la théorie de l'utilité (ou de la valeur) multiattribut (MAUT : MultiAttribute Utility Theory (Keeney and Raiffa 1976)), qui construisent une fonction de valeur $U : X \to \mathbb{R}$ telle que $U(x) \geq U(y)$ si l'alternative x est jugée au moins aussi bonne que y ($x \succeq y$).

Un principe de base s'applique pour toutes les écoles : ce n'est pas le décideur qui règle lui-même les paramètres du modèle afin qu'il modélise ses préférences. Celles-ci sont identifiées avec l'aide d'un *analyste* au cours d'un processus *d'élicitation des préférences*. La recherche en AMCD porte donc également sur le processus, les méthodes et outils d'élicitation des préférences, qui intègrent entre autres les algorithmes d'apprentissage et la détection d'incohérences dans l'expression des préférences du décideur.

Dans la perspective d'une intégration en optimisation combinatoire, les approches de surclassement et MAUT présentent des propriétés très différentes. Les algorithmes d'optimisation combinatoire sont focalisés sur le parcours le plus restreint possible de l'ensemble des solutions du problème, hors les méthodes de surclassement supposent que la comparaison de chaque paire de solutions suivant chaque critère est préalable à l'agrégation. De nombreuses approches visant à optimiser une fonction issue d'une modélisation des préférences en AMCD se basent donc sur l'approche MAUT, plus naturelle dans ce contexte. L'intégration revient alors à rechercher une solution de valeur maximale pour une fonction objectif complexe.

2.1.3 Précédentes contributions

Ma thèse a porté sur l'intégration en programmation par contraintes d'un modèle multicritère de type MAUT basé sur une fonction d'agrégation dont le pouvoir d'expression est particulièrement utile en AMCD : l'intégrale de Choquet (Choquet 1953). Cette thèse CIFRE a été financé par le groupe Thales dont les chercheurs Michel Grabisch (maintenant Professeur à Paris I) et Christophe Labreuche ont significativement contribué au développement d'outils et modèles basés sur l'intégrale de Choquet en AMCD (Grabisch 1995; 1997, Grabisch and Labreuche 2005; 2010).

Mes principales contributions ont porté en premier lieu sur des règles générales et des algorithmes incrémentaux pour la propagation de l'intégrale de Choquet lorsqu'elle est utilisée dans une recherche arborescente (et en particulier dans le cadre de la programmation par contraintes) (Lehuédé et al. 2006a). En second lieu, j'ai travaillé sur les stratégies de recherche de solutions dans ce même contexte (Lehuédé et al. 2006b).

2.1.4 Contributions présentées dans ce chapitre

Ce chapitre porte sur des contributions réalisées depuis ma thèse sur l'optimisation de l'intégrale de Choquet. Les travaux les plus poussés, présentés en Section 2.3, concernent la recherche de plus courts chemins Choquet–optimaux. La Section 2.4 présente la mise en œuvre et l'optimisation d'un modèle d'AMCD basé sur l'intégrale de Choquet avec un algorithme ALNS pour un cas de transport à la demande multicritère.

Par souci de cohérence du mémoire, je ne détaillerai pas mes quelques contributions sur la modélisation des préférences réalisés à Thales Research & Technology après ma thèse dans le cadre du développement du logiciel d'AMCD Myriad (Labreuche and Lehuédé 2005, Lehuédé and Labreuche 2006).

2.2 Modélisation des Préférences en Aide à la Décision Multicritère

Cette section présente en premier lieu le cadre MAUT, illustré par quelques méthodes basées sur la somme pondérée pour la partie agrégation. La dernière partie présente l'intégrale de Choquet.

La modélisation des préférences n'étant pas l'objet de ce manuscript, je me concentrerai sur la présentation des modèles, en n'évoquant que très brièvement les méthodes. Une présentation vaste du domaine peut être trouvée dans l'ouvrage de synthèse de Figueira et al. (2005). Concernant l'utilisation de l'intégrale de Choquet en AMCD, l'état de l'art de Grabisch and Labreuche (2010) reste la meilleure référence à ce jour. Le mémoire de thèse de Brice Mayag (Mayag 2010) présente également un grand nombre d'avancées récentes sur la construction du modèle. Le mémoire d'habilitation à diriger les recherches de Patrick Meyer (Meyer 2013) présente également une vision originale de l'AMCD avec en particulier une comparaison de l'expressivité des modèles et un focus sur les outils logiciels indispensables à la construction des modèles.

2.2.1 Cadre théorique : MAUT

Considérons un modèle multicritère à p *critères*, comportant un unique niveau d'agrégation. Les ensembles de définition des *métriques* sur lesquelles sont construits ces critères sont notés $\Omega_i, i \in \{1, \ldots, p\}$. X désigne l'ensemble des *alternatives* du problème.

Une première étape consiste à établir la commensurabilité entre les critères en les représentant sur une *échelle commune de satisfaction*, généralement $[0, 1]$. Cette opération est réalisée via l'introduction de *fonctions d'utilité monocritères* $u_i : \Omega_i \to [0, 1]$, pour tout critère $i \in \{1, \ldots, p\}$. Notons $(x_1, \ldots, x_p) \in \Omega = \Omega_1 \times \cdots \times \Omega_p$ les valeurs d'une alternative $x \in X$ suivant chaque métrique. La valeur $u_i(x_i)$ représente le niveau de satisfaction du décideur pour x sur le critère i : 1 indiquant une satisfaction parfaite ; 0 un niveau pas du tout satisfaisant.

La valeur globale de x est donnée par :

$$U(x) = H(u_1(x_1), \ldots, u_p(x_p))$$

où $H : [0, 1]^p \to [0, 1]$ est la *fonction d'agrégation* des critères.

La relation de préférence $\succeq$ entre deux solutions x et $x' \in X$, est donc telle que $x \succeq x' \iff U(x) \geq U(x')$.

Sans perte de généralité on considèrera que les fonctions d'utilité monocritères sont croissantes dans cette partie.

2.2.2 Méthodes basées sur une somme pondérée

On peut noter que l'introduction de fonctions d'utilité monocritères règle un grand nombre de critiques adressées à la somme pondérée. De nombreuses méthodes sont donc basées sur ce modèle. Je présente ci-dessous deux des principales méthodes qui donnent une bonne illustration des approches employées.

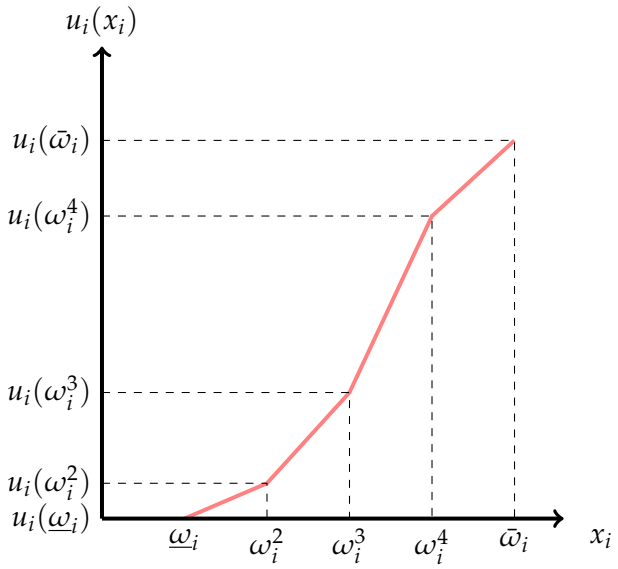


FIGURE 2.1 – *Fonction d'utilité linéaire par morceaux*

La méthode UTA

UTA ("UTilité Additive") (Jacquet-Lagreze and Siskos 1982) est une des premières méthodes d'AMCD. UTA suppose que la valeur d'une fonction est donné par la somme

$$U(x) = \sum_{i=1,\dots,p} u_i(x_i)$$

Les fonctions monocritères ne sont pas définies sur l'intervalle $[0,1]$: si l'on note $\Omega_i = [\underline{\omega}_i, \bar{\omega}_i]$ le domaine de définition d'une métrique i, celles-ci doivent vérifier les contraintes de normalisation $\sum_{i=1,\dots,p} u_i(\bar{\omega}_i) = 1$ et $u_i(\underline{\omega}_i) = 0,\ \forall i \in \{1,\dots,p\}$. Sur chaque critère $i \in \{1,\dots,p\}$, l'intervalle $[\underline{\omega}_i, \bar{\omega}_i]$ est découpé en $(\alpha_i - 1)$ segments qui décrivent une fonction linéaire par morceaux continue et monotone caractérisée par ses sommets $(\omega_i^1, u_i(\omega_i^1)), \dots, (\omega_i^{\alpha_i}, u_i(\omega_i^{\alpha_i}))$ avec $\underline{\omega}_i = \omega_i^1$ et $\bar{\omega}_i = \omega_i^{\alpha_i}$ (Figure 2.1).

Un ensemble d'alternatives de références X_R est défini. Il est ensuite ordonné par le décideur qui peut exprimer soit une préférence stricte $\succ$ entre deux alternatives consécutives, soit une équivalence $\approx$. On note $a^1, \dots, a^m$ les alternatives ordonnées. Soit la fonction Δ, définie de la manière suivante :

$$\Delta(a^k, a^{k+1}) = \sum_{i=1,\dots,p} u_i(a_i^k) + \sigma(a^k) - \Big(\sum_{i=1,\dots,p} u_i(a_i^{k+1}) + \sigma(a^{k+1})\Big)$$

où $\sigma(a)$ représente une erreur potentielle d'évaluation de a.

Le modèle d'AMCD est construit en résolvant le programme linéaire suivant :

$$\textit{minimiser } z = \sum_{a \in X_R} \sigma(a) \tag{2.1}$$

sous contraintes :

$$\Delta(a^k, a^{k+1}) \geq \delta \qquad \forall k \in \{1, \ldots, m-1\} \text{ tel que } a^k \succ a^{k+1} \tag{2.2}$$

$$\Delta(a^k, a^{k+1}) = 0 \qquad \forall k \in \{1, \ldots, m-1\} \text{ tel que } a^k \approx a^{k+1} \tag{2.3}$$

$$u_i(\omega_i^{j+1}) - u_i(\omega_i^j) \geq 0 \qquad \forall i \in \{1, \ldots, p\}, j \in \{1, \ldots, \alpha_i - 1\} \tag{2.4}$$

$$\sum_{i=1,\ldots,p} u_i(\bar{\omega}_i) = 1 \tag{2.5}$$

$$u_i(\underline{\omega}_i) = 0 \qquad \forall i \in \{1, \ldots, p\} \tag{2.6}$$

$$u_i(\omega_i^j) \in \mathbb{R}^+ \qquad \forall i \in \{1, \ldots, p\}, j \in \{1, \ldots, \alpha_i\} \tag{2.7}$$

$$\sigma(a) \in \mathbb{R}^+ \qquad \forall a \in X_R \tag{2.8}$$

où les variables sont les valeurs $u_i(\omega_i^j)$ associés aux extrémités des segments des fonctions d'utilité monocritères et $\sigma_i(a)$ les erreurs entre le modèle et les préférences de l'expert que le programme linéaire cherche à minimiser. Les contraintes (2.2) et (2.3) sont exprimées sur ces variables par interpolation linéaire. Les contraintes (2.4) à (2.6) représentent les contraintes de monotonie et de normalisation du modèle.

Le modèle et la méthode représentent l'avantage d'être simple. Cependant l'information préférentielle fournie par le décideur reste relativement pauvre et l'ensemble de référence doit être très conséquent pour que le modèle soit suffisamment bien défini par les préférences exprimées.

La méthode MACBETH

La méthode MACBETH (Bana e Costa and Vansnick 1994, Bana e Costa et al. 2005) repose sur une somme pondérée qui est équivalente au modèle UTA :

$$U(x) = \sum_{i=1,\ldots,p} w_i u_i(x_i)$$

où : $u_i(\underline{\omega}_i) = 0$ et $u_i(\bar{\omega}_i) = 1,\ \forall i \in \{1, \ldots, p\}$; et $\sum_{i=1,\ldots,p} w_i = 1$. Cette expression permettant de séparer la construction des fonctions d'utilité monocritères u_i (également linéaires par morceaux) et la construction des poids w_i.

Un des principaux apports de la méthode MACBETH est l'introduction d'une échelle portant sur l'intensité de la différence de préférence entre deux éléments. Cette échelle qualitative comporte six niveaux allant de "très faible" à "extrême". Elle est utilisé dans un premier temps pour la construction de chaque fonction d'utilité monocritère. Des préférences sont ainsi exprimées sur des valeurs caractéristiques de chaque métrique Ω_i fixées avec l'analyste. En second lieu, des alternatives de référence sont comparées par le décideur, l'échelle MACBETH permettant de fournir une information préférentielle plus riche que celle d'UTA. Pour chaque fonction, un programme linéaire est résolu pour déterminer des valeurs correspondant aux préférences énoncées. MACBETH intègre de plus un mécanisme qui attire l'attention du décideur sur les incohérences pouvant exister entre différentes comparaisons d'alternatives.

2.2.3 Modèles basé sur l'intégrale de Choquet

Les méthodes UTA et MACBETH reposent toutes les deux sur une somme pondérée qui suppose une *indépendance préférentielle* entre les critères : chaque critère contribue indépendamment de la valeur des autres à la qualité de la solution. Néanmoins cette hypothèse n'est pas toujours vérifiée. Un exemple en est donné dans Grabisch and Labreuche (2010) sur le choix d'un candidat scientifique sur la base de notes en mathématiques, physique, anglais : dans cet exemple, le décideur considère que si le candidat est bon en mathématique alors l'anglais compte plus que la physique. S'il est moyen en mathématiques alors c'est sa note en physique qui sera regardée en priorité. On montre facilement que ce type de jugement, pourtant simple, ne peut être modélisé par une fonction additive. Des notions d'*interactions* entre critères apparaissent.

Considérons un vecteur d'utilités $v = (v_1, \ldots, v_p) \in [0,1]^p$. La manière dont sont intégrées ces interactions dans l'intégrale de Choquet est facile à percevoir dans sa forme *2–additive*, définie de la manière suivante (Grabisch 1997) :

$$U(x) = \sum_{i=1,\ldots,p} \left(\phi_i - \frac{1}{2} \sum_{j \neq i} |I_{ij}| \right) v_i + \sum_{I_{ij}>0} I_{ij} \min(v_i, v_j) + \sum_{I_{ij}<0} |I_{ij}| \max(v_i, v_j) \tag{2.9}$$

où ϕ_i représente l'importance relative d'un critère i et I_{ij} représente l'interaction entre deux critères i et j. Les coefficients des termes v_i, $\min(v_i, v_j)$ et $\max(v_i, v_j)$ sont définis de manière à ce que leur somme soit égale à 1.

L'intégrale de Choquet 2–additive est composée d'une partie linéaire, suivie d'une partie conjonctive et d'une partie disjonctive. L'importance ϕ_i représentant à la fois l'importance propre d'un critère et celle en conjonction avec d'autres, la moitié des interactions dans lesquelles i intervient est soustraite à ϕ_i pour former son coefficient linéaire. Dans la partie conjonctive, un index d'interaction positif indique une *complémentarité* entre les critères (les deux doivent être bien satisfaits ensemble pour que la solution soit jugée bonne), ceci est traduit par l'opérateur min. Dans la partie disjonctive, un indice d'interaction négatif indique des critères *redondants* ou *substituables* (la satisfaction d'un des deux critères suffit à l'obtention d'une bonne solution). Ceci est traduit par l'opérateur max.

Pour aller vers encore plus d'expressivité de la fonction d'agrégation on peut définir un poids pour chaque sous-ensemble de l'ensemble des critères. Cet ensemble de poids constitue une *fonction de capacité*, fonction d'ensemble également appelée *mesure floue* et notée $\mu : \mathcal{P}(\{1, \ldots, p\}) \longrightarrow [0,1]$. Cette fontion vérifie les deux propriétés suivantes : $\mu(\emptyset) = 0$, $\mu(\{1, \ldots, p\}) = 1$ et $A \subset B \subset \{1, \ldots, p\} \Rightarrow \mu(A) \leq \mu(B)$. La forme générale de l'intégrale de Choquet permet alors d'agréger un ensemble d'utilités par rapport à une fonction de capacité afin d'établir l'utilité globale d'une alternative. Les utilités sont préalablement triées en ordre croissant et les termes de la capacité sont utilisés en fonction de cet ordre de la manière suivante :

Définition 2.1 *(Intégrale de Choquet (Choquet 1953))*
Soit $\mu : \mathcal{P}(\{1, \ldots, p\}) \longrightarrow [0,1]$ une fonction de capacité sur $\{1, \ldots, p\}$ et $v = (v_1, \ldots, v_p) \in [0,1]^p$. L'intégrale de Choquet $\mathcal{C}_\mu$ du vecteur v par rapport à μ est définie par :

$$\mathcal{C}_\mu(v_1, \ldots, v_p) = \sum_{i=1}^{p} (\mu(A_{\sigma(i)}) - \mu(A_{\sigma(i+1)})) v_{\sigma(i)} \tag{2.10}$$

avec $0 \leq v_{\sigma(1)} \leq \cdots \leq v_{\sigma(p)} \leq 1$, $A_{\sigma(i)} = \{\sigma(i), \ldots, \sigma(p)\}$ et $v_{\sigma(0)} = 0$.

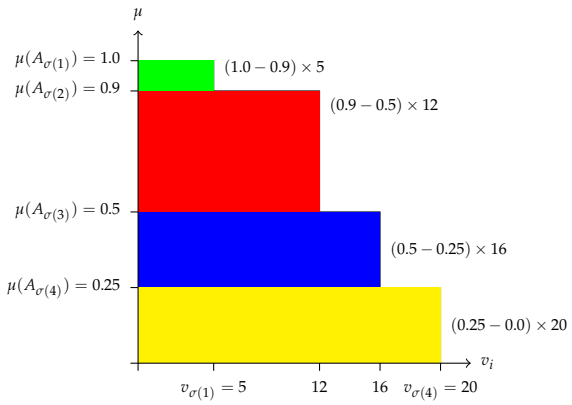


FIGURE 2.2 – *Représentation d'une intégrale de Choquet C_μ agrégeant 4 critères. La surface en couleur est égale à la valeur de l'intégrale de Choquet pour le vecteur $(5, 20, 12, 16)$ selon la Définition (2.1) : $C_\mu((5, 20, 12, 16)) = (1.0 - 0.9) \times 5 + (0.9 - 0.5) \times 12 + (0.5 - 0.25) \times 16 + (0.25 - 0.0) \times 20$. Chaque couleur représente une partie de cette somme.*

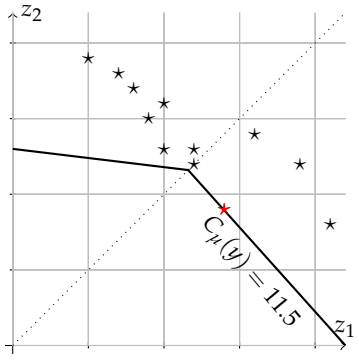


FIGURE 2.3 – *Minimisation d'une intégrale de Choquet C_μ : $\mu(\{1\}) = 0.5$, $\mu(\{2\}) = 0.9$, $\mu(\{1, 2\}) = 1$. Le trait en gras représente la courbe de niveau de l'intégrale de Choquet. Les étoiles ($\star$) représentent les points des solutions du problème.*

L'intégrale de Choquet peut donc être vue comme une somme pondérée dont les poids dépendent de l'ordre entre les arguments. C'est une fonction d'agrégation de type moyenne, idempotente, croissante, continue, invariante au changement d'échelle affine, et linéaire pour un ordre donné des arguments. On peut noter que l'intégrale de Choquet est une fonction d'agrégation très générale, qui contient comme cas particulier les opérateurs d'agrégation usuels tels que le minimum, le maximum, la médiane, la moyenne pondérée et la moyenne ordonnée pondérée (OWA) (Grabisch 1996; 1995).

Une écriture alternative de l'intégrale de Choquet peut être obtenue en utilisant la transformée de Möbius de la fonction de capacité (Grabisch 1997). La transformation de Möbius $m : \mathfrak{P}(P) \to \mathbb{R}$ d'une fonction de capacité μ est une fonction d'ensemble définie par : $m(A) = \sum_{B \subseteq A} (-1)^{|A \setminus B|} \mu(B), \quad \forall A \subseteq \{1, \ldots, p\}$. L'intégrale de Choquet d'une alternative v en fonction d'une transformation de Möbius m s'écrit :

$$C_m(v) = \sum_{A \subseteq \{1,\ldots,p\}} m(A) \times \min_{k \in A} v_k \tag{2.11}$$

L'intégrale de Choquet peut être utilisée à partir du moment où les valeurs agrégées sont sur la même échelle. La Figure 2.2 illustre la Définition 2.1 dans le contexte de la thèse d'Hugo Fouchal (Fouchal 2011) sur le calcul de plus courts chemins Choquet-optimaux (cette thèse est présentée plus en détail dans la suite de ce chapitre). La Figure 2.3 montre une courbe de niveau de l'intégrale utilisée en minimisation sur un problème à deux objectifs.

La construction d'un modèle basé sur l'intégrale de Choquet en AMCD suit un processus similaire aux méthodes MACBETH et UTA. Différentes méthodes ont été développées et intégrées dans des outils tels que la librairie Kappalab (Grabisch et al. 2008) de R, également intégrée dans le logiciel Diviz (Meyer and Bigaret 2012), ou le logiciel Myriad (Labreuche and Lehuédé 2005) à Thales. La méthode MACBETH a notamment été étendue au modèle basé sur l'intégrale

de Choquet 2–additive (Mayag 2010). L'intégrale de Choquet étant relativement complexe, le logiciel Myriad intègre un module d'explications permettant d'analyser les recommandations du modèle (Labreuche 2011).

2.3 Plus Court Chemin Choquet–Optimal

Les travaux réalisés dans le cadre de la thèse d'Hugo Fouchal ont pour motivation la conception d'algorithmes exacts pour des problèmes d'optimisation combinatoire où la fonction objectif est une intégrale de Choquet. Ces travaux ont porté entièrement sur la résolution de problèmes de plus court chemin multiobjectif avec une application en routage dans les réseaux internet. J'ai co-encadré cette thèse de 2008 à 2011 avec Xavier Gandibleux de l'Université de Nantes dans le cadre du projet MILES financé par la région Pays de Loire. Je résume ici les principales contributions présentées dans le mémoire de thèse d'Hugo Fouchal (Fouchal 2011).

La Section 2.3.1 présente le problème de plus court chemin multiobjectif et l'algorithme d'étiquetage qui sert de cadre général à l'ensemble des développements.

La première partie des travaux de recherche est présentée en Section 2.3.2. Elle concerne la recherche d'un chemin entre un sommet source s et un sommet t, optimal pour une intégrale de Choquet donné. Nous définissons des bornes inférieures permettant de réduire l'espace de recherche couvert par l'algorithme d'étiquetage et par là même de réduire le temps de calcul.

La seconde partie (Section 2.3.3) étudie la dominance entre labels dans le cas plus général de problèmes de plus court chemin entre un sommet source s et tous les autres sommets du graphe.

2.3.1 Plus court chemin multiobjectif et algorithme d'étiquetage

Les problèmes de plus court chemin multiobjectif (MOSP : MultiObjective Shortest Path problem) ont été étudiés à de nombreuses reprises depuis les années 80 (se référer aux états de l'art de Ehrgott and Gandibleux (2000), Tarapata (2007) et Climaco and Pascoal (2010)). Dans ce domaine, les algorithmes dis à fixation d'étiquettes (label setting) ou à correction d'étiquettes (label correcting) ont montré à parts égales leur supériorité dans la résolution de divers problèmes MOSP (Tarapata 2007).

Cette section présente en premier lieu le problème MOSP et les notations qui seront utilisées. En second lieu, je présenterai le principe général des algorithmes d'étiquetage qui a servi de support à l'ensemble des travaux sur le problème de plus court chemin Choquet-optimal.

Le problème de plus court chemin multiobjectif

On considère un graphe $G = (N, A, c)$ où $N = \{1, \ldots, n\}$ est un ensemble de n sommets et $A \subset N \times N$ un ensemble d'arcs orientés. Le vecteur c associe un coût positif à tout arc $(i, j) \in A$ selon chaque objectif : $c(i, j) = (c_1(i, j), \ldots, c_p(i, j)) \in \mathbb{R}_+^p$. L'ensemble N comprend un sommet source noté s et un sommet puit noté t. L'ensemble des successeurs d'un sommet $i \in N$ est noté $\Gamma^+(i)$ (i.e. $\Gamma^+(i) = \{j : (i, j) \in A\}$).

L'ensemble des chemins d'un sommet $i \in N$ vers un sommet $j \in N$ est noté $R_{(i,j)}$. On distingue deux ensembles particuliers :

- $R_{(s,t)}$: l'ensemble des chemins reliant une source s à un puit t.

- $R_{(s,\bullet)} = \cup_{j \in N} R_{(s,j)}$: l'ensemble des chemins entre un sommet source et les autres sommets du graphe.

On parlera par extension de problème $R_{(s,t)}$ s'il s'agit de trouver un ou des chemins d'un sommet s à un sommet t, ou de problème $R_{(s,\bullet)}$ s'il faut trouver un ensemble de chemins de s aux autres sommets du graphe.

On considère une fonction objectif vectorielle $z : R_{(\bullet,\bullet)} \rightarrow \mathbb{R}_+^p$ définie sur l'ensemble des sommets du graphe. L'ensemble $Y = z(R_{(s,\bullet)}) \subseteq \mathbb{R}_+^p$ est appelé *espace des objectifs*. On considèrera ici que les objectifs doivent être minimisés et qu'ils sont *additifs* : la longueur d'un chemin r selon la fonction objectif $k \in \{1, \ldots, p\}$ est la somme des coûts d'indice k sur les arcs empruntés par le chemin : $z_k(r) = \sum_{(i,j) \in r} c_k(i,j)$.

Pour deux solutions a et b représentées selon leurs vecteurs $y^a, y^b \in Y$ sur l'espace des objectifs, on dit que y^a domine y^b selon la relation de Pareto ($y^a \prec_P y^b$) si et seulement si : $\forall i \in P, y_i^a \leq y_i^b$ et $y^a \neq y^b$. Pour un sommet $t \in N$, $y^a \in z(R_{(s,t)})$ est un vecteur non dominé si et seulement si $\nexists y^b \in z(R_{(s,t)})$ tel que $y^b \prec_P y^a$. Un chemin $r \in R_{(s,t)}$ est dit *efficace* si et seulement s'il n'existe pas d'autre chemin qui le domine : $\nexists r' \in R_{(s,t)}$ tel que $z(r') \prec_P z(r)$. L'ensemble maximal complet des solutions efficaces d'un problème $R_{(s,\bullet)}$ est défini par $E(R_{(s,\bullet)}) \subseteq R_{(s,\bullet)}$. La représentation de cet ensemble sur l'espace des objectifs est également appelée *front de Pareto*.

L'énumération des solutions de $E(R_{(s,\bullet)})$ n'est en général pas possible en temps polynomial (Hansen 1980) : la version à deux objectifs ou plus du problème de plus court chemin est NP–difficile. Certains travaux montrent néanmoins que le nombre de chemins efficaces n'est en général pas exponentiel (Gandibleux et al. 2006).

Principe des algorithmes d'étiquetage

Les algorithmes d'étiquetage utilisent des étiquettes (*labels*) comme structure de données efficace pour stocker des chemins avec leurs sous-chemins. Une étiquette ou label $\ell_j = [y_1, \ldots, y_p, i, \ell_i]$, $j \in N$, correspond à un chemin $r = \langle s, \ldots, i, j \rangle \in R_{s,j}$ tel que $z(r) = (y_1, \ldots, y_p)$ est le vecteur des coûts du chemin r et $i \in N$ est le sommet auquel est associé le plus court sous-chemin $r' = \langle s, \ldots, i \rangle \in R_{s,i}$ de r. Ainsi dans un problème $R_{\bullet,\bullet}$, l'ensemble des labels forme l'arbre des plus courts chemins du sommet source à tous les autres sommets. L'ensemble de ces chemins est reconstruit par récursion depuis chaque sommet puits. On note L_i, $i \in N$ l'ensemble des labels ℓ_i associé au sommet i.

La *propagation* d'un label ℓ_j, $j \in N$, consiste à la création de nouveau labels associés aux sommets adjacents à $i : j \in \Gamma^+(i)$. Ces nouveaux labels correspondent aux extensions du chemin associé à ℓ_j par les arcs (i,j), $j \in \Gamma^+(i:)$.

Une *règle de dominance* $\prec$ est utilisée pour comparer les labels entre eux. Cette règle joue un rôle majeur dans l'efficacité des algorithmes. Les problèmes mono-objectifs par exemple se résolvent efficacement grâce à la propriété de Bellman qui implique qu'un seul label peut être conservé à chaque sommet du graphe pendant la résolution.

Le schéma général des algorithmes d'étiquetage est décrit par l'Algorithme 2.1.

Algorithme 2.1 : Algorithme d'étiquetage général

1. **Initialisation :** $L_i := \emptyset, \forall i \in N$, *le label initial* $\ell_s = [\mathbf{0}_1, \ldots, \mathbf{0}p, \star, \star]$ *est ajouté à l'ensemble des labels du sommet source* $L_s := \{\ell_s\}$ *et à l'ensemble des labels ouverts* $\mathfrak{L} := \{\ell_s\}$.

2. **Sélection :** *Le label courant* ℓ_i, *associé au sommet* $i \in N$, *est sélectionné parmi l'ensemble des labels ouverts* $\mathfrak{L}$ *suivant une* **règle de sélection**. ℓ_i *est retiré de cet ensemble :* $\mathfrak{L} := \mathfrak{L} \setminus \{\ell\_i\}$.

3. **Propagation :** *Le label* ℓ_i *est propagé aux sommets adjacents à* i *:* $\forall j \in \Gamma^+(i)$, $\ell_j := [z_1(\ell_i) + c_1(i,j), \ldots, z_p(\ell_i) + c_p(i,j), i, \ell_i]$. *Chaque label propagé,* ℓ_j, $j \in \Gamma^+(i)$ *est ajouté à l'ensemble des labels de* j *et à l'ensemble des labels ouverts :* $L_j := L_j \bigcup \{\ell_j\}$ *et* $\mathfrak{L} := \mathfrak{L} \bigcup \{\ell_j\}$.

4. **Règle de dominance :** *La dominance* $\prec$ *est vérifiée dans les ensembles de labels associés aux sommets adjacents à* i *: les labels dominés sont supprimés,* $\forall j \in \Gamma^+(i)$, $L_j := L_j \setminus \{\ell_j^a \in L_j : \exists \ell_j^b \in L_j, z(\ell_j^b) \prec z(\ell_j^a)\}$. *Les labels dominés sont aussi supprimés de l'ensemble des labels ouverts :* $\forall j \in N$, $\mathfrak{L} := \mathfrak{L} \setminus \{\ell_j^a \in \mathfrak{L} : \exists \ell_j^b \in L_j, z(\ell_j^b) \prec z(\ell_j^a)\}$.

5. **Test de terminaison :** *Si* $\mathfrak{L} = \emptyset$ *alors l'algorithme se termine et* $\bigcup_{i \in N} L_i$ *correspond à l'ensemble des chemins* $\prec$*-efficaces construits, sinon aller en 2.*

Si le graphe est acyclique alors l'algorithme d'étiquetage est équivalent à un algorithme de programmation dynamique : un label ℓ_i ($i \in N$) est sélectionné uniquement si il n'existe plus de label ouvert associé aux sommets précédents du sommet i, $\Gamma^-(i)$ (Henig 1986, Carraway et al. 1990, Gabrel and Vanderpooten 2002).

Les deux grandes classes d'algorithmes d'étiquetage diffèrent principalement en fonction de la règle de sélection des labels à chaque itération. Les algorithmes dis à *fixation d'étiquette* (e.g. Dijkstra) sélectionnent un label correspondant à un chemin efficace. Lorsque ceci est possible, un nombre minimal de propagations d'étiquettes est garanti. Les algorithmes à *correction d'étiquette* (e.g. Bellman) ne supposent pas une telle propriété.

Algorithme de fixation d'étiquette multiobjectif

L'algorithme de Martins (Martins 1984) est un algorithme à fixation d'étiquette permettant de trouver l'ensemble complet des chemins efficaces dans un graphe sans circuit de coût négatif. Il utilise une règle de dominance basée sur la relation de Pareto entre labels, en éliminant les chemins dominés à chaque itération. Cette dominance s'appuie sur une extension du principe d'optimalité de Bellman : *tout chemin efficace est composé de sous-chemins efficaces*. La règle de sélection est basée sur une comparaison des labels suivant un ordre lexicographique des objectifs.

Pour trouver un chemin préféré suivant un modèle multicritère, il est possible de construire l'ensemble des chemins efficaces et de tous les évaluer pour en retenir le meilleur. Un des objectifs des travaux sur la recherche d'un plus court chemin Choquet–optimal étant d'améliorer

la performance des algorithmes d'optimisation en exploitant les préférences du décideur, on comparera les temps de résolution des algorithmes proposés à ceux de l'algorithme de Martins.

2.3.2 Plus court chemin Choquet–optimal d'un sommet source à un sommet puits

On considère ici le problème de la recherche d'un chemin optimal dans $R(s,t)$ pour une intégrale de Choquet agrégeant directement un ensemble de p fonctions objectif. Ce problème est noté COP (Choquet Optimal Path problem). Les fonctions d'utilité monocritère ne sont pas incluses dans les travaux présentés : on considère que les objectifs sont commensurables et que la valeur de l'intégrale de Choquet doit être minimisée. Dans un souci de simplification je continuerai dans ce contexte à parler d'interactions positives pour des critères complémentaires et d'interactions négatives pour des critères substituables.

On considère un exemple simple de COP pour illustrer différents résultats :

Exemple 2.1 *Considérons le graphe $G = (N, A, c)$ de la Figure 2.4 et deux fonctions objectif additives. On s'intéresse au problème MOSP $R_{1,3}$. L'ensemble des solutions du problème est représenté sur l'espace*

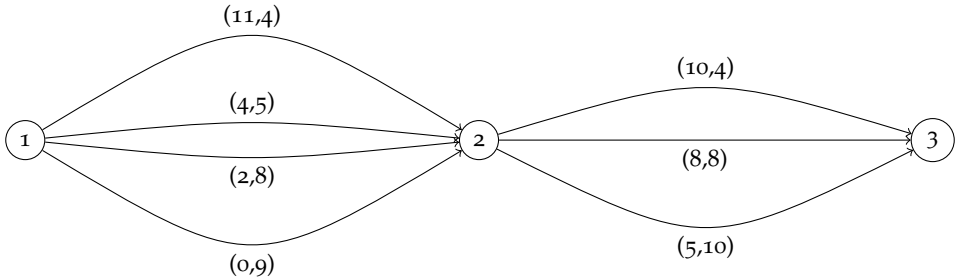


FIGURE 2.4 – *Définition d'un graphe $G = (N, A, c)$ avec $N = \{1, 2, 3\}$, les vecteurs de coûts sont définis sur les arcs.*

des objectifs sur la figure 2.5. On considère la minimisation d'une intégrale de Choquet C_μ avec μ la fonction de capacité définie telle que : $\mu(\emptyset) = 0$, $\mu(\{1\}) = 0.7$, $\mu(\{2\}) = 0.7$ $\mu(\{1,2\}) = 1$. La solution optimale de ce problème selon C_μ et la courbe de niveau correspondante sont représentés sur la Figure 2.6.

Le problème COP $R(s,t)$ comporte comme cas particulier la recherche d'un chemin pour lequel le maximum des objectifs est minimal et est donc NP-difficile (Murthy and Her 1992). Plus généralement, la minimisation d'une intégrale de Choquet ne respecte pas le principe d'optimalité de Bellman exploité dans l'algorithme de Martins : un chemin Choquet–optimal n'est pas nécessairement composé de chemins Choquet–optimaux.

Le problème COP $R(s,t)$ a été étudié par Galand and Perny (2007) dans le cas particulier où tous les critères sont complémentaires. Cette propriété est notamment respectée en optimisation robuste dans le cadre de solutions équilibrées. Leurs travaux sont basés sur la définition d'une somme pondérée H_λ des objectifs bornant inférieurement une intégrale de Choquet C_μ : un vecteur de poids $\lambda = (\lambda_1, \ldots, \lambda_p)$ est défini relativement à la fonction de capacité μ de telle sorte que $\forall y \in \mathbb{R}^p_+, H_\lambda(y) = \sum_{i=1,\ldots,p} \lambda_i y_i \leq C_\mu(y)$. Si l'on dispose du coût d'une solution $r^\star \in R(s,t)$ du problème (par exemple, la solution optimale pour la somme pondérée), la borne inférieure peut-être utilisée au sein de l'algorithme pour élaguer un label l_i correspondant à un chemin $r_{s,i}$ de s vers un sommet i : Soit $r_{i,t}$ le plus court chemin selon H_λ de i vers t. Alors si $H_\lambda(z(r_{s,i} \oplus r_{i,t})) > C_\mu(z(r^\star))$, tout chemin comportant le sous-chemin $r_{s,i}$ aura une valeur

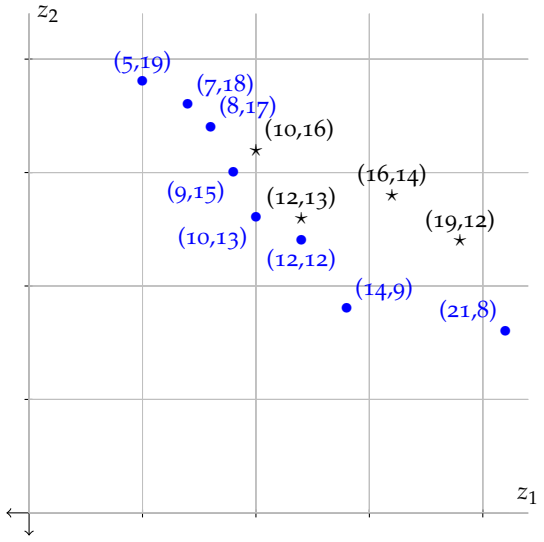


FIGURE 2.5 – *Ensemble des solutions du MOSP sur l'espace des objectifs. Les points représentent les solutions efficaces. Les étoiles sont des solutions dominées.*

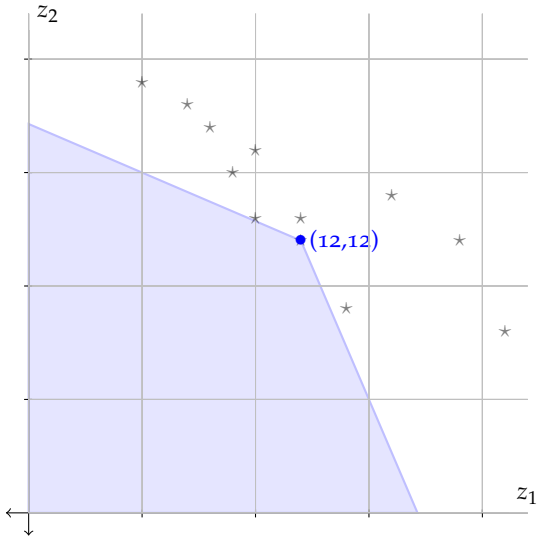


FIGURE 2.6 – *Représentation de la solution Choquet-Optimale. La zone bleu représente les points dans l'espace des objectifs qui sont meilleurs que le point* $(12, 12)$*, optimal ici pour l'intégrale de Choquet* C_μ.

supérieure à $r^\star$ selon C_μ. Le label l_i peut être élagué avant même l'application de la règle de dominance dans l'algorithme d'étiquetage.

On peut noter que le problème du chemin optimal pour une somme pondérée des objectifs peut être transformée en un problème mono-objectif en définissant le coût de chaque arc comme étant la somme pondérée de ses coûts. La résolution de ce problème est alors extrêmement efficace comparativement au problème multiobjectif.

Plusieurs extensions des travaux de Galand and Perny (2007) ont été réalisées. En premier lieu, cette borne a été généralisée à l'ensemble des fonctions de capacité possibles pour l'intégrale de Choquet pour aller vers une intégration en AMCD. Cette borne est calculée en résolvant un programme linéaire pour lequel différentes fonctions objectif ont été comparées. Finalement, les bornes des objectifs du MOSP ont été intégrées en modifiant ce programme linéaire de manière à améliorer la qualité de la borne inférieure, notamment en présence de critères substituables.

Définition d'une borne inférieure pour l'intégrale de Choquet

Une somme pondérée H_λ définit une borne inférieure de l'intégrale de Choquet dans le COP si pour tout chemin $r \in R_{s,t}$, $H_\lambda(z(r)) \leq C_\mu(z(r))$. L'ensemble des chemins n'étant pas connu à l'avance, on définit cette borne inférieure sur $\mathbb{R}^p_+$.

Propriété 2.1 *Soit μ une fonction de capacité et H_λ une somme pondérée. H_λ définit une borne inférieure de l'intégrale de Choquet C_μ si elle vérifie les relations :*

$$\forall y \in \mathbb{R}^p_+,\ H_\lambda(y) \leq C_\mu(y)$$

On montre alors l'équivalence suivante :

Théorème 2.1 *Soit μ une fonction de capacité et $\lambda \in \mathbb{R}_+^p$ un vecteur de poids, alors :*

$$\forall A \subseteq E,\ \sum_{i \in A} \lambda_i \leq \mu(A) \quad \Leftrightarrow \quad \forall y \in \mathbb{R}_+^p,\ H_\lambda(y) \leq C_\mu(y) \tag{2.12}$$

Le calcul du vecteur de poids d'une somme pondérée bornant inférieurement l'intégrale de Choquet peut donc être réalisé en résolvant le programme linéaire (*LP1*) :

$$\begin{array}{lll} \max & f(\lambda) & \\ \text{s.t.} & \sum_{i \in A} \lambda_i \leq \mu(A) & ,\forall A \subseteq P \quad (a) \\ & \lambda_i \in [0,1] & ,\forall i \in P \quad (b) \end{array} \tag{LP1}$$

Les variables du programme linéaire sont les coefficients de la somme pondérée. Les contraintes découlent directement du Théorème 2.1 et la fonction objectif est une fonction des coefficients à maximiser pour définir une borne inférieure la plus haute possible.

Plusieurs fonctions objectif sont proposées dans la thèse d'Hugo Fouchal. On détermine expérimentalement qu'une fonction $f(\lambda) = \sum_{k=1}^{p} \lambda_k + \epsilon \times \min_{k \in P}(\lambda_k - \phi_i)$, avec $\epsilon > 0$ une petite valeur positive, et $\phi(\mu)$ le vecteur des valeurs de Shapley qui définit pour chaque critère son importance propre en fonction des termes de la fonction de capacité (Shapley 1953). Cette fonction objectif permet d'obtenir une somme de poids maximum. Entre tous les jeux de poids de somme équivalente, l'agrégation min a pour fonction de déterminer un jeu de poids qui approxime les importances modélisées par la fonction de capacité.

L'exemple suivant montre comment une telle borne peut-être utilisée sur l'exemple de MOSP 2.1 pour supprimer un label au cours de l'algorithme :

Exemple 2.2 *Utilisation de la borne inférieure 2.12, suite de l'exemple 2.1.*

Soit une somme pondérée $H_{(0.3,0.7)}$ obtenue avec le PL (LP1) pour l'intégrale de Choquet C_μ présentée en Exemple 2.1. On considère deux labels ℓ_2, ℓ_2' associés au sommet 2 : $z(\ell_2) = (0,9)$ et $z(\ell_2') = (11,4)$. Le plus court chemin du sommet 2 au sommet 3 selon H correspond à l'arc de vecteur coût $(10,4)$. Soit $r_{1,3}^* \in R_{1,3}$ un chemin connu de 1 à 3, tel que $z(r_{1,3}^*) = (12,12)$. Les Figures 2.7 et 2.8 montrent la zone dominée par ce point selon la borne $H_{(0.3,0.7)}$, les deux labels, et les bornes inférieures trouvées pour chaque label.

La borne 2.12 permet de supprimer le label ℓ_2 (cf. Figure 2.7) :

$$C_\mu(12,12) = 12 \leq H_{(0.3,0.7)}(z(\ell_2) + (10,4)) = 10 \times 0.3 + 13 \times 0.7 = 12.1$$

Cette borne ne permet par contre pas de supprimer le label ℓ_2' (cf. Figure 2.8) :

$$C_\mu(12,12) = 12 > H_{(0.3,0.7)}(z(\ell_2') + (10,4)) = 21 \times 0.3 + 8 \times 0.7 = 11.9$$

.

Utilisation de bornes sur les objectifs pour la définition d'une meilleure borne

Le Théorème 2.1 découlant d'une définition des fonctions objectif sur $\mathbb{R}_+^p$, il est pertinent de chercher à réduire ce domaine en exploitant la structure du problème.

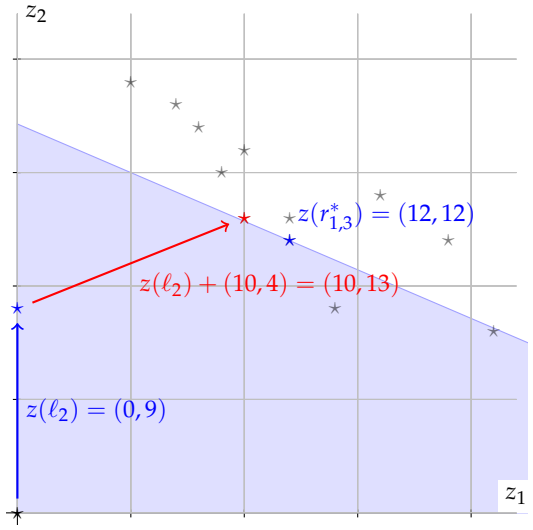


FIGURE 2.7 – *Représentation graphique de la borne calculée pour le label ℓ_2.*

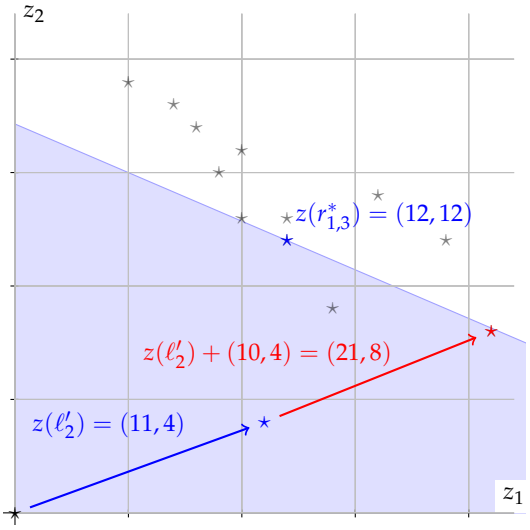


FIGURE 2.8 – *Représentation graphique de la borne calculée pour le label ℓ'_2.*

On peut en particulier noter que la somme pondérée H_λ borne (inférieurement) l'intégrale de Choquet C_μ si et seulement s'il existe une solution optimale $\mathbf{r}_{s,t} \in R_{s,t}$ selon C_μ ($C_\mu(z(\mathbf{r}_{s,t})) \leq C_\mu(z(r'_{s,t}))\ \forall r'_{s,t} \in R_{s,t}$) telle que :

$$H_\lambda(z(\mathbf{r}_{s,t})) \leq C_\mu(z(\mathbf{r}_{s,t})) \tag{2.13}$$

En particulier, sur l'espace des objectifs, le point optimal pour l'intégrale de Choquet appartient au front de Pareto. En résolvant un problème de plus court chemin mono-objectif par objectif on obtient un encadrement de ces derniers. On montre que si les valeurs des objectifs des solutions efficaces appartiennent à un domaine $[\underline{\xi}, \overline{\xi}]$, l'équivalence suivante peut être établie :

Théorème 2.2

$$\forall y \in [\underline{\xi}, \overline{\xi}]^p,\ H_\lambda(y) \leq C_\mu(y) \Leftrightarrow \forall y \in \{\underline{\xi}, \overline{\xi}\}^p,\ H_\lambda(y) \leq C_\mu(y)$$

Ce théorème permet d'aboutir au programme linéaire (*LP2*), qui définit une fonction H_λ bornant l'intégrale de Choquet par valeur inférieure.

$$\begin{array}{lll} \max & f(\lambda) & \\ \text{s.c.} & H_\lambda(y) \leq C_\mu(y) & \forall y \in \{\underline{\xi}, \overline{\xi}\}^p \\ & \lambda_k \in]0, 1] & \forall k \in P \end{array} \tag{LP2}$$

On note que si $\underline{\xi} = 0$ et $0 < \overline{\xi}$ alors les problèmes linéaires (*LP1*) et (*LP2*) sont équivalents.

Comparaison des deux approches et évaluations numériques

Le Tableau 2.1 présente les poids obtenus pour une fonction de capacité μ et trois calculs de sommes pondérées : Le vecteur λ^a est calculé sans borne avec le problème linéaire (*LP1*). Le vecteur λ^b est calculé avec le problème linéaire (*LP2*), avec des bornes $\overline{\xi} = 12$ et $\underline{\xi} = 69$. Le vecteur λ^c est calculé avec le problème linéaire (*LP2*) et $\underline{\xi} = 20$ et $\overline{\xi} = 25$. L'utilisation de bornes

sur les objectifs, $\underline{\xi}$ et $\overline{\xi}$ permet d'améliorer considérablement la somme des poids. Plus les bornes sont strictes, plus l'amélioration est importante.

ensemble	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	P
μ	0	0.3	0.2	0.2	0.3	0.5	0.4	1
λ^a	–	0.3	0.0	0.2	–	–	–	–
λ^b	–	0.3	0.079	0.279	–	–	–	–
λ^c	–	0.314	0.214	0.414	–	–	–	–

TABLE 2.1 – *Comparaison des sommes pondérées calculées suivant les différentes méthodes.*

Sur le plan expérimental, l'utilisation des bornes inférieures au sein de l'algorithme d'étiquetage est évalué pour 7 intégrales de Choquet : 4 intégrales de Choquet $C_{\mu^a}, C_{\mu^b}, C_{\mu^c}, C_{\mu^d}$ définies sur 3 objectifs (Tableau 2.2) et 3 intégrales de Choquet $C_{\mu^e}, C_{\mu^f}, C_{\mu^g}$ définies sur 5 objectifs (Tableau 2.3). Ces intégrales représentent diverses formes de fonction d'agrégation :

- *Critères substituables :* μ^e comporte des interactions négatives extrêmes, c'est en fait une fonction min sur cinq critères. μ^a modélise également des critères substituables mais avec des interactions moins prononcées.
- *Critères complémentaires :* la fonction μ^f comporte elle des interactions positives extrêmes, c'est en fait une fonction max sur cinq critères. μ^c modélise des interactions positives moins prononcées.
- *Cas mixte :* les capacités μ^b, μ^d et μ^g combinent des interactions positives et négatives. μ^b comporte néanmoins de fortes interactions négatives.

ensemble	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	P
μ^a	0.0	0.2	0.2	0.1	0.6	0.7	0.7	1.0
μ^b	0.0	0.3	0.1	0.2	0.9	0.3	0.8	1.0
μ^c	0.0	0.5	0.7	0.6	0.8	0.9	0.8	1.0
μ^d	0.0	0.4	0.3	0.3	0.4	0.4	0.3	1.0

TABLE 2.2 – *Fonctions de capacité pour l'intégrale de Choquet avec 3 objectifs.*

ensemble	$\lvert A\rvert = 0$	$\lvert A\rvert = 1$	$\lvert A\rvert = 2$	$\lvert A\rvert = 3$	$\lvert A\rvert = 4$	$\lvert A\rvert = 5$
μ^e	0.0	0.0	0.0	0.0	0.0	1.0
μ^f	0.0	1.0	1.0	1.0	1.0	1.0
μ^g	0.0	0.3	0.4	0.4	0.4	1.0

TABLE 2.3 – *Fonctions de capacité pour l'intégrale de Choquet avec 5 objectifs.*

Les algorithmes sont évalués sur un ensemble de graphes générés par le générateur *ggen*. Les graphes ont des coûts positifs sur les arcs définis entre 1 et 10. Le nombre de sommets des graphes est : $n = 1000$ ou 3000 ou 6000 ou 10000. Le nombre d'arcs des graphes est défini selon le nombre de sommets tel que : $m = 5 \times n$ ou $50 \times n$ ou $500 \times n$. Nous définissons la densité du graphe comme le nombre moyen d'arcs pour chaque sommet, i.e. 5 ou 50 ou 500. Pour chaque combinaison de taille et de densité, un ensemble de 50 graphes a été généré et est utilisé pour

l'expérimentation. Les combinaisons de 10000 sommets avec $m = 50 \times n$ ou $m = 500 \times n$ ne sont pas testés car de taille trop importante pour la mémoire du PC de test. Les sommets sources et destinations sont choisis aléatoirement.

Les tableaux 2.4 et 2.5 montrent les temps de calcul pour l'algorithme d'étiquetage décrit ci-dessus, pour l'optimisation des intégrales de Choquet $C_{\mu^a}, C_{\mu^b}, C_{\mu^c}, C_{\mu^d}$ et $C_{\mu^e}, C_{\mu^f}, C_{\mu^g}$ où les sommes pondérées bornant ces intégrales de Choquet sont construites selon (*LP1*) ou (*LP2*). Dans ce dernier cas, les bornes $\underline{\xi}$ et $\overline{\xi}$ sur les objectifs sont calculées à partir du point idéal et d'une approximation du point Nadir. Dans chaque tableau, les déviations maximales et moyennes correspondent aux pourcentages de temps de calcul maximums et moyens relativement à l'algorithme de Martins.

		Temps de calcul (s)										Déviation (%)	
		$m = 5 \times n$				$m = 50 \times n$			$m = 500 \times n$				
Fct. cap.	PL	1000	3000	6000	10000	1000	3000	6000	1000	3000	6000	max	moy.
C_{μ^a}	(*LP1*)	0.087	0.240	2.56	3.60	0.24	1.22	4.21	2.86	6.79	21.87	29.18	11.34
	(*LP2*)	0.027	0.085	0.34	0.55	0.09	0.47	1.33	1.19	4.10	14.03	12.14	4.32
C_{μ^b}	(*LP1*)	0.300	1.850	9.40	32.92	0.47	8.84	29.87	7.93	32.44	137.41	80.92	49.20
	(*LP2*)	0.140	0.390	3.50	7.95	0.37	5.56	17.81	5.97	21.55	95.40	60.92	26.83
C_{μ^c}	(*LP1*)	0.011	0.030	0.07	0.13	0.08	0.27	0.65	0.89	3.90	11.86	9.08	3.02
	(*LP2*)	0.014	0.035	0.08	0.14	0.09	0.28	0.71	0.93	4.05	12.09	9.49	3.23
C_{μ^d}	(*LP1*)	0.043	0.106	0.48	0.72	0.08	0.30	1.33	1.25	5.18	12.59	12.76	4.88
	(*LP2*)	0.016	0.053	0.12	0.23	0.05	0.25	0.82	1.05	4.40	9.75	10.71	3.25
Algorithme de Martins		0.466	4.657	19.00	63.99	2.35	26.54	99.26	9.80	58.63	204.18	100	100

TABLE 2.4 – *Expérimentations sur des problèmes de plus court chemin nœud à nœud ($R_{s,t}$) avec 3 objectifs additifs.*

préférences		temps de calcul (s)				déviation (%)	
		$m = 5 \times n$					
fct. cap.	problème linéaire	1000	3000	6000	10000	max	moy.
C_{μ^e}	(*LP1*)	1.04	10.28	45.10	148.13	83.87	79.57
	(*LP2*)	1.06	10.63	44.60	149.24	85.48	80.59
C_{μ^f}	(*LP1*)	0.02	0.10	0.19	0.47	1.61	0.74
	(*LP2*)	0.02	0.10	0.20	0.49	1.61	0.75
C_{μ^g}	(*LP1*)	0.04	0.51	0.76	1.56	4.03	2.34
	(*LP2*)	0.02	0.15	0.33	0.52	1.61	0.91
Algorithme de Martins		1.24	12.66	57.59	197.78	100	100

TABLE 2.5 – *Expérimentations sur des problèmes de plus court chemin nœud à nœud ($R_{s,t}$) avec 5 objectifs additifs.*

Les temps de calcul montrent l'utilisation de la borne inférieure 2.12 dans l'algorithme d'étiquetage peut permettre un gain d'efficacité important relativement au même algorithme basé sur la seule relation de Pareto (l'algorithme de Martins). Clairement, la réduction des temps de calcul est liée à la présence d'interactions positives dans le modèle. Ces gains semblent également augmenter avec le nombre de critères.

Quand les critères sont tous complémentaires (μ^c, μ^f), l'utilisation d'une borne inférieure est extrêmement efficace, avec un gain allant jusqu'à deux ordres de grandeur dans le cas min-max à 5 critères. Dans ces cas, l'utilisation de bornes sur les objectifs (*LP2*) n'est par contre pas utile.

Le temps de calcul peut même être légèrement dégradé en raison du surcout induit par le calcul des bornes $\underline{\xi}$ et $\overline{\xi}$ (cas μ^f).

Les gains sont moins important lorsque les critères sont substituables (μ^e, μ^a, μ^b). On note que dans ce cas, la somme des poids de la somme pondérée bornant l'intégrale de Choquet est inférieure à 1. L'utilisation de bornes sur les objectifs (*LP2*) peut alors permettre d'améliorer de manière significative les temps de calcul par rapport à (*LP1*) (μ^a, μ^b), mais elle reste inefficace dans le cas extrême min-min (μ^e).

2.3.3 Plus court chemin Choquet–optimal d'un sommet source à tous les autres

La plupart des algorithmes de plus court chemin tels que ceux de Dijkstra, Bellman ou Martins permettent de calculer le plus court chemin d'un sommet source à l'ensemble des autres sommets du graphe (problème $R_{s,\bullet}$). Si l'on désire résoudre ce problème, l'utilisation d'une borne inférieure, qui nécessite l'identification d'un sommet puits, n'est plus possible. On doit se baser sur la relation de dominance entre labels de l'algorithme d'étiquetage, pour éliminer des labels ne pouvant générer que des chemins sous-optimaux et réduire l'espace de recherche.

Problématique

Sur un MOSP avec objectifs additifs, une relation de dominance entre des labels sur la base de leurs coûts, suivant une fonction d'utilité multicritère U, se définit de la manière suivante :

Définition 2.2 *Dominance selon une fonction d'utilité pour le MOSP avec objectifs additifs. Soient $y^a, y^b \in Y$ les deux vecteurs de coûts respectifs de deux labels l_a et l_b. Si pour tout vecteur de coûts $y^c \in Y$ l'utilité de $y^a + y^c$ est plus petite ou égale à l'utilité de $y^b + y^c$ alors y^a U-domine y^b :*

$$y^a \precsim_U y^b \iff \forall y^c \in Y,\ U(y^a + y^c) < U(y^b + y^c)$$

Pour étudier la relation de dominance entre deux labels selon l'intégrale de Choquet (notée ($\precsim_{C_\mu}$)) on peut donc se focaliser sur l'étude de la fonction $\delta_{C_\mu}(y^a, y^b)$:

$$\delta_{C_\mu}(y^a, y^b) = \max_{y^c \in Y} C_\mu(y^a + y^c) - C_\mu(y^b + y^c) \tag{2.14}$$

Si $\delta_{C_\mu}(y^a, y^b)$ est négative ou nulle, on peut en déduire que $y^a \precsim_{C_\mu} y^b$ et le label l_b peut être supprimé.

La valeur de cette fonction ne peut être trouvée par la résolution d'un programme linéaire à chaque comparaison de label. Elle n'est par ailleurs pas croissante, comme illustré dans l'exemple suivant (Fouchal 2011) :

Exemple 2.3 *Soit un problème MOSP avec 3 objectifs additifs. On cherche à vérifier la dominance selon une intégrale de Choquet C_μ entre deux vecteurs de coûts $y^a, y^b \in Y = [0, 100]^3$ tels que $y^a = (100, 0, 100)$, $y^b = (0, 0, 50)$. L'intégrale de Choquet est définie par la fonction de capacité μ telle que $\mu(\{1\}) = 0$, $\mu(\{2\}) = \mu(\{3\}) = 0.5$, $\mu(\{1, 2\}) = \mu(\{1, 3\}) = 0.6$ et $\mu(\{2, 3\}) = 0.7$.*

Soit $f_{y^a,y^b} : Y \to \xi$ une fonction vectorielle telle que $f_{y^a,y^b}(y^c) = C_\mu(y^a + y^c) - C_\mu(y^b + y^c)$. La fonction f_{y^a,y^b} est la fonction à maximiser de l'équation (2.14) : $\delta_{C_\mu}(y^a, y^b) = \max_{y^c \in Y} f_{y^a,y^b}(y^c)$.

On étudie la fonction f_{y^a,y^b} selon le vecteur de coûts $y^c \in Y$ en considérant que le premier coût $y^c_1 \in [0, 100]$ de y^c varie entre 0 et 100 et les deux autres coûts $y^c_2 = y^c_3 = 0$ sont fixés à 0 :

- *Pour* $y_1^c \in [0, 50]$ *on obtient :*

$$\begin{aligned} f_{y^a,y^b}(y^c) &= [0.0 \times (100+0) + 0.4 \times (0+y_1^c) + 0.6 \times (100+0)] \\ &\quad -[0.3 \times (0+0) + 0.2 \times (0+y_1^c) + 0.5 \times (50+0)] \\ &= 0.2 \times y_1^c + 35 \end{aligned}$$

  Alors f_{y^a,y^b} *est strictement croissante sur* $[0, 50] \times \{0\} \times \{0\}$.

- *Pour* $y_1^c \in [50, 100]$ *on obtient :* $f_{y^a,y^b}(y^c) = -0.1 \times y_1^c + 50$, *ce qui implique que* f_{y^a,y^b} *est strictement décroissante sur* $[50, 100] \times \{0\} \times \{0\}$.

La fonction f_{y^a,y^b} *n'est donc ni croissante ni décroissante sur le premier objectif. Cet exemple peut aussi être appliqué aux autres objectifs.*

Une condition suffisante de dominance selon l'intégrale de Choquet

En utilisant l'écriture de l'intégrale de Choquet (2.11) donnée selon la transformée de Möbius m de μ, la fonction $\delta_{C_\mu}(y^a, y^b)$ s'écrit

$$\delta_{C_\mu}(y^a, y^b) = \max_{y^c \in Y} \sum_{A \subseteq \{1,\dots,p\}} m(A) \times \left[\min_{k \in A}(y_k^a + y_k^c) - \min_{k \in A}(y_k^b + y_k^c) \right] \tag{2.15}$$

Relativement à cette écriture, on montre que la relation suivante est vérifiée :

Lemme 2.1 *Pour tout vecteurs de coûts* $y^a, y^b \in Y$, *et tout ensemble d'objectifs* $A \subseteq \{1, \dots, p\}$ *l'inéquation suivante est vérifiée :*

$$\min_{k \in A}(y_k^a - y_k^b) \leq \min_{k \in A}(y_k^a) - \min_{k \in A}(y_k^b) \leq \max_{k \in A}(y_k^a - y_k^b) \tag{2.16}$$

Cette relation nous permet de déduire une borne supérieure $\overline{\delta}_{C_\mu}(y^a, y^b)$ de $\delta_{C_\mu}(y^a, y^b)$ qui établit une relation suffisante de dominance entre deux labels :

Théorème 2.3 *Condition suffisante de dominance selon l'intégrale de Choquet.*
Soient deux vecteurs de coûts $y^a, y^b \in Y$, *et une intégrale de Choquet* C_μ. y^a *domine* y^b *selon l'intégrale de Choquet* ($y^a \prec_{C_\mu} y^b$) *si :*

$$\overline{\delta}_{C_\mu}(y^a, y^b) = \sum_{A \subseteq \{1,\dots,p\} | m(A) > 0} m(A) \times (\max_{k \in A} y_k^a - y_k^b) + \sum_{A \subseteq \{1,\dots,p\} | m(A) < 0} m(A) \times (\min_{k \in A} y_k^a - y_k^b) < 0$$

Avec m la représentation de Möbius de la fonction de capacité μ.

Cette condition suffisante permet donc en théorie d'identifier et de supprimer un certain nombre de labels au cours du déroulement de l'algorithme d'étiquetage 2.1.

L'exemple suivant illustre l'utilisation de la condition suffisante de la C_μ-dominance du Théorème 2.3 dans un algorithme d'étiquetage pour le problème de l'exemple 2.1.

Exemple 2.4 *Dominance selon une intégrale de Choquet, suite de l'exemple 2.1.*
On considère trois labels au sommet 2 du graphe de la Figure 2.4 : ℓ_2^a, ℓ_2^b, ℓ_2^c *tels que* $z(\ell_2^a) = (4, 5)$, $z(\ell_2^b) = (0, 9)$, $z(\ell_2^c) = (11, 4)$. *La représentation de Möbius* $m : N \to [-1, 1]$ *de la fonction de capacité* μ *de l'exemple 2.1 est :* $m(\emptyset) = 0$, $m(\{1\}) = 0.7$, $m(\{2\}) = 0.7$ *et* $m(\{1, 2\}) = -0.4$.

On compare le label ℓ^a aux labels ℓ^b et ℓ^c selon la C_μ-dominance. Sur les figures 2.9 et 2.10, la zone grisée désigne l'espace des points non C_μ-dominés par le label ℓ^a : $\{y \in Y : 0 < \overline{\delta}_{C_\mu}(z(\ell^a), y)\}$. La condition suffisante de C_μ-dominance entre le label ℓ_2^a et le label ℓ_2^b (Figure 2.9) n'est pas vérifiée car $\overline{\delta}_{C_\mu}(z(\ell^a), z(\ell^b)) \geq 0$:

$$\overline{\delta}_{C_\mu}(z(\ell^a), z(\ell^b)) = 0.7 \times (4-0) + 0.7 \times (5-9) - 0.4 \times (5-9) = 2.8 - 2.8 + 1.6 = 1.6$$

Par contre, grâce cette condition suffisante on peut montrer que le label ℓ_2^a C_μ-domine le label ℓ_2^c (Figure 2.10), en effet $\overline{\delta}_{C_\mu}(z(\ell^a), z(\ell^c)) < 0$:

$$\overline{\delta}_{C_\mu}(z(\ell^a), z(\ell^c)) = 0.7 \times (4-11) + 0.7 \times (5-4) - 0.4 \times (4-11) = -4.9 + 0.7 - 2.8 = -1.4$$

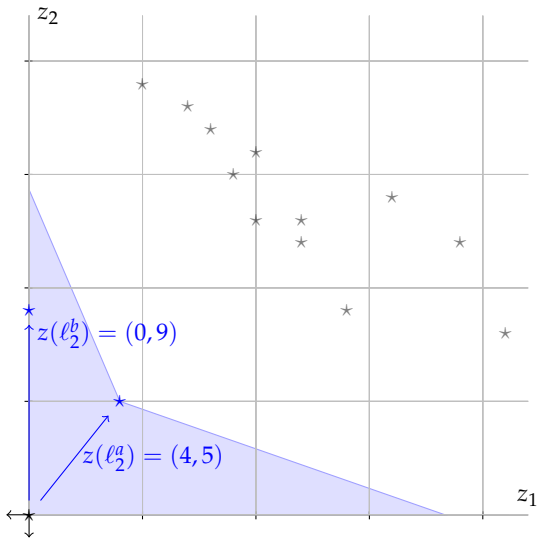


FIGURE 2.9 – *Le vecteur $(0,9)$ correspond au label ℓ_2^b, le vecteur $(4,5)$ correspond au label ℓ_2^a.*

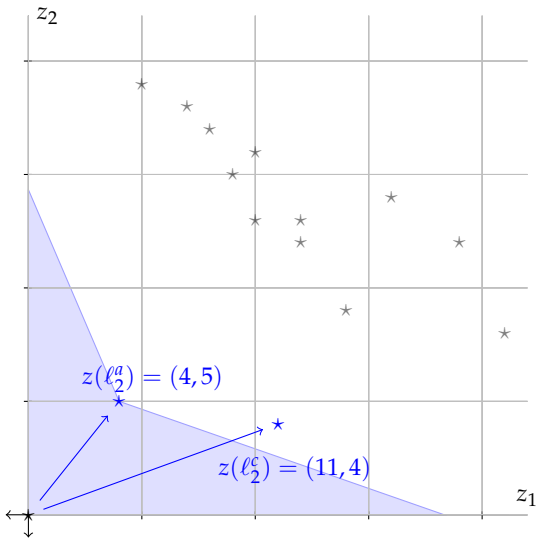


FIGURE 2.10 – *Le vecteur $(11,4)$ correspond au label ℓ_2^c, le vecteur $(4,5)$ correspond au label ℓ_2^a.*

Expérimentations

On cherche ici à évaluer l'efficacité de la condition suffisante de dominance selon une intégrale de Choquet C_μ dans l'algorithme d'étiquetage. Cet algorithme d'étiquetage permet de résoudre exactement l'optimisation d'une intégrale de Choquet dans un problème de plus court chemin multiobjectif d'un sommet à tous les sommets. De manière à évaluer l'algorithme pour différents profils de décideurs, plusieurs intégrales de Choquet sont définies par les fonctions de capacité μ^a, μ^b, μ^c, μ^d, μ^e, μ^f, μ^g de la table 2.6. Les fonctions de capacité μ^a et μ^c modélisent des fonctions d'agrégation (quasi)-linéaires (peu ou aucune interactions). Les fonctions de capacité μ^e et μ^f modélisent des interactions fortes entre les objectifs. Les fonctions de capacité μ^b, μ^d, μ^g et μ^h sont des fonctions intermédiaires.

Les temps de calcul observés dans le Tableau 2.7 montrent que plus l'intégrale de Choquet est proche d'une somme pondérée, plus la C_μ-dominance est efficace.

Fonctions de capacité	Ensembles d'objectifs							
	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	P
μ^a	0.0	0.333	0.333	0.333	0.666	0.666	0.666	1.0
μ^b	0.0	0.3	0.3	0.3	0.7	0.7	0.7	1.0
μ^c	0.0	0.6	0.2	0.2	0.8	0.8	0.4	1.0
μ^d	0.0	0.3	0.3	0.3	0.8	0.4	0.6	1.0
μ^e	0.0	0.4	0.3	0.3	0.4	0.4	0.3	1.0
μ^f	0.0	0.4	0.1	0.4	1.0	0.8	1.0	1.0
μ^g	0.0	0.6	0.1	0.3	0.9	1.0	0.4	1.0
μ^h	0.0	0.3	0.3	0.3	0.8	0.4	0.9	1.0

TABLE 2.6 – *Fonctions de capacité pour l'intégrale de Choquet avec 3 objectifs.*

Ainsi avec une intégrale de Choquet quasi-linéaire C_{μ^a}, C_{μ^c}, l'algorithme d'étiquetage utilisant la condition suffisante obtient des temps de calcul 90% plus rapide en moyenne que le même algorithme avec la dominance de Pareto. Lorsque les interactions sont légèrement plus importantes (C_{μ^b}, C_{μ^d}), la C_μ-dominance garde des gains significatifs ($90\% - 80\%$ en moyenne). Pour une intégrale de Choquet avec des interactions plus importantes (C_{μ^g} et C_{μ^h}), la C_μ-dominance permet de diminuer légèrement les temps de calcul : $60\% - 70\%$ de réduction par rapport à l'utilisation seule de la dominance de Pareto. Par contre, dans le cas d'interactions extrêmes, négatives ou positives, ce qui est le cas pour C_{μ^e} et C_{μ^f} alors les temps de calcul obtenus en utilisant la C_μ-dominance souffrent d'une perte de performance par rapport à l'utilisation uniquement de la dominance de Pareto. On peut noter que dans ces cas où l'intégrale de Choquet se rapproche des fonctions min ou max, il est difficile d'espérer un filtrage plus important que celui de la dominance de Pareto. L'utilisation de la C_μ-dominance représente donc un surcout.

Par ailleurs, plus la taille et la densité des graphes, augmente plus la C_μ-dominance permet un gain important.

Extensions

La notion de dominance présentée ci-dessus pour la minimisation d'une intégrale de Choquet dans le MOSP présente de nombreux intérêts applicatifs : Tout d'abord elle a été étendue dans la thèse d'Hugo Fouchal pour prendre en compte plusieurs types de fonction objectif (min-max par exemple) et pas uniquement la fonction additive pour le calcul du coût d'un chemin à partir des arcs qui le compose. Cette généralisation a permis de plus d'intégrer les fonctions d'utilité partielles du modèle multicritère basé sur l'intégrale de Choquet. Il est donc possible de considérer des problèmes de plus court chemin multicritère plus réalistes où les critères sont de natures différentes (distance, temps, types de voies, niveau de sécurité).

Ces avancées on permis la modélisation et la résolution d'un problème de routage sur réseau IP permettant de trouver un bon compromis entre le délai d'acheminement d'un paquet, le nombre de sauts sur le chemin et l'utilisation de la bande passante sur le réseau. Les deux premiers objectifs sont additifs et le dernier est de type min-max. Un modèle MAUT basé sur l'intégrale de Choquet a été présenté ainsi qu'une démonstration de l'efficacité des algorithmes intégrant la relation de dominance proposée. Dans un cadre temps-réel qu'on peut imaginer

Intégrales de Choquet	Temps de calcul (s)						Déviation (%)	
	$m = 5 \times n$			$m = 50 \times n$		$m = 500 \times n$		
	1000	6000	10000	1000	6000	1000	max.	moy.
C_{μ^a}	0.004	0.038	0.073	0.062	0.509	0.796	12.820	7.720
C_{μ^b}	0.010	0.100	0.197	0.196	2.037	1.780	32.051	20.395
C_{μ^c}	0.006	0.052	0.100	0.085	0.734	0.955	19.230	10.592
C_{μ^d}	0.006	0.057	0.111	0.091	0.829	0.970	19.230	11.145
C_{μ^e}	0.035	0.461	1.086	1.411	24.798	8.553	135.896	113.002
C_{μ^f}	0.036	0.468	1.106	1.529	23.091	8.847	137.261	116.911
C_{μ^g}	0.015	0.152	0.301	0.399	4.336	3.493	48.076	34.915
C_{μ^h}	0.017	0.175	0.347	0.496	5.652	4.263	57.233	41.575
Martins	0.031	0.462	0.957	1.318	21.729	7.448	100	100

TABLE 2.7 – *Expérimentations sur un problème de plus courts chemins d'un sommet à tous les sommets ($R_{s,\bullet}$) avec 3 objectifs additifs. Comparaison des temps de calcul entre un algorithme d'étiquetage 2.1 utilisant la C_μ-dominance et l'algorithme de Martins qui utilise la dominance de Pareto. Les colonnes déviation (exprimées en pourcentage) donnent le maximum et la moyenne des temps de calcul par rapport au temps de calcul de l'algorithme de Martins.*

de plus en plus flexible dans le futur, l'intégration d'algorithme rapide est en effet un élément crucial.

2.3.4 Conclusion et perspectives

Les travaux réalisés pour la résolution de problèmes de plus courts chemins présentent de nombreuses perspectives intéressantes pour la résolution exacte de problèmes d'optimisation multicritère. En effet, la définition d'une borne de l'intégrale de Choquet peut s'étendre à de nombreux algorithmes de *branch-and-bound*. La notion de dominance elle est très présente dans les algorithmes de programmation dynamique.

Depuis la thèse d'Hugo Fouchal, on peut signaler que de nombreux travaux ont été réalisés autour de l'optimisation de l'intégrale de Choquet, par Galand et al. (2013) notamment, par exemple sur la poursuite des travaux autour de la relation de dominance. L'optimisation de l'intégrale de Choquet en optimisation continue a également été étudié par Timonin (2012) et par (Lesca et al. 2013) dans le cas linéaire.

2.4 Optimisation Multicritère en Transport à la Demande

Si l'on regarde les problématiques de tournées de véhicules, auxquelles je me suis particulièrement intéressé depuis mon arrivée à Nantes, la dimension multicritère apparaît clairement dans le cadre de l'optimisation du transport de passagers (Paquette et al. 2009). Lorsqu'il s'agit de services sanitaires, la prise en compte de critères de qualité de service fait partie de la réglementation. C'est le cas dans le service récurrent de transport pour des personnes handicapées étudié dans le logiciel Marika (Lehuédé et al. 2009). La notion de qualité de service est également très présente dans le transport de patients assurées par les compagnies d'ambulances.

Cette problématique a fait l'objet de la thèse de Sophie Parragh (Parragh 2009), qui traite le cas de la compagnie *Australian Red Cross*.

L'utilisation d'un modèle MAUT basé sur l'intégrale de Choquet pour modéliser des préférences multicritères dans un problème de transport de personnes a été étudié avec Olivier Péton et Renaud Masson en collaboration avec Sophie Parragh et Fabien Tricoire de l'université de Vienne en Autriche dans le cadre d'un partenariat Hubert Curien Amadeus. Ces travaux ont permis d'établir plusieurs contributions intéressantes sur les aspects modélisation et résolution avec des préférences réalistes construites à partir de l'expérience acquise par les deux équipes sur deux projets proches. Ces travaux ont été publiés dans la revue *Journal of the Operational Research Society* en 2013 (Lehuédé et al. 2013).

Ce problème ainsi que les aspects les plus significatifs des travaux réalisés sont résumés dans les trois sections suivantes.

2.4.1 Problématique

La structure du problème de transport à la demande considéré est relativement classique : on considère un ensemble R de requêtes de transport. P désigne l'ensemble des points de départ et D l'ensemble des points d'arrivée de ces requêtes. Une requête $r \in R$ est définie par son point de départ $p(r) \in P$, son point d'arrivée $d(r) \in D$, un nombre de passagers q_r devant être transportés ensemble de leur point de départ à leur point d'arrivée, et une durée de trajet maximum $\bar{L}_r$. A chaque point $i \in P \cup D$ est associé une durée de service s_i reflétant le temps nécessaire à la montée ou la descente des passagers, et une fenêtre de temps $[e_i, l_i]$. On considère qu'un départ ou une arrivée au point i après l_i n'est pas autorisée. Un véhicule arrivant au point i en avance doit attendre l'heure e_i pour que l'opération prévue à ce point puisse avoir lieu.

Pour satisfaire ces requêtes on considère un ensemble K de véhicules identiques de capacité C personnes. Chaque véhicule $k \in K$ réalise une tournée entre un point $o(k)$ et un point $o'(k)$ appartenant respectivement aux ensembles O et O' des points de départ et d'arrivée du problème. La durée d'une tournée doit être inférieure à une valeur fixée notée T.

On note $V = P \cup D \cup O \cup O'$ l'ensemble des sommets du graphe sur lequel on modélise ce problème. t_{ij} représente la durée du plus court chemin entre un sommet $i \in V$ et un sommet $j \in V$.

Le problème de transport à la demande multicritère considéré, appelé MCDARP, consiste à construire un ensemble de tournées préféré en utilisant les véhicules de la flotte K de manière à satisfaire l'ensemble des requêtes de transport.

Pour modéliser ce problème, et en particulier les fonctions objectif présentées ci-dessous, on définit les variables booléennes x_{ij}^k, égales à 1 si l'arc (ij) est parcouru par le véhiule k, 0 sinon.

Quatre familles de variables réelles sont également définies :

- A_i^k l'heure d'arrivée du véhicule k au sommet $i \in V$,
- B_i^k l'heure de début de service du véhicule k au sommet $i \in P \cup D$,
- L_r la durée du trajet de la requête $r \in R$,
- Q_i^k le nombre de passagers dans le véhicule $k \in K$ lorsqu'il quitte le sommet $i \in P \cup D$.

Sur la base de ces notations on définit le problème MCDARP (MultiCriteria Dial A Ride Problem) qui consiste à définir un ensemble de tournées, permettant de servir l'ensemble des requêtes de transport, tout en maximisant le niveau de satisfaction établi par un modèle multicritère basé sur une intégrale de Choquet.

2.4.2 Modèle multicritère

La démarche suivie pour la modélisation des préférences va de l'identification d'objectifs pertinents à la construction du modèle multicritère. Dans le but de résoudre le problème MCDARP pour plusieurs jeux de données, on peut noter que certaines fonctions objectif traditionnellement utilisées en tournées de véhicules ne constituent pas de bonnes métriques. Il est nécessaire de définir des métriques dont les valeurs sont relativement indépendantes de la taille des instances considérées.

Fonctions objectif

La modélisation des préférences entre solutions de l'ensemble X de solutions du problème MCDARP est basée sur cinq fonctions objectif. Elles sont considérées comme suffisamment significatives et différentes en terme d'impact sur les solutions.

Trois objectifs relatif à la *qualité du service* ont été retenus :

$$f_{mrt}(x) = \max_{r \in R}(L_r - t_{p(r),d(r)}) \qquad \textit{(le maximum des temps de trajet supplémentaires)},$$

$$f_{art}(x) = \frac{1}{\sum_{r \in R} q_r} \sum_{r \in R} q_r (L_r - t_{p(r),d(r)}) \qquad \textit{(le temps de trajet supplémentaire moyen)},$$

$$f_{wtd}(x) = \sum_{r \in R} q_r (l_{d(r)} - B^k_{d(r)}) \qquad \textit{(le temps total d'attente à destination)}.$$

La notion de temps de trajet supplémentaire, utilisée notamment par Spada et al. (2005), modélise la différence entre la durée du trajet réalisé et la durée d'un trajet direct entre le point de départ et le point d'arrivée d'une requête.

Les objectifs *économiques* identifiés sont les deux fonctions classiquement utilisées en tournées de véhicules :

$$f_{tt}(x) = \sum_{k \in K} \sum_{i,j \in V} x^k_{ij} t_{ij} \qquad \textit{(le temps de conduite total)},$$

$$f_v(x) = \sum_{k \in K} \sum_{i \in P} x^k_{o(k),i} \qquad \textit{(le nombre de véhicules utilisés)}.$$

Définition de métriques pour le modèle AMCD

Dans un modèle MAUT basé sur l'intégrale de Choquet, une fonction d'utilité monocritère permet d'associer un niveau de satisfaction à la valeur d'une métrique. Lorsqu'il s'agit de modéliser des préférences pour un problème d'optimisation, les fonctions objectif considérées comme pertinentes pour le problème sont potentiellement les métriques du modèle multicritère. Néanmoins, si l'on désire construire un modèle de préférence valable quelle que soit l'instance considérée, il est important de définir des métriques relativement indépendantes de la taille de l'instance. Dans certains travaux on parle de *normalisation* des objectifs.

Dans les fonctions objectif présentées ci-dessous, les fonctions relatives à la qualité de service, orientées passager, sont relativement adaptées : on prendra le temps moyen d'attente à destination pour le troisième objectif au lieu du temps total.

Les fonctions de coût elles, doivent être révisées. En considérant que l'ensemble des instances correspondent à différentes journées de demande sur un même territoire, on peut considérer que le temps moyen de conduite par passager est un indicateur pertinent de la consommation des véhicules. Pour modéliser le coût de la flotte on propose de baser le modèle sur le taux d'utilisation moyen des véhicules utilisés.

Les cinq métriques définies pour le modèle sont donc les suivantes :

$$(\Omega_1) \qquad m_{mrt}(x) = f_{mrt}(x) \qquad \textit{(le maximum des temps de trajet supplémentaire)},$$

$$(\Omega_2) \qquad m_{art}(x) = f_{art}(x) \qquad \textit{(le temps de trajet supplémentaire moyen)},$$

$$(\Omega_3) \qquad m_{wtd}(x) = \frac{f_{wtd}(x)}{\sum_{r \in R} q_r} \qquad \textit{(le temps moyen d'attente à destination)},$$

$$(\Omega_4) \qquad m_{tt}(x) = \frac{f_{tt}(x)}{\sum_{r \in R} q_r} \qquad \textit{(le temps moyen de conduite par passager)},$$

$$(\Omega_5) \qquad m_v(x) = \frac{\sum_{r \in R} q_r}{f_v(x) \times C} \qquad \textit{(le taux d'utilisation moyen des véhicules utilisés)}.$$

On peut noter que pour un ensemble donné de requêtes à transporter, maximiser le taux d'utilisation des véhicules est strictement équivalent à minimiser le nombre de véhicules. Pour un problème de tournées avec collecte et livraison comme le problème MCDARP, ce taux peut être supérieur à un.

Le modèle multicritère

La Figure 2.11 présente les fonctions d'utilité monocritères définies sur chacune des métriques.

Les cinq critères sont agrégés par une intégrale de Choquet 2-additive (2.9) définie en Section 2.2.3 page 15. Quatre fonctions de capacité ont été définies pour les expérimentations. Les deux premières correspondent à des profils utilisateur : μ_{cost} modélise un utilisateur accordant une importance particulière aux centres de coûts, μ_{QoS} modélise un utilisateur orienté qualité de service. Les deux capacités suivantes sont définies pour les expérimentations : μ_{avg} modélise une moyenne arithmétique, μ_{bal} modélise des critères complémentaires d'importances égales.

2.4.3 Recherche à voisinage large

La résolution du problème MCDARP est réalisée par une métaheuristique de type Large Neighborhood Search (LNS). Elle a été développée pour résoudre des instances issues de problèmes de transport de personnes handicapées vers leurs établissements d'éducation ou de travail sur des données de l'ADAPEI 44 et du conseil général de Loire Atlantique. L'algorithme LNS proposé est issu des travaux de Syefan Ropke et David Pisinger sur le PDPTW Ropke and Pisinger (2006a). L'algorithme initial, présenté en détail au Chapitre 3, Section 3.3, comporte une partie adaptative. Les expérimentations n'ayant pas montré l'apport de cette couche, celle-ci a été omise. L'essentiel des évolutions proposées portent sur les opérateurs de destruction et reconstruction des solutions. Les opérateurs ont des probabilités de sélection identiques à chaque itération. Trois configurations ont été proposées : la configuration *basic* comporte des opérateurs issus des heuristiques LNS de Shaw (1998), Ropke and Pisinger (2006a;b), Pisinger and Ropke (2007), Prescott-Gagnon et al. (2009a) ; la configuration *advanced* ajoute quelques opérateurs qui

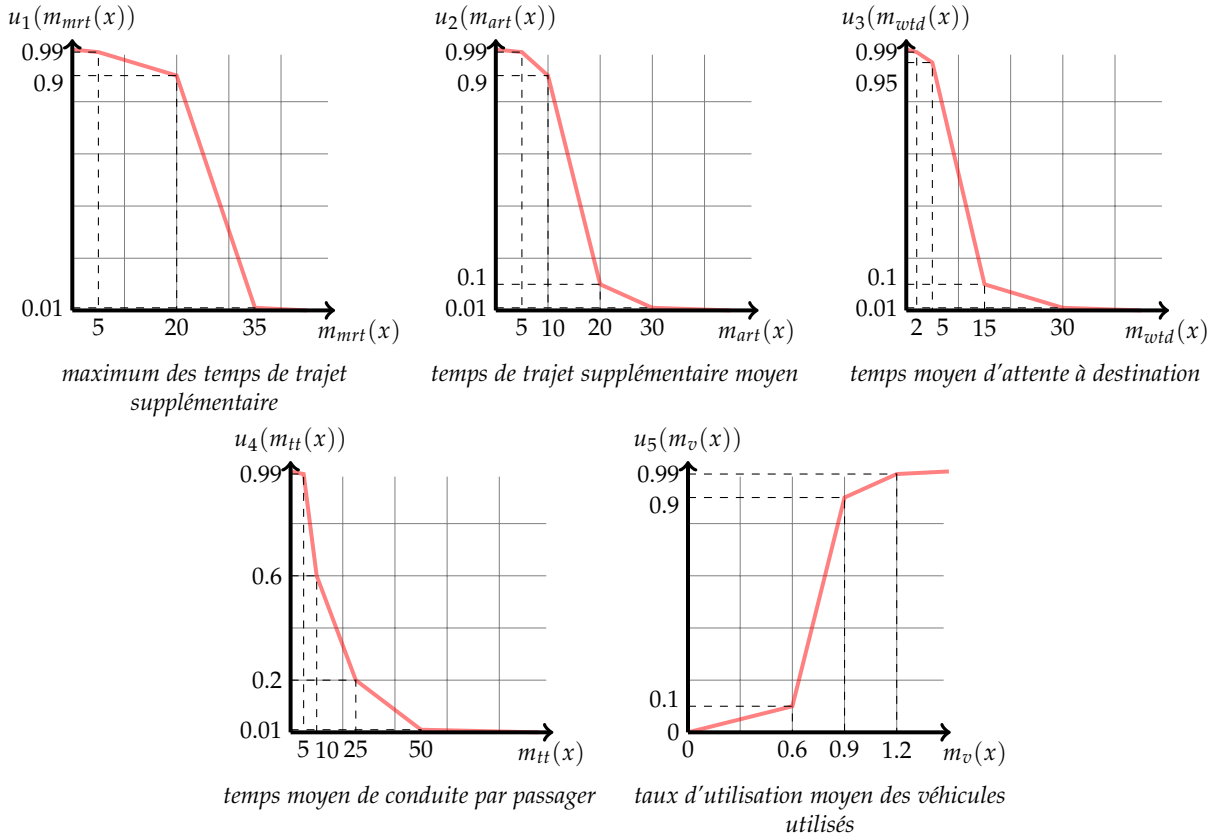


FIGURE 2.11 – *Fonctions d'utilité monocritères du problème* MCDARP

Opérateur	**Principe général**
	Configuration *basic*
Random removal	Retraits aléatoires.
Worst removal	Retire les requêtes qui génèrent les plus grandes améliorations de l'objectif.
Distance-related removal	Retire un ensemble de requêtes proches géographiquement.
Time-related removal	Retire un ensemble de requêtes servies à des heures proches.
Proximity-based removal	Retire des requêtes proches géographiquement et dans le temps.
Best insertion	Insère la requête qui augmente le moins l'objectif à sa meilleure position.
k-regret insertion	Insère les requêtes à leurs meilleures positions par ordre décroissant de regret.
	Configuration Advanced (opérateurs supplémentaires)
Delivery removal	Retire les requêtes se rendant à une même destination dans une tournée.
Clustered pickup removal	Retire des requêtes ayant la même destination et dont les origines se suivent dans une tournée.
Route removal	Retire des tournées entières.
	Configuration parametrized (opérateurs supplémentaires)
Parametrized worst removal	Worst removal guidé par une des métriques du problème.
Parametrized best insertion	Best instertion guidé par une des métriques du problème.
Parametrized k-regret	k-regret guidé par une des métriques du problème.

TABLE 2.8 – *Opérateurs des différentes configurations du LNS MCDARP*

exploitent le fait que de nombreuses requêtes ont la même destination ; la configuration *parametrized* intègre, en plus des opérateurs d'*advanced*, une paramétrisation des opérateurs de reconstruction où les évaluations de coût d'insertion utilisent une des cinq métriques du problème à la place de la fonction objectif multicritère. Dans cette dernière configuration on crée donc un opérateur de chaque type par métrique (opérateur monocritère).

Les opérateurs des différentes configurations sont listés dans le Tableau 2.8.

Des expérimentations ont été réalisées sur 14 instances provenant de l'ADAPEI 44 et du conseil général de Loire Atlantique. Le nombre d'itérations de l'algorithme LNS a été calibré de manière à produire des temps d'exécutions similaires pour chaque configuration : 10 000 itérations pour *basic*, 12 000 pour *advanced* et 20 000 pour *parametrized*. 10 exécutions de l'algorithme ont été réalisées pour chaque instance et chaque configuration.

L'écart de temps de calcul est principalement dû à la complexité du calcul de l'intégrale de Choquet dans les heuristiques de reconstruction : l'évaluation d'une insertion suivant une seule métrique a un temps de calcul inférieur.

La synthèse de ces résultats est présentée dans le Tableau 2.9.

Les conclusions de l'étude des résultats sont les suivantes : En premier lieu, il est difficile de départager très clairement les heuristiques. Une première raison est que les objectifs ont été normalisés, évalués séparément par des fonctions d'utilité monocritères et agrégés pour établir une valeur variant entre 0 et 1. Les différences de performance sont également quelque peu gommées par l'agrégation des résultats de 10 exécutions sur chaque instance. Une seconde raison provient de l'aspect multicritère du problème qui peut rendre plus complexe l'exploration de l'espace de recherche. On peut l'expliquer intuitivement de la manière suivante : si la taille de l'ensemble des solutions du problème est le même pour le problème multicritère et sa version mono-objectif, l'ensemble des «bonnes» solutions peut être plus large dans le problème multicritère.

Sur le plan de la performance, si on regarde la valeur moyenne des solutions et le nombre de meilleures solutions trouvées (Tableau 2.9), on observe que la configuration *parametrized* l'emporte légèrement lorsqu'une partie des critères est prépondérante (μ_{cost}, μ_{QoS}), mais qu'elle est

Configuration		μ_{cost}	μ_{QoS}	μ_{avg}	μ_{bal}
Basic	Valeur moyenne de l'objectif	0,5073	0,7612	0,7101	0,5296
	Nombre de meilleures performances moyennes	4	7	7	6
	Ecart moyen à la meilleure moyenne	1,74%	0,23%	0,08%	0,32%
	Ecart max à la meilleure moyenne	14,96%	1,35%	0,44%	1,88%
Advanced	Valeur moyenne de l'objectif	0,5112	0,7605	0,7099	0,5304
	Nombre de meilleures performances moyennes	3	0	5	4
	Ecart moyen à la meilleure moyenne	0,63%	0,34%	0,11%	0,20%
	Ecart max à la meilleure moyenne	1,90%	1,87%	0,48%	0,98%
Parametrized	Valeur moyenne de l'objectif	0,5135	0,7621	0,7097	0,5295
	Nombre de meilleures performances moyennes	9	7	6	4
	Ecart moyen à la meilleure moyenne	0,22%	0,11%	0,14%	0,36%
	Ecart max à la meilleure moyenne	1,75%	0,71%	0,95%	1,63%

TABLE 2.9 – *Synthèse des expérimentations des configurations de l'algorithme LNS pour les différents modèles multicritères*

dépassée à chaque fois par une des autres configurations lorsque les critères ont la même importance.

En regard de la complexité d'exploration de l'espace de recherche, on évalue la capacité des algorithmes à trouver des solutions proches de la meilleure connue (robustesse). Sur ce plan on note un apport des opérateurs de la configuration *advanced* (il aurait été intéressant d'évaluer l'apport individuel du voisinage *route removal*, le critère lié au nombre de véhicules étant très discret). Les opérateurs d'insertion monocritères apportent également en robustesse dans les cas où quelques critères sont prépondérants. Ces observations sont corroborées par l'étude des résultats individuels de l'ensemble des exécutions des algorithmes.

De nombreuses illustrations sont données dans le papier (Lehuédé et al. 2013). On observe notamment que la performance d'une heuristique provient généralement de sa capacité à trouver souvent une solution proche de la meilleure connue sur les 10 exécutions (voir par exemple l'analyse donnée en Figure 2.12).

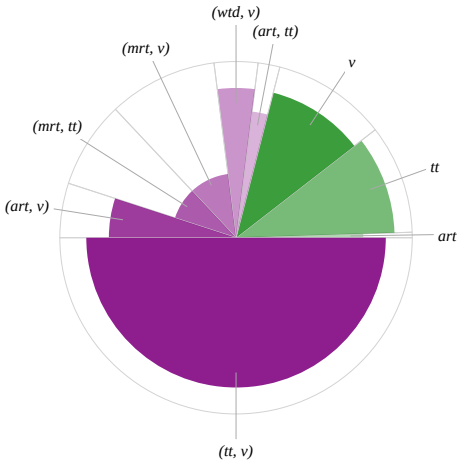
2.4.4 Conclusion et perspectives

En conclusion, ce travail constitue une première étude sur l'utilisation de l'intégrale de Choquet dans une métaheuristique. Une attention particulière est à porter lors de la définition du modèle multicritère sur la définition de métriques indépendantes de la taille du problème résolu. Pour la résolution, le modèle multicritère peut a priori être intégré comme une fonction objectif relativement classique. L'utilisation d'opérateurs guidés par les objectifs apporte un gain de performance et de robustesse à la méthode.

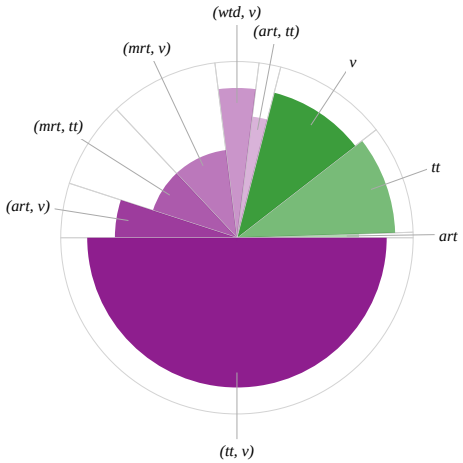
Sur cette étude, un grand nombre de développements, d'analyses et d'expérimentations ont été réalisées mais on peut noter plusieurs points d'amélioration pour de futurs travaux :

- Tout d'abord, l'évaluation de la qualité des solutions obtenues est relativement difficile. Il aurait été intéressant de pousser l'étude de sensibilité pour voir l'impact de la dégradation d'une métrique sur l'évaluation globale de la solution.

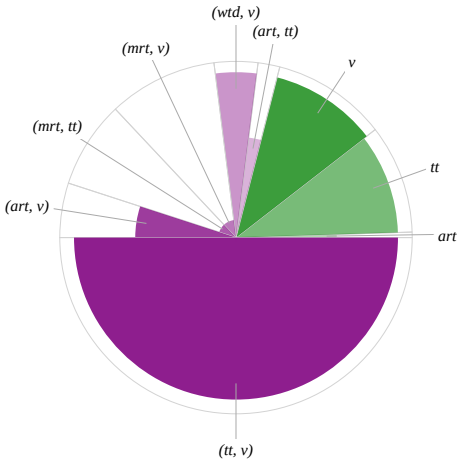
- L'analyse de l'espace de recherche et en particulier autour de la meilleure solution connue pourrait également être poussée : il serait intéressant par exemple de montrer la proportion



(a) $U_{\mu_{cost}}(x_a) = 0.6087$



(b) $U_{\mu_{cost}}(x_b) = 0.6266$



(c) $U_{\mu_{cost}}(x_c) = 0.6583$

		mrt	*art*	*wtd*	*tt*	*v*	$U_{\mu_{cost}}$
x_a	*u*	0.133	0.52	0.823	0.809	0.722	0.6087
	m	32.93	14.74	6.5	7.32	0.83	
	f	32.93	14.74	649.2	731.5	15	
x_b	*u*	0.251	0.48	0.743	0.807	0.722	0.6266
	m	30.93	15.25	7.44	7.35	0.83	
	f	30.93	15.25	742.8	734.7	15	
x_c	*u*	0.01	0.327	0.912	0.845	0.881	0.6583
	m	42.41	17.16	5.44	6.86	0.89	
	f	42.41	17.16	544	686	14	

(d) *Utility and metrix values for each solution*

FIGURE 2.12 – *Représentation de la valeur de l'intégrale de Choquet $C_{\mu_{cost}}$ pour 3 solutions x_a, x_b, x_c de l'instance center-100-3 :*
La proportion du disque couvert par chaque camembert représente la valeur de l'intégrale de Choquet, on y perçoit la part de chaque interaction dans le modèle. Le Tableau 2.12d montre les valeurs des utilités, métriques et des fonctions objectif initiales. Le caractère discret du nombre de véhicules apparaît clairement.
Sur 10 exécutions, la configuration basic trouve une solution de type x_c, la configuration advanced ne trouve ce type de solution que 3 fois sur 10, parametrized trouve une solution de ce type 7 fois sur 10.

de solutions explorées à moins de 1% de la meilleure solution dans un problème multicritère et dans un problème mono-objectif.

- Les temps de calcul des configurations *basic* et *advanced* sont supérieurs à *parametrized* en raison d'un nombre plus important de calculs de l'intégrale de Choquet et des fonctions d'utilité. Une implémentation plus incrémentale du modèle telle que proposée dans Lehuédé et al. (2006) est susceptible de réduire un peu ces temps.

Plus généralement, la qualité de l'exploration réalisée reste difficile à évaluer sans borne inférieure d'une solution optimale. La résolution exacte sur un problème tournées de véhicules Choquet-optimales reste un problème jamais abordé et certainement extrêmement complexe.

Enfin, l'exploration de l'espace de recherche semble rendue difficile par la présence de plusieurs critères. Des techniques issues des métaheuristiques multiobjectif et en particulier les méthodes basées sur le maintient d'une population diversifiée pourraient permettre de concevoir des algorithmes plus performants.

2.5 Conclusion et Perspectives sur l'Optimisation d'un Modèle Multicritère basé sur l'Intégrale de Choquet

En conclusion, mes travaux autour de l'optimisation de l'intégrale de Choquet ont porté à la fois sur des méthodes de résolution exactes et approchées. C'est certainement sur le côté exact que de nombreuses avancées sont à réaliser pour permettre la résolution de problèmes réels. Néanmoins le domaine de la résolution approchée reste encore le moins exploré.

Pour prendre un peu de recul sur l'utilisation de ce modèle on peut noter que d'un point de vue pratique, les modèles d'aide à la décision multicritère sont complexes à mettre en œuvre en optimisation. Le processus de construction du modèle étant presque aussi coûteux que celui de conception des algorithmes d'optimisation, de nombreuses applications font le choix tout à fait justifié de procédures plus simples. Tout d'abord, on obtient très souvent des résultats très intéressants en optimisant une fonction objectif et en offrant la possibilité de faire varier des limites sous forme de contraintes sur les autres objectifs. Ce principe peut être facilement étendu à deux objectifs où l'analyse d'un front Pareto est relativement aisée. Une autre possibilité est d'intégrer des fonctions d'agrégations plus simples, telles que la norme de Tchebycheff, intégrée par Sauvanet and Néron (2010) sur la plateforme géo-vélo (La compagnie des mobilités 2014), pour le calcul d'itinéraires adaptés aux cyclistes. On peut néanmoins noter qu'il reste indispensable d'établir la commensurabilité des critères avant agrégation. Au delà du modèle d'agrégation le travail sur la formalisation de fonctions objectif pertinentes est également essentiel. Une perspective proche sur ce plan est la modélisation d'un critère d'équilibrage des charges pour le VRP. Cette recherche a été récemment initiée avec Fabien Tricoire dans la poursuite des travaux sur le DARP multicritère.

Cependant, lorsque l'intégration de préférences est stratégique pour l'acceptabilité de l'application, ou que le mode d'utilisation de l'outil d'aide à la décision ne permet pas d'itérations entre l'utilisateur et le logiciel, certains organismes n'hésitent pas à investir sur les deux champs. Par exemple, concernant l'intégrale de Choquet, le logiciel PTIME (Berry et al. 2011) est un assistant personnel à la planification de rendez-vous basé sur l'intégrale de Choquet et la programmation par contraintes. Le mode d'apprentissage du modèle est ici très intéressant : l'utilisateur exprime

dans un premier temps des préférences relativement simples en indiquant des préférences relatives entre critères. Des préférences entre solutions sont ensuite générées au fur et à mesure de l'utilisation de l'outil, pour affiner le modèle, et sans qu'il n'ait à les exprimer explicitement. On pourrait donc investiguer davantage l'extension de méthodes interactives pour l'aide à la décision multicritère (Vanderpooten 1989, Branke et al. 2008) en y intégrant un apprentissage des préférences pour un modèle basé sur l'intégrale de Choquet. Un outil de ce type pourrait proposer à l'expert de s'exprimer sur des comparaisons de solutions pour lequel le modèle multicritère est mal défini. Cette piste a été initiée dans le cadre du stage de Master d'Hugo Fouchal (Fouchal 2009), mais sans poursuites. Très récemment, la minimisation de l'interaction avec l'expert a été étudiée avec la dénomination « élicitation incrémentale » de l'intégrale de Choquet par Benabbou et al (Benabbou et al. 2014).

Dans l'univers du «big data» où l'on amasse un grand nombre de données, l'utilisation de l'intégrale de Choquet comme un outil puissant d'apprentissage des préférences, sans que la phase d'élicitation des préférences ne soit visible par l'expert, offre des voies très intéressantes. Pour se cantonner au transport, l'application communautaire Waze (Waze Mobile 2014), se base à la fois sur les informations remontées par une communauté d'utilisateurs et sur les usages du conducteur pour proposer un chemin préféré. Ce principe est repris dans l'application Copilot (ALK Technologies 2014), qui propose d'adapter ce principe à un usage professionnel pour la construction de tournées de véhicules. On distingue dans ces applications trois sources d'informations : l'information trafic remontée en temps-réel, l'historique des itinéraires empruntés par le conducteur et les informations saisies par l'utilisateur lors du calcul de chemin. Une couche d'apprentissage des préférences basée sur un modèle MAUT et l'intégrale de Choquet pourrait être ici tout à fait pertinente.

Pour finir, on peut déplorer que l'agrégation d'objectifs ou d'indicateurs hétérogènes à l'aide d'une somme pondérée reste un réflexe pour beaucoup d'ingénieurs ou chercheurs dès qu'il s'agit d'établir une évaluation globale d'objets complexes. Un travail d'éducation est à fournir avec la mise en exergue d'exemples montrant les mauvaises propriétés d'une somme pondérée mal construite et la présentation des outils de l'AMCD (voir par exemple Vallin and Vanderpooten (2000)).

Chapitre 3

Problèmes de Tournées de Véhicules avec Synchronisation

Les problématiques de synchronisation en tournées de véhicules constituent un champs d'investigation encore peu exploré mais maintenant clairement identifié. Sous la dénomination synchronisation sont regroupées les caractéristiques qui créent une interdépendance entre tournées. Dans un monde où le transport se veut de plus en plus multi-modal, inter-connecté et où les systèmes d'information permettent un accroissement de la flexibilité pour atteindre davantage de performance, nul doute que de nouveaux problèmes de tournées avec synchronisation vont continuer à apparaître. Dans ce cadre, l'exploration de l'espace de recherche et la vérification des contraintes temporelles offrent des sujets de recherche tout à fait intéressants, à la croisée entre optimisation de tournées et ordonnancement. Ce chapitre présente la synthèse de plusieurs travaux, fruits de trois thèses (Renaud Masson, Philippe Grangier, Axel Grimault), dont deux sont toujours en cours, et de multiples collaborations.

3.1 Introduction

La classe des problèmes de tournées de véhicules avec synchronisation (VRPMS : Vehicle Routing Problems with Mutiple Synchronization constraints) a été récemment introduite dans l'article état de l'art Drexl (2012) et définie de la manière suivante :

> *A VRPMS is a vehicle routing problem in which more than one vehicle may or must be used to fullfill a task.*

Cet état de l'art montre l'aspect émergent de la problématique avec un grand nombre de travaux identifiés dans les trois ans avant son écriture (2011). Ce sujet a fait l'objet du worshop SynchroTrans à Mainz en 2013, d'une session spéciale à la conférence Verolog à Bologne en 2012 et à OSLO en 2014, et d'une session invitée à INCOM en 2015 à Ottawa. Le nombre continuellement croissant de publications sur le sujet est également souligné dans le récent rapport de Drexl (2014).

La synchronisation est au cœur des réseaux de transports modernes, de plus en plus réactifs. C'est d'abord le cas des réseaux logistiques basés sur des cross-docks (Van Belle et al. 2012). Ces plateformes modernes permettent une consolidation des flux venant de tournées de collecte, des expéditions en camions pleins vers d'autres cross-docks éloignés et un éclatement des flux dans des tournées de distribution. Un des avantages majeur d'un réseau de cross-docks est sa capacité à répondre aux demandes de transport dans des délais records. Les problèmes de planification en découlant sont complexes et directement liés à la synchronisation temporelle entre véhicules Gonzalez-Feliu (2013). Les problèmes de tournées de véhicules autour de cette organisation en réseau sont multiples. Le problème à un unique cross-dock a par exemple été récemment étudié par Wen et al. (2009) et Santos et al. (2013) et fait toujours l'objet d'investigations. La pratique actuelle des logisticiens consiste à réduire la flexibilité du réseau en figeant une partie de l'organisation (sectorisation par cross-dock et par chauffeur, heures de départ des camions complets,...). Le coût de ces contraintes reste à évaluer, notamment via la création d'algorithmes d'optimisation capable de réaliser une optimisation globale du réseau. L'idée d'une plus forte flexibilité dans le futur est également très présente dans les réseaux multi-modaux. Le concept de synchro-modalité est notamment proposé dans le récent état de l'art de SteadieSeifi et al. (2014) sur ce thème.

En second lieu, la logistique urbaine est un domaine où se développent de plus en plus de modèles flexibles et synchronisés pour livrer les centres villes. Ce thème et les enjeux inhérents font l'objet du livre de Gonzalez-Feliu et al. (2014), issu du projet ANR MODUM, achevé en 2014. La logistique du dernier kilomètre est en effet reconnue comme un problème majeur, générant des coûts, de la pollution et de la congestion, aussi bien en ville que dans les périphéries ou arrière pays. Les systèmes de transport innovants imaginés pour résoudre ces problèmes vont vers des systèmes à plusieurs niveaux Crainic et al. (2009), Mancini et al. (2014), collaboratifs (Morana et al. 2014), ou partagés par les marchandises et les passagers (Trentini 2012, Li et al. 2014). Ces systèmes impliquent pour la plupart une meilleure utilisation des moyens, notamment en créant des points de consolidation et d'échange entre véhicules. Cattaruzza et al. (2015) présente un état de l'art des problèmes de tournées étudiés en logistique urbaine. Dans ce domaine, les algorithmes de recherche opérationnelle ont un clairement rôle à jouer, en particulier en tant qu'outils de simulation de réseaux encore existants.

Un troisième domaine majeur où la synchronisation se retrouve dans des problèmes de tournées concerne les transports de personnes. En effet, si les villes sont bien couvertes par les

transports public, de nombreuses questions se posent en périphérie et dans les zones peu denses en population. La mobilité des personnes dans les zones rurales est un problème de fond et les systèmes de transport à la demande peinent à satisfaire les besoins de la population. Une piste est récemment mis en évidence, avec l'utilisation combinée de lignes flexibles de transport à la demande et de lignes régulières Lee and Savelsbergh (2014). La combinaison de petits véhicules avec des lignes de bus est également le sujet du challenge VeRoLog 2015 proposé par l'éditeur de logiciel de tournées de véhicules PTV.

Par ailleurs, les problèmes de synchronisation apparaissent dans de nombreuses autres applications : les tournées de personnel (médical ou technique) (Bredström and Rönnqvist 2008, Kovacs et al. 2012) ; en logistique forestière (El Hachemi et al. 2013) ; dans les problèmes de tournées avec moyens passifs et actifs (par exemple en présence de remorques détachables) (Drexl 2011, Meisel and Kopfer 2014) ; ou dans le cadre d'opérations de déneigement ou de peinture des routes (Salazar-Aguilar et al. 2012; 2013).

Sur le plan algorithmique, les problèmes de tournées de véhicules avec synchronisation sont présentés comme particulièrement difficiles en raison de l'*inter-dépendance* créée entre les tournées. En présence de fenêtres de temps notamment, la modification d'une tournée peut en rendre une autre non réalisable.

3.1.1 Organisation de la présentation des travaux

Mes premiers travaux sur cette thématique ont été initiés en collaboration avec Olivier Péton dans le cadre de la thèse de Renaud Masson qui a porté sur la résolution de problèmes de collectes et livraisons avec transferts. Ces problèmes sont issus des problèmes de transport de personnes handicapés étudiés lors du développement du logiciel Marika. Ces travaux sont actuellement poursuivis avec la thèse de Philippe Grangier, co-encadré avec Michel Gendreau et Lous-Martin Rousseau. Cette thèse porte sur la résolution de problèmes de logistique urbaine ou de tournées avec cross-dock. L'aspect synchronisation apparaît également via des contraintes sur les ressources de chargement et déchargement des camions dans un problème de tournées de camions pleins pour la construction de routes. Ce problème fait partie du projet Orloges. Il est actuellement traité dans la thèse d'Axel Grimault, co-encadré avec Nathalie Bostel.

Deux cadres communs sont développés à travers ces trois thèses : pour la recherche de solutions, avec l'algorithme ALNS ; et pour l'évaluation de la réalisabilité temporelle des solutions. Ce chapitre tente en premier lieu de présenter une vue unifiée de ces travaux : Les problèmes étudiés sont présentés en Section 3.2. Ils sont résolus avec l'algorithme ALNS dont les principes sont brièvement rappelés en Section 3.3. Les aspects exploration de l'espace de recherche et les opérateurs développés pour ces problèmes sont ensuite mis en regard en Section 3.4. Finalement, les aspects réalisabilité sont discutés plus en détail en Section 3.5.

La Section 3.6 présente trois contributions qui sortent de ce cadre méthodologique : une approche exacte pour un problème de collectes et livraisons avec transferts ; un cas d'étude en logistique urbaine ; et une méthode innovante pour un problème de transport de personnes avec un critère de régularité.

3.2 Problématiques

On considère ici quatre problèmes de de tournées qui, selon la classification de Drexl (2012), possèdent différents types de contraintes de synchronisation. Ces quatre cas peuvent être vus comme des problèmes de tournées avec collectes et livraisons.

3.2.1 Notations

La présentation des problèmes est basée sur les notations suivantes :

On considère un ensemble R de requêtes de transport. Chaque requête $r \in R$ correspond à une demande de transport de q_r unités d'un point p_r à un point d_r. L'ensemble des points de collecte est noté P et l'ensemble des points de livraison est noté D.

L'ensemble des véhicules disponibles est noté K. Un véhicule $k \in K$ réalise une tournée entre un point de départ $o(k)$ et un point d'arrivée $o'(k)$. On note $O = \cup_{k \in K} o(k)$ et $O' = \cup_{k \in K} o'(k)$.

Les problèmes sont modélisés sur un graphe $G = (V, A)$, spécifique à chaque problème, avec néanmoins $P \cup D \cup O \cup O' \subseteq V$.

Pour tout véhicule $k \in K$ et tout arc $(i, j) \in A$, on note c_{ij}^k et t_{ij}^k les coûts et temps de trajet respectifs du véhicule k sur le chemin entre i et j. Dans le cas d'une flotte homogène de véhicules, on utilisera les notations c_{ij} et t_{ij} pour ces valeurs. On suppose que l'inégalité triangulaire sur les temps de parcours est respectée.

A tout sommet $i \in V$ on peut associer une fenêtre de temps $[e_i, l_i]$ qui délimite l'heure de début du service en i. La durée du service à ce point est notée s_i. Un véhicule doit arriver avant l_i pour servir ce sommet, s'il arrive avant e_i, il devra attendre cette date pour y démarrer le service. Dans une solution, l'heure de début du service d'un sommet $i \in V$ est notée h_i.

3.2.2 Deux problèmes de tournées de collectes et livraisons avec transferts (PDPT et DARPT)

Le PDPT (*Pickup and Delivery Problem with Transfers*) est une généralisation du problème classique de tournées de véhicules avec collectes et livraisons où l'on autorise qu'une requête soit transférée d'un véhicule à un autre à un type de lieu spécifique appelé point de transfert.

Dans ce problème on considère un ensemble de points de transferts noté T. Les véhicules de l'ensemble K sont identiques et de capacité Q. Le graphe du PDPT est définit par $V = P \cup D \cup T \cup O \cup O'$ et $A = O \times P \cup \{P \cup D \cup T\}^2 \cup D \times O' \cup \{(o(k), o'(k)) | k \in K\}$.

Une solution du PDPT est un ensemble de $|K|$ tournées desservant l'ensemble des requêtes de telle sorte que : pour chaque requête $r \in R$, les sommets p_r et d_r sont soit servis par la même tournée (et alors p_r est servi avant d_r dans cette tournée), soit ils sont servis par deux tournées distinctes $k \in K$ et $k' \in K$. Dans ce dernier cas, les tournées k et k' visitent toutes deux un point de transfert $\tau \in T$, de telle sorte que τ soit visité après d_r dans k et avant d_r dans k'. La requête r est déchargée du véhicule k au point τ et rechargée ensuite dans k' à ce point.

Le coût d'une tournée est la somme des coûts des arcs qu'elle emprunte. L'objectif est de trouver un ensemble de tournées de coût minimal.

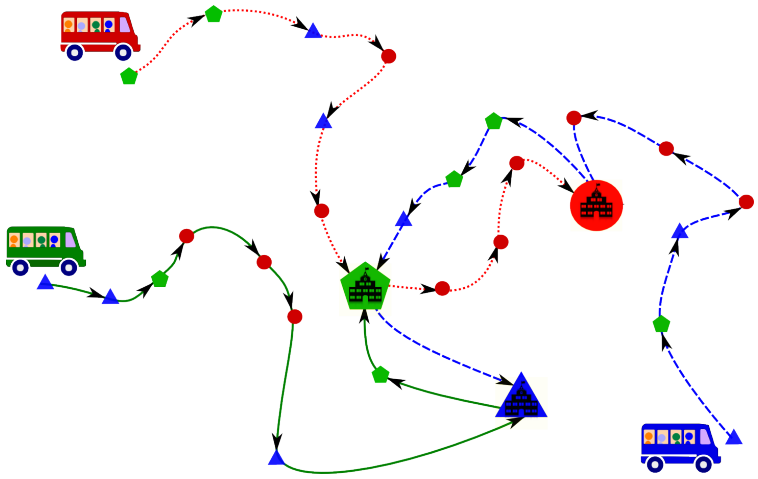


FIGURE 3.1 – *Une solution à trois véhicules pour un PDP.*
Les ronds rouges, triangles bleu et pentagones verts se rendent à l'établissement représenté par une forme correspondante. Les centres tiennent lieu de points de transfert. La tournée verte, en traits continus, charge trois ronds rouges qui sont transférés dans la tournée rouge en pointillés courts au centre vert (pentagone). Cette dernière tournée dépose deux triangles bleus au centre vert, qui sont chargés dans la tournée bleue (pointillés longs) pour être menés à destination.

On voit ici que l'interdépendance entre tournées est créée par la contrainte de précédence entre le service d'un point τ par deux tournées lors d'un transfert. Durant cette opération, les deux véhicules n'ont cependant pas lieu d'être présents ensemble sur le point de transfert. Ce type de synchronisation est appelé *synchronisation d'une opération avec précédences* (Drexl 2012).

Une solution du PDPT est illustrée en Figure 3.1 pour un problème de transport de personnes vers plusieurs établissements. Le PDPT a été étudié en particulier par Shang and Cuff (1996), Mitrović-Minić and Laporte (2006), Cortés et al. (2010), Qu and Bard (2012), Masson et al. (2013a;b).

Dans le cas du transport de passagers, il est d'usage de prendre en compte une durée de trajet maximum $\bar{L}_r$ pour toute requête $r \in R$ (c'est la durée écoulée entre le service de son point de départ p_r et le service de sa destination d_r). Le problème ainsi modifié est alors appelé «Problème de transport à la demande avec transferts» (*DARPT : Dial–A–Ride Problem with Transfers* (Masson et al. 2014a)).

3.2.3 Le problème de tournées de véhicules multi-trip à deux échelons avec synchronisation aux satellites (2E–MTVRP–SS)

Le 2E–MTVRP–SS (Two-Echelon Multiple-Trip Vehicle Routing Problem with Satellite Synchronization) est issu d'un modèle innovant de distribution de marchandises visant à réduire la congestion dans les centre villes en logistique urbaine (Crainic et al. 2009). Il est basé sur un système à deux échelons : depuis un centre de distribution urbain, une flotte de gros porteurs dessert des satellites via les axes principaux (premier échelon) ; sur ces espaces, les marchandises sont transférées vers des véhicules légers qui réalisent les derniers kilomètres jusqu'au point de livraison (deuxième échelon). Ce système est illustré en Figure 3.2.

Le centre de distribution urbain est noté CDC (City Distribution Center), l'ensemble des satellites est noté V_Δ. Les requêtes de transport sont servies à l'aide de deux flottes de véhicules

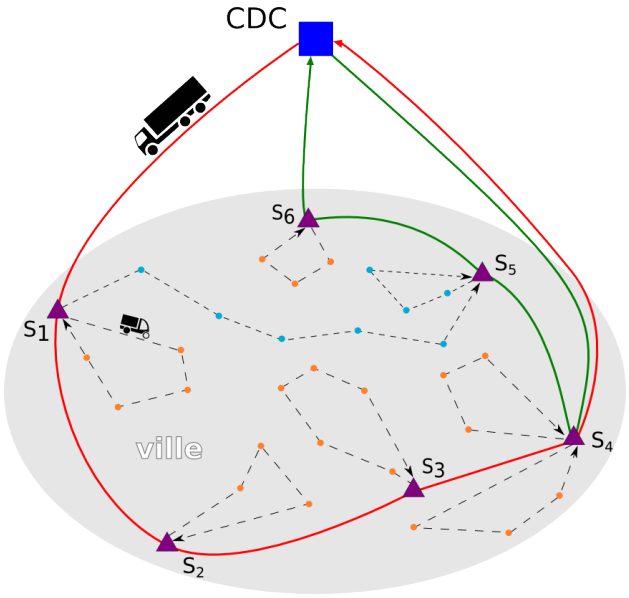


FIGURE 3.2 – *Un exemple de problème de tournées multi-trip à deux échelons et contraintes de synchronisation aux satellites*

K_1 et K_2 comportant respectivement des véhicules de capacités Q_1 et Q_2. Les dépôts pour chaque flotte sont notés o_1 et o_2. Les requêtes $r \in R$ partent toutes du CDC en début de période ($P = \{CDC\}$). Les livraisons directes depuis le CDC ne sont pas autorisées. Les véhicules du second niveau réalisent plusieurs tournées de livraison et peuvent être approvisionnés sur n'importe quel satellite. Le stockage de marchandises aux satellites n'étant pas autorisé, on impose que deux véhicules du premier et second échelon devant se transmettre une charge soient présents simultanément à un satellite pendant la durée du transfert. Cette forme de synchronisation est appelée « *synchronisation exacte d'opération* » (Drexl 2012).

Le 2E–MTVRP–SS est défini sur un graphe orienté $G = (V, A)$ qui reflète le système à deux échelons. Le premier échelon est défini par le graphe $G_1 = (V_1, A_1)$, avec $V_1 = \{o_1\} \cup \{CDC\} \cup V_\Delta$ et $A_1 = \{(o_1, CDC)\} \cup \{(CDC, i) | i \in V_\Delta\} \cup \{(i, j) | i, j \in V_\Delta\} \cup \{(i, o_1) | i \in V_\Delta\}$. Le second échelon est défini par $G_2 = (V_2, A_2)$ avec $V_2 = \{o_2\} \cup D \cup V_\Delta$ et $A_2 = \{(o_2, i) | i \in V_\Delta\} \cup \{(i, j) | i \in V_\Delta, j \in D\} \cup \{(i, j) | i, j \in D\} \cup \{(i, j) | i \in D, j \in V_\Delta\} \cup \{(i, o_2) | i \in D\}$.

Le problème consiste à définir un ensemble de tournées pour les véhicules de chaque échelon permettant de servir l'ensemble des requêtes en minimisant lexicographiquement les objectifs suivants : le nombre de véhicules utilisés au premier échelon ; le nombre de véhicules utilisés au second échelon ; la somme des coûts des arcs empruntés par les véhicules.

Les travaux sur ce problème font l'objet d'un rapport technique (Grangier et al. 2014) en cours de révision pour publication dans la revue EJOR.

3.2.4 Le problème de tournées de collectes et livraisons en camions pleins avec synchronisation sur les ressources (FT–PDP–RS)

Le FT–PDP–RS (Full Truckload Pickup and Delivery Problem with Resource Synchronization) a été abordé dans le cadre du projet Orloges où il est question d'optimiser les transports en travaux public entre sites (d'extraction, stockage, recyclage et production de matériaux) et

chantiers de construction de routes. Dans ce contexte, un ensemble de demandes de tailles supérieures aux capacités des véhicules sont divisées en un ensemble de requêtes de transport en camions pleins (Grimault et al. 2014). Un site ou chantier est origine ou destination de nombreuses requêtes. Dans certains cas, le chargement ou déchargement utilise une ressource sur le site qui impose un ordonnancement des opérations sur ce point. C'est le cas notamment de l'application d'un enrobé sur un chantier : pour cette opération, le véhicule est déchargé au fur et à mesure de l'application dans un *finisseur*. L'objectif est de déterminer un ensemble de tournées de coût minimum pour une flotte hétérogène de véhicules ayant différentes capacités, fonctionnalités et coûts, de manière à servir l'ensemble des requêtes. Une solution de ce problème est illustrée en Figure 3.3.

Chaque requête r correspond à un trajet entre son origine $p_r \in P$ et sa destination $d_r \in D$, qui peut être réalisé par un ensemble de véhicules compatibles K_r (la quantité transportée n'importe pas, la requête occupe un camion plein si celui-ci est compatible). Le coût fixe journalier d'utilisation d'un véhicule $k \in K$ est noté f_k. Le problème est modélisé sur un graphe $G = (V, A)$, avec $V = P \cup D \cup O \cup O'$ et $A = O \times P \cup \{(p_r, d_r) | r \in R\} \cup D \times P \cup D \times O'$. On note S l'ensemble des ressources du problème. Pour toute ressource $s \in S$ on note $V_s \subset P \cup D$ l'ensemble des sommets qui correspondent à des opérations qui ne peuvent pas se chevaucher sur cette ressource. Pour deux requêtes $r, r' \in R$ utilisant une ressource commune $s \in S$ à leur point de livraison ($d_r, d_{r'} \in V_s$), on a donc $h_{d_r} + s_{d_r} \leq h_{d_{r'}}$ ou $h_{d_{r'}} + s_{d_{r'}} \leq h_{d_r}$. On parle ici de *synchronisation sur les ressources* (Drexl 2012).

La fonction objectif du problème intègre un coût fixe par véhicule utilisé et un terme dépendant du nombre de kilomètres parcourus pour chaque type de véhicule.

3.2.5 Caractéristiques des problèmes et réalisabilité

Une particularité du travail poursuivi à travers ces différents projets est le développement d'une approche commune pour la vérification des contraintes temporelles dans des problèmes de synchronisation. Deux conditions doivent être vérifiées pour sa mise en œuvre :

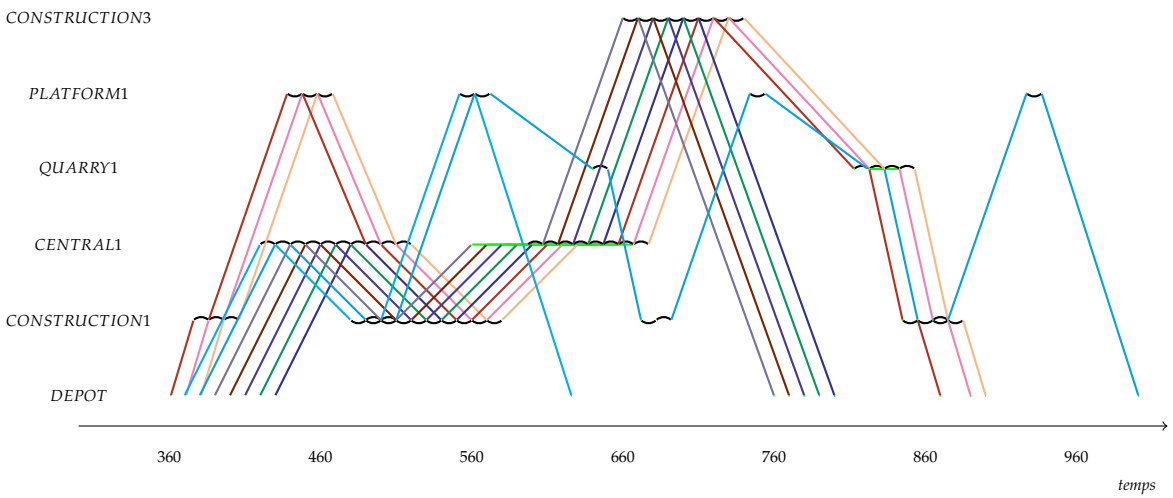
- il n'existe pas de contraintes d'écart temporel maximum entre le service de deux sommets (lag times) ;
- les tournées peuvent être ordonnancées « au plus tôt ».

Les algorithmes de réalisabilité développés pour ce type de problème sont présentés en Section 3.5.1.

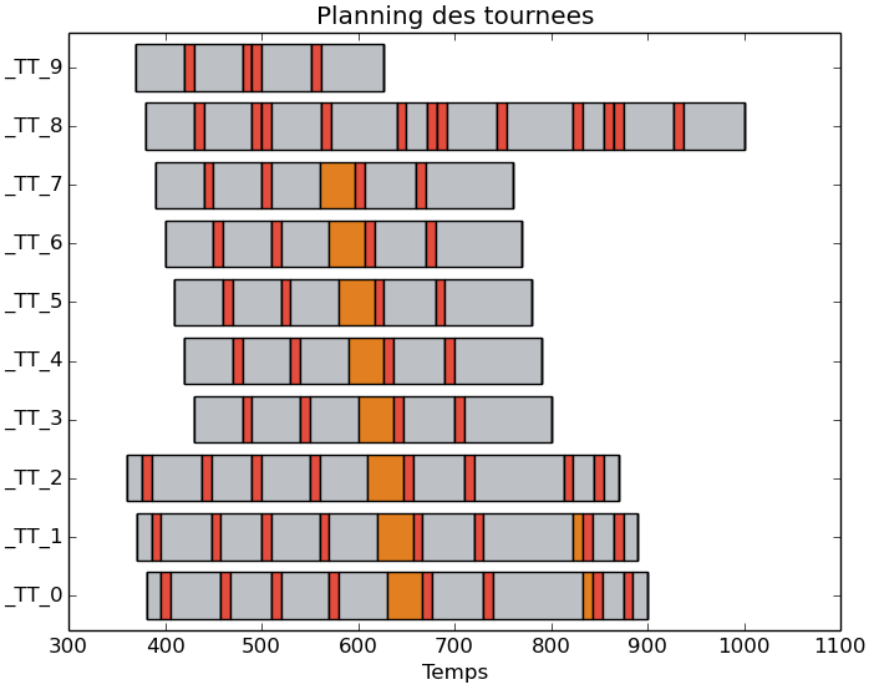
L'intégration de contraintes d'écart maximum a été étudiée dans le cadre du transport de personnes où la durée d'un trajet doit être limitée. Les procédures de réalisabilité développées sont alors plus complexes, ce qui nous amène à proposer des conditions suffisantes ou nécessaires pour réduire les temps de calcul (Section 3.5.2).

3.3 Méthode de résolution : recherche locale à voisinage large

En raison de leur complexité, les précédentes problématiques ont surtout été résolues de manière approchée. Le cadre de résolution choisi est l'algorithme ALNS (Adaptive Large Neighborhood Search), proposé par Ropke and Pisinger (2006a) pour la résolution du problème de



(a) *Représentation du passage des tournées sur les sites : Les lignes représentent le trajet des tournées sur les différents sites en fonction du temps. A la visite d'un site, une bosse représente un chargement, un creux représente un déchargement, un trait horizontal vert représente une attente.*



(b) *Planning des tournées. Pour chaque véhicule, on représente les opérations réalisées en fonction du temps. Un rectangle gris clair représente un trajet, un rectangle rouge (foncé) représente un chargement ou un déchargement, un rectangle orange représente une attente.*

FIGURE 3.3 – *Représentations d'une solution du FT–PDP–RS*

tournées de véhicules avec collectes et livraisons et fenêtres de temps (PDPTW : Pikup and Delivery Problem with Time Windows). Cette métaheuristique a montré sa performance sur un grand nombre de problèmes de tournées de véhicules (Ropke and Pisinger 2006b, Pisinger and Ropke 2007). Cet algorithme est toujours référencé parmi les meilleurs pour la résolution du PDPTW (Battarra et al.).

3.3.1 L'ALNS : cadre général

L'Algorithme 1 propose une présentation de l'ALNS, inspirée de Sørensen et al. (2012).

Algorithme 1 Adaptive Large Neighborhood Search

Requis : Solution initiale x, opérateurs de retrait Ω^-, opérateurs d'insertion Ω^+

$x^\star \leftarrow x$
tant que le critère d'arrêt n'est pas satisfait **faire**
  $x' \leftarrow x$
  Nombre de retraits : sélection d'un nombre q de requêtes à retirer
  Sélection des opérateurs : sélection d'un opérateur de retrait $or \in \Omega^-$ et d'un opérateur d'insertion $oi \in \Omega^+$
  Retrait : $\mathcal{L} \leftarrow \mathcal{L}_{x'} \cup or(x', q)$
  Insertion : $x' \leftarrow oi(x', \mathcal{L})$
  si $f(x') \leq f(x^\star)$ **alors**
    $x^\star \leftarrow x'$
  fin si
  Critère d'acceptation : initialiser la solution courante x à x', $x^\star$ ou x
fin tant que
retourner $x^\star$

Les principales caractéristiques de l'algorithme sont les suivantes :

- A chaque itération, un opérateur de retrait (destruction) et un opérateur d'insertion (reconstruction) sont sélectionnés pour construire un voisin de la solution courante. Le nombre de voisinages possible est donc $|\Omega^-| \times |\Omega^+|$, mais ces voisinages peuvent être couplés.
- Le nombre de requêtes q à retirer de la solution courante est pris aléatoirement à chaque itération dans un intervalle $[\underline{\xi}, \bar{\xi}]$. Ce choix détermine la taille du voisinage, il a une influence notable sur le temps de calcul. Dans Pisinger and Ropke (2007), cet intervalle est $[\min(0, 1 \times |R|, 30), \min(0, 4 \times |R|, 60)]$.
- La stratégie de **Sélection des opérateurs** est *adaptative* : à chaque itération, un opérateur $i \in \Omega^- \cup \Omega^+$ est sélectionné aléatoirement en fonction d'une probabilité de sélection qui lui est attribuée pour un nombre donné d'itérations (time segment). Cette probabilité est révisée à la fin de chaque time segment sur la base d'un score π_i, incrémenté de diverses valeurs à chaque itération suivant sa capacité à trouver : de nouvelles meilleures solutions ; des solutions non visitées qui améliorent la solution courante ; des solutions non visitées de qualité inférieures à x'.

- A chaque solution x est associée une **liste de requêtes non insérées** $\mathcal{L}_x$ (*request bank*). Une solution de l'ALNS peut donc être *partielle*. Le nombre de requêtes dans $\mathcal{L}_x$ est pénalisé dans la fonction objectif.
- Le **critère d'acceptation** détermine si une solution x' doit être acceptée comme nouvelle solution courante ou non. La plupart des implémentations de l'ALNS se basent sur un critère de type *recuit simulé* pour réaliser ce choix. Lorsque la solution courante est trop éloignée de la meilleure solution connue, il est aussi possible de la réinitialiser à $x^\star$.

3.3.2 Opérateurs de la littérature

De nombreux opérateurs ont été développés dans différentes versions de la métaheuristique Shaw (1998), Ropke and Pisinger (2006a;b), Pisinger and Ropke (2007), Prescott-Gagnon et al. (2009a), Demir et al. (2012). Les plus utilisés sont les suivants :

Opérateurs de retrait

Chaque opérateur de retrait retire q requêtes de transport de la solution courante et les place dans $\mathcal{L}$.

1. **Retrait aléatoire (Random Removal)**. Cet opérateur sélectionne aléatoirement q requêtes de la solution courante.
2. **Retrait des requêtes les plus coûteuses (Worst Removal)**. Cet opérateur est le symétrique d'une heuristique de meilleure insertion. Sur q itérations, cet opérateur sélectionne la requête dont le retrait génère la plus forte réduction de coût et la place dans la liste $\mathcal{L}$.
3. **Retrait de requêtes proches (related removal)**. Ce voisinage est basé sur une mesure de proximité entre requêtes et suit le processus suivant : La liste $\mathcal{L}$ est initialisée avec une requête sélectionnée aléatoirement dans la solution courante. A chaque itération, une requête r est tirée aléatoirement dans $\mathcal{L}$. Une requête r' est sélectionnée dans la solution courante et placée dans $\mathcal{L}$ avec une probabilité croissante en fonction de la proximité avec la requête r. La procédure de Shaw (1998), utilisée dans Ropke and Pisinger (2006a), mesure la proximité via une mesure agrégeant distances, heures de service et quantités (*shaw removal*). Dans plusieurs travaux ultérieurs (Pisinger and Ropke 2007, Prescott-Gagnon et al. 2009b, Lehuédé et al. 2013), deux opérateurs sont utilisés : l'un est basé uniquement sur la distance entre les requêtes (*distance related*), l'autre est basé sur la proximité des heures de service (*time related*).

Opérateurs d'insertion

Les opérateurs d'insertion tentent d'insérer l'ensemble des requêtes de $\mathcal{L}$ dans la solution courante. Les requêtes qui ne peuvent être insérées sont conservées dans la liste de requêtes non insérées de la nouvelle solution.

1. **Meilleure insertion** : A chaque itération, le meilleur coût d'insertion est calculé pour chaque requête non planifiée et la requête ayant le plus petit coût d'insertion est insérée à sa meilleure position.

2. k**-regret** : la mesure de regret de niveau 2 d'une requête non insérée est la différence entre : le coût d'insertion de cette requête à la meilleure position dans sa seconde meilleure tournée ; et le coût de son insertion à la meilleure position dans sa meilleure tournée. Ce principe est étendu par Ropke and Pisinger (2006a) à la somme des regrets de niveau 2 à k, il permet de définir $k-1$ heuristiques d'insertion.

3.3.3 Un cadre propice pour la prise en compte de contraintes de synchronisation

Tout comme il l'a été pour la résolution de problèmes de tournées avec collectes et livraisons, qui supposent déjà une précédence entre point de collecte et point de livraison dans une tournée, l'algorithme ALNS me paraît particulièrement adapté à la résolution de problèmes avec synchronisation.

Comparativement à d'autre méthodes de recherche locale, une première observation porte sur l'impact des modifications réalisées sur la solution courante : dans le cadre de tournées avec fenêtres de temps, les voisinages qui réalisent d'importantes modifications dans la structure de la solution ont du mal à rester sur des solutions réalisables. Les algorithmes à base de croisements ou de recherche locale passent alors souvent par une pénalisation de la violation de contraintes temporelles dans la fonction objectif. L'ALNS lui ne réalise que des insertions. Les précédences existantes dans la solution courante ne sont pas remises en cause. Il est alors beaucoup plus aisé de rester sur des solutions partielles réalisables et de ne tester que des insertions qui ne remettent pas en cause cette réalisabilité.

Le second argument porte sur des aspects pratiques dans l'implémentation des algorithmes : la vérification de la réalisabilité d'une solution dans un problème avec synchronisation est nécessairement complexe. Le test de réalisabilité d'un voisin dans l'ALNS et réalisé par l'évaluation des diverses formes d'insertion, qui restent en nombre limité. Il n'est pas nécessaire de développer des procédures différentes pour un grand nombre de voisinages.

3.4 Exploration de l'Espace de Recherche

Les problèmes présentés en Section 3.2 ont été résolus avec des algorithmes ALNS. Pour chaque problème, des heuristiques adaptées ont été conçues. Cette section synthétise les aspects les plus intéressants de ces algorithmes en se focalisant particulièrement sur les développements relatifs à la synchronisation.

3.4.1 Guidage de la recherche pour l'insertion de points de consolidation

Dans le PDPT, le DARPT et le 2E–MTVRP–SS, les points de synchronisation (points de transfert, satellites), sont des points de consolidation. Il est rarement intéressant de les utiliser pour un faible nombre de requêtes. Dans les heuristiques d'insertion de l'ALNS, qui insèrent les requêtes une par une, il faut trouver des solutions pour faire apparaître ces points dans les tournées.

Guidage pour le PDPT / DARPT

Les algorithmes de recherche de solutions pour le PDPT et le DARPT, présentés respectivement dans Masson et al. (2013a) et Masson et al. (2014a), sont relativement proches. Les

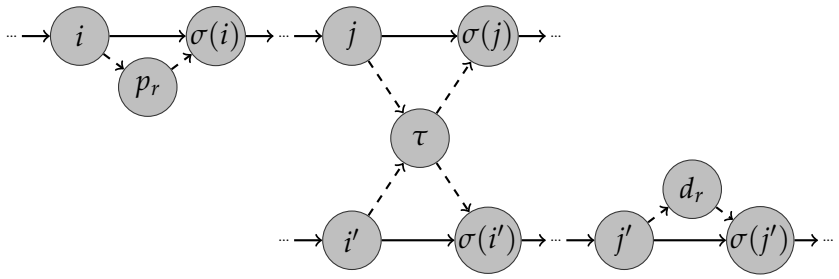


FIGURE 3.4 – *Exemple d'insertion d'une requête r via un point de transfert τ.*

heuristiques réalisent, entre autres, des insertions avec transfert. Ce type d'insertion est illustrée en Figure 3.4.

Pour l'insertion dans le PDPT on ajoute aux heuristiques de la littérature trois heuristiques permettant d'ajouter des transferts :

Meilleure insertion avec transferts : considère parallèlement les insertions avec et sans transfert pour les requêtes non insérées et réalise la meilleure à chaque itération.

Transfert d'abord Comme illustré en Figure 3.5, souvent, l'utilisation d'un point de transfert ne devient intéressante que si plusieurs requêtes sont consolidées sur celui-ci. Il peut alors être bénéfique d'imposer des transferts qui paraissent localement sous-optimum. L'opérateur *transfert d'abord* réalise en priorité les insertions avec transfert (pour une requête donnée, une insertion sans transfert n'est cherchée que si aucune insertion avec transfert n'est réalisable). Lorsque toutes les requêtes ont été insérées, pour chaque requête transférée, on essaye un retrait suivi d'une réinsertion sans transfert pour supprimer les transferts non bénéfiques.

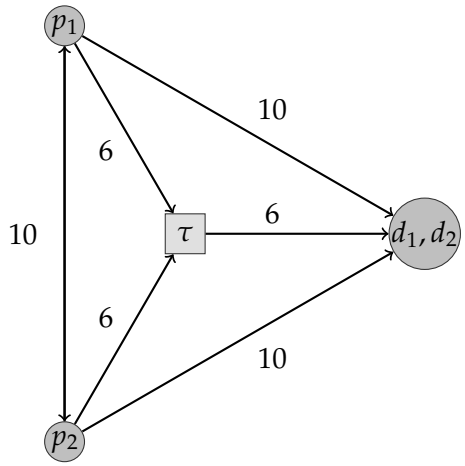


FIGURE 3.5 – *Exemple où les insertions successives n'utilisent pas le point de transfert*

Insertion avec regret pour les transferts Cette heuristique a pour objectif d'insérer au plus tôt les requêtes susceptibles de bénéficier d'un transfert. Elle calcule un regret égal au coût de la meilleure insertion sans transfert moins celui de la meilleure insertion avec transfert pour chaque requête. A chaque itération, la requête de regret maximum est insérée à sa meilleure position.

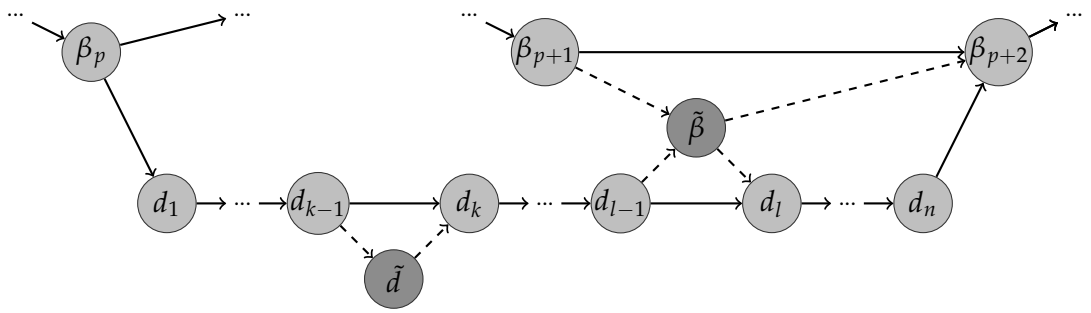


FIGURE 3.6 – *Exemple d'insertion avec séparation d'un trip dans le* 2*E–MTVRP–SS*

De plus, la concentration des flux sur un même point de transfert est favorisée par une stratégie visant à réduire la taille des voisinages explorés : pour une itération donnée de l'ALNS, chacune de ces trois heuristiques ne considère qu'un seul point de transfert dans T. Cette stratégie est décrite plus en détail en Section 3.4.2.

Insertion de satellites au premier échelon dans le 2E–MTVRP–SS

Dans le cas du 2E–MTVRP–SS, l'utilisation des transferts aux satellites est obligatoire. On retrouve néanmoins une difficulté liée à l'aspect local des insertions lorsqu'on essaie d'insérer le service d'un nouveau satellite au premier échelon. Considérons le cas d'un nouveau trip, servant un client, depuis un satellite non encore visité au premier échelon. Ici, pour servir un client, on crée un accroissement du coût de la tournée au premier niveau. Cette création a donc peu de chance d'être réalisée, on préfèrera créer de nouveaux trips partant de satellites déjà servis. Au final, les trips partent tous des satellites proches du CDC. On ne demande jamais aux véhicules du premier échelon de couvrir de longues distances alors que cela pourrait être intéressant.

Dans ce problème on distingue trois types d'insertions qui sont implémentées dans les heuristiques adaptées de meilleure insertion et de k-regret :

- *Insertion d'un client dans un trip existant*.
- *Insertion d'un client dans un nouveau trip*. Ce trip peut soit partir d'un arrêt existant au premier niveau, soit d'un arrêt à insérer dans une tournée du premier niveau.
- *Insertion avec séparation de trip*. Cette insertion d'un client c suppose : la séparation d'un trip t en deux trips t_1 et t_2 ; le choix du point de départ de t_2 (arrêt de premier niveau existant ou nouvel arrêt) ; et l'insertion de c à la meilleure position dans l'un des deux trips (Figure 3.6).

Le dernier type d'insertion permet de créer des arrêts mieux positionnés, pour des trips de plus grandes tailles. Néanmoins, pour inciter davantage à la création de nouveaux arrêts au premier échelon, on biaise l'évaluation du coût d'insertion via un nouveau satellite :

$$\textit{Coût biaisé} = \textit{coût d'insertion au second niveau} + \textit{coût d'insertion au premier niveau} \times \max(\alpha, \frac{\textit{charge du nouveau trip de second niveau}}{\textit{capacité d'un véhicule de second niveau}})$$

Avec $\alpha = 0,7$ pour l'opérateur d'insertion dans un nouveau trip et $\alpha = 0$ pour l'opérateur qui utilise la séparation des trips. Ainsi la partie du coût payé pour l'insertion dans une tournée du premier niveau est proportionnelle à la capacité utilisée et tient compte du fait que le trip nouvellement créé sera utilisé pour de futures insertions.

3.4.2 Taille des voisinages

L'insertion des points de synchronisation entre tournées dans les solutions implique une seconde complexité : dans les problèmes étudiés, la combinatoire est plus importante que dans les problèmes classiques de tournées de véhicules. Les voisinages permettant de créer des points de synchronisation sont plus importants. Cette complexité est rencontrée bien sûr dans le PDPT, le DARPT et le 2E–MTVRP–SS, mais également de manière très importante dans le FT–PDP–RS.

Heuristiques d'insertion de points de transfert

Dans le cas d'une insertion avec transfert pour le PDPT et le DARPT (Figure 3.4), la généralisation d'une heuristique de meilleure insertion pour le PDP implique, pour toute requête $r \in \mathcal{L}$, d'évaluer en plus : pour chaque position possible de p_r dans une tournée k_1 et d_r dans une tournée k_2, $k_1 \neq k_2$, l'insertion de chaque point de transfert $\tau \in T$ à toutes les positions après p_r dans k_2 et toutes les positions avant d_r dans k_2. Pour réduire la complexité de ce voisinage, on utilise une procédure moins complexe, proposée initialement par Mitrović-Minić and Laporte (2006). Elle suit le principe suivant (pour toute requête $r \in R$ et $\tau \in T$) :

1. La meilleure insertion sans transfert de la requête (p_r, d_r) est déterminée.
2. La meilleure insertion de la requête (p_r, τ) est déterminée puis cette insertion est «simulée». La meilleure insertion de la requête (τ, d_r) est ensuite déterminée.
3. La meilleure insertion de la requête (τ, d_r) est déterminée, puis «simulée». La meilleure insertion de la requête (p_r, t) est ensuite déterminée.
4. L'insertion donnant la solution de coût minimum parmi les précédentes est considérée comme la meilleure pour la requête r.

De plus, l'ensemble des points de transfert T n'est pas testé à chaque insertion. Cinq méthodes heuristiques ont été proposées pour sélectionner un point de transfert qui sera utilisé pendant une itération de l'ALNS. Les règles de sélection de ce point sont basées sur les principes suivants : aléatoirement ; en fonction de sa distance aux requêtes non insérées ; en fonction de son taux d'utilisation dans la solution courante ; ou en fonction de son historique d'utilisation dans les précédentes itérations pour les requêtes à insérer.

Chaque heuristique d'insertion avec transfert est associée à une méthode de sélection d'un sous-ensemble de points de transfert. Le couplage heuristique d'insertion, méthode de sélection constitue une heuristique pour l'ALNS. Comme signalé précédemment, cette restriction contribue également à l'utilisation de points de transferts par plusieurs requêtes.

Insertions avec création de trip dans le 2E–MTVRP–SS

Dans le 2E–MTVRP–SS, deux opérateurs permettent de créer de nouveaux trips : l'insertion dans un nouveau trip ou l'insertion par séparation de trip. Dans les deux cas, des stratégies de réduction de la taille du voisinage exploré ont montré leur efficacité.

L'insertion d'une requête avec *séparation de trip* est illustrée en Figure 3.6. Ce type d'insertion implique : l'insertion d'un client $\tilde{d}$ dans un trip du second niveau ; la séparation de ce trip en deux via un réapprovisionnement au satellite $\tilde{\beta}$; et l'insertion du service d'un satellite $\tilde{\beta}$ dans une tournée du premier niveau.

Pour complètement explorer le voisinage correspondant à cet opérateur il faut donc évaluer un nombre conséquent de combinaisons de positions d'insertion. Comme pour le PDPT, toutes les positions ne sont pas évaluées. On utilise notamment deux stratégies pour réduire le temps d'évaluation des insertions considérées :

- *Arrêts servis* : cette stratégie ne considère que des arrêts existants dans les tournées du premier niveau pour placer l'approvisionnement du second trip $\tilde{\beta}$.
- *Client d'abord* : on cherche ici en premier lieu la meilleure position d'insertion pour le client $\tilde{d}$. Une fois cette position déterminée, on cherche la meilleure position d'insertion pour $\tilde{\beta}$ dans la tournée de $\tilde{d}$ et les tournées du premier niveau.

Par ailleurs, lors de la *création d'un nouveau trip* pour servir le client à insérer, le satellite choisi sera inséré au second niveau entre le dernier client du trip précédent et le nouveau client. Pour réduire la distance parcourue au premier niveau, il peut être pertinent de ne pas choisir le satellite minimisant la somme des distances à ces deux clients. Néanmoins, on réduit la complexité en déterminant avant la résolution les s satellites les mieux placés entre chaque paire de clients et en limitant les points de départ à cet ensemble pour la création de nouveaux trips.

Dans les heuristiques *meilleure insertion* et *k-regret* adaptée pour le 2E–MTVRP–SS, on ne considère donc que les insertions dans les trips existants ; les insertions avec séparation de trip suivant les stratégies *arrêts servis* et *client d'abord* ; ainsi que les arrêts insertion par création d'un nouveau trip restreint aux s satellites les mieux placés. La mesure de regret pour une requête est basée sur des différences de coût d'insertion entre trips et non entre tournées.

Les expérimentations menées sur ce problème ont montré que sur 10 instances représentatives de 100 clients, l'utilisation des stratégies *Existing stops* et *Customer first* permet de réduire le temps calcul de plus de 50% pour des performances équivalentes. En limitant les satellites considérés pour la création de nouveaux trips aux trois mieux placés ($s = 3$), on arrive à une réduction d'environ 70% du temps de calcul.

Insertions sur des ressources

Les insertions dans le FT–PDP–RS impliquent une insertion des requêtes dans les tournées et sur les ressources. Ceci représente un nombre extrêmement important de possibilités et toutes ne peuvent être testées à chaque insertion. On utilise donc une heuristique générale qui permet de reconstruire une solution réalisable, sans pour autant évaluer toutes les possibilités d'insertion. Cette procédure est présentée dans l'Algorithme 2.

Cette procédure intègre trois types de décisions pour lesquelles différentes stratégies ont été développées :

Algorithme 2 FT–PDP–RS Heuristique d'Insertion Générale

Input une solution partielle s et la liste $\mathcal{L}$ des requêtes à insérer
Output une solution $(s, \mathcal{L})$

1: **tant que** $\mathcal{L} \neq \emptyset$ **et** qu'il existe une solution réalisable **faire**
   // Sélection de la prochaine requête à insérer
2: $r \leftarrow$ `RequestSelection`($\mathcal{L}$)
   // Insertion de r sur une tournée s
3: `RequestInsertion`(r, s)
   // Insertion de r sur les ressources impliquées
4: `ResourceScheduling`(r, s)
5: $\mathcal{L} \leftarrow \mathcal{L} - \{r\}$
6: **fin tant que**

- `RequestSelection` : Cette décision détermine la prochaine requête $r \in \mathcal{L}$ à insérer. Les stratégies de sélection sont :

  EDD (Earliest Due Date) : sélectionne la requête devant être terminée le plus tôt.

  EDRD (Earliest Delivery Release Date) : sélectionne la requête r ayant la plus petite date d'ouverture pour sa fenêtre de temps à sa destination (e_{d_r}).

  NSR (Number of Similar Request) : une demande dans le FT–PDP–RS peut être découpée en un grand nombre de requêtes identiques. Cette stratégie considère les ensembles de requêtes identiques par cardinal décroissant et sélectionne la première disponible dans $\mathcal{L}$.

  CI (Cheapest Insertion) : sélectionne la requête ayant le plus petit coût d'insertion.

  2R (2-Regret) : sélectionne la requête avec la plus forte mesure de regret de rang 2.

  CR (Critical Resource) : sélectionne une requête parmi celles devant être insérées sur la ressource qui a le plus fort taux d'occupation dans la solution courante.

- `RequestInsertion` : détermine sur quelle tournée et à quelle position dans cette tournée la requête r doit être insérée. Les stratégies proposées sont :

  CI (Cheapest Insertion) : insère la requête à la position qui accroît le moins le coût de la solution.

  ETA (Earliest Truck Appending) : insère la requête à la fin d'une tournée de telle sorte que cette requète soit servie le plus tôt possible.

- `ScheduleOnResource` : détermine la position d'insertion des sommets p_r et/ou d_r de la requête r sur les ressources concernées. Les stratégies proposées sont :

  LF (Latest Feasible) Insère le sommet à la dernière position réalisable sur la ressource.

  EF (Earliest Feasible) Insère le sommet à la première position réalisable sur la ressource.

Ceci représente un nombre important de possibilités qui est néanmoins réduit par une utilisation de combinaisons pertinentes. Par exemple : la stratégie d'insertion sur les ressources **LF** n'est utilisée qu'avec la stratégie d'insertion sur les tournées **ETA**. De même, **EF** n'est utilisée qu'avec **CI**.

On utilise en tout dix combinaisons `RequestSelection / RequestInsertion / ScheduleOnResource` : (i) **EDD/CI/EF** ; (ii) **EDD/ETA/LF** ; (iii) **EDRD/CI/EF** ; (iv) **EDRD/ETA/LF** ; (v) **NSR/CI/EF** (vi) **NSR/ETA/LF** ; (vii) **CI/CI/EF** ; (viii) **2R/CI/EF** ; (ix) **CR/CI/EF** ; et (x) **CR/CI/EF**.

3.5 Réalisabilité des Contraintes Temporelles

Une difficulté majeure en présence de contraintes de synchronisation est de construire des solutions qui respectent l'ensemble des contraintes temporelles du problème (précédences, synchronisations, fenêtres de temps). Les travaux sur le PDPT, puis sur le 2E–MTVRP–SS et le FT–PDP–RS ont permis de développer un cadre relativement général. Il permet, moyennant le calcul préalable de certaines variables pour une solution partielle réalisable s, de vérifier la réalisabilité de l'insertion d'une requête en temps constant dans s. Ce cadre est présenté en Section 3.5.1 avec son adaptation aux trois problèmes.

Cette méthode ne peut néanmoins pas être utilisée pour la résolution du DARPT, qui implique la vérification d'une durée de trajet maximum pour chaque requête. Les travaux réalisés pour la résolution du problème de réalisabilité du DARPT sont synthétisés en Section 3.5.2.

3.5.1 Un cadre général pour les problèmes sans durée de trajet maximum

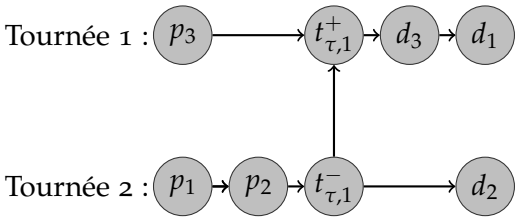
L'étude des contraintes temporelles du PDPT a permis de mettre au point une procédure efficace permettant de tester la réalisabilité d'une insertion en temps constant (Masson et al. 2013b). Ces travaux ont été étendus aux problèmes 2E–MTVRP–SS et FT–PDP–RS, ils permettent en fait de modéliser un grand nombre de problèmes de synchronisation en tournées de véhicules.

Les sections suivantes présentent comment les diverses contraintes de synchronisation sont modélisées pour les trois problèmes étudiés. Le principe général du test de réalisabilité est présenté et une évaluation de son efficacité est montrée sur le 2E–MTVRP–SS.

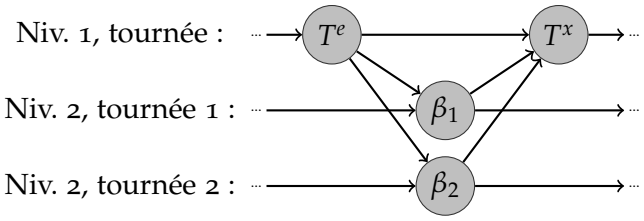
Modélisation des contraintes temporelles

Pour les problèmes PDPT, 2E–MTVRP–SS et FT–PDP–RS, les contraintes temporelles sont modélisées à l'aide d'un *graphe de précédence* comme illustré sur la Figure 3.7. Dans ces graphes on associe un poids à chaque arc, qui représente le temps minimum entre l'arrivée à son point d'origine et sa destination. Chaque sommet représente une opération (début, fin de la tournée, collecte, livraison,...) à laquelle est associée une fenêtre de temps.

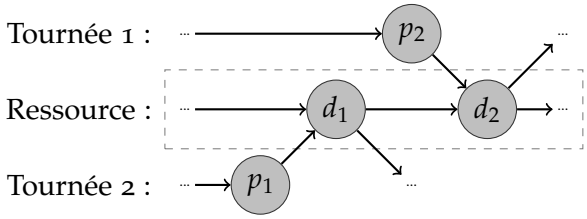
Dans le cas du PDPT, le transfert d'une requête sur un point de transfert est modélisée à l'aide de deux sommets dans le graphe temporel. Sur la Figure 3.7.a, la requête 1 est transférée de la tournée 2 vers la tournée 1 au point de transfert τ. Ceci est modélisé par le sommet $t^-_{\tau,1}$, qui représente la dépose de cette requête au point τ, et le sommet $t^+_{\tau,1}$, qui représente son chargement dans le second véhicule. L'arc $(t^-_{\tau,1}, t^+_{\tau,1})$ modélise la précédence entre le déchargement et le chargement. On lui associe un poids égal à la durée minimale du transfert.



(a) *Modélisation d'un transfert dans le PDPT*



(b) *Synchronisation dans le 2E-MTVRPTW*



(c) *Précédence sur un point de chargement dans le FT–PDP–RS*

FIGURE 3.7 – *Modélisation de la synchronisation à l'aide de graphes de précédences*

Dans le 2E–MTVRP–SS (Figure 3.7.b) on modélise la synchronisation exacte entre un véhicule du premier niveau et un ou plusieurs véhicules au second niveau. Ceci est réalisé à l'aide d'un sommet entrant T_e qui représente l'arrivée du véhicule de premier niveau au satellite et un sommet sortant T_x qui représente son départ. Le début du chargement dans les véhicules du second niveau est modélisé par un unique sommet par véhicule.

Les contraintes de précédences sur le chargement ou déchargement entre deux véhicules dans le FT–PDP–RS sont représentées en Figure 3.7.c. Dans ce cas il n'est pas nécessaire de créer de sommet supplémentaire pour modéliser la synchronisation. L'arc (d_1, d_2) est créé entre les deux opérations de déchargement successives sur la ressource. On lui associe une durée égale au temps de blocage de la ressource pour le déchargement de d_1 (typiquement la durée d'application d'un chargement d'enrobés dans le projet Orloges).

Ce graphe temporel est équivalent à un diagramme d'ordonnancement de projet, utilisé notamment dans la méthode PERT. Ce graphe est orienté et acyclique, ce qui permet le calcul d'un ordonnancement au plus tôt en temps linéaire (Cormen et al. 2001). Cet ordonnancement détermine des heures de service réalisables $h_i \in [e_i, l_i]$ pour chaque sommet i du graphe dans les solutions partielles ou complètes de l'ALNS.

Pré-processing

L'évaluation efficace de la réalisabilité d'une insertion est basée sur un pré-calcul de variables pour toute solution réalisable, pour laquelle on dispose d'un ordonnancement au plus tôt. La notion de *Forward Time Slacks (FTS)* introduite par Savelsbergh (1985) a ainsi été étendue dans Masson et al. (2013b) pour résoudre le PDPT. Ces résultats sont en fait valables pour tout graphe de précédence acyclique $\mathcal{P}$.

En premier lieu, on calcule les *Slack Times* $ST_{u,v}$ entre chaque paire de sommets du graphe de

précédence $\mathcal{P}$:

$$ST_{u,v} = \min_{w \in \Omega_{u,v}} TWT_w,$$

où $\Omega_{u,v}$ représente l'ensemble des chemins entre deux sommets u et $v \in \mathcal{P}$ et TWT_w est la somme des temps d'attente sur le chemin w. La détermination de la valeur $ST_{u,v}$ peut donc être formulée comme un problème de plus court chemin sur $\mathcal{P}$ où le poids d'un arc correspond au temps d'attente à sa destination. Si l'on note n_{prec} le nombre de sommets de $\mathcal{P}$, le calcul de l'ensemble des plus courts chemins entre toute paire de sommet du graphe peut donc être réalisé en $\mathcal{O}(n_{prec}^3)$ à l'aide de l'algorithme de Floyd-Warshall.

Au cours de cet algorithme, on enregistre la valeur booléenne $\Gamma^+(u,v)$ pour chaque paire de sommets, qui est vraie si un chemin existe entre le sommet u et le sommet v dans $\mathcal{P}$, fausse sinon.

A partir des Slack Times, on définit le FTS d'un sommet u de la manière suivante :

$$F_u = \min_{i \in \{u\} \cup \Gamma^+(u)} \{ST_{u,i} + l_i - h_i\}$$

où h_i et l_i désignent respectivement l'heure de service du sommet i dans la solution courante et son heure de service au plus tard. $\Gamma^+(u)$ représente l'ensemble des successeurs (directs et indirects) du sommet u dans le graphe de précédence $\mathcal{P}$.

Si un chemin existe dans $\mathcal{P}$ entre un sommet u et un sommet v, retarder le service au sommet u d'un temps δ implique un retard de $\max(\delta - ST_{u,v}, 0)$ au sommet v. Le FTS F_u représente le retard maximum qui peut être attribué au sommet u sans générer la violation de sa fenêtre de temps ou de celle de l'un de ses successeurs.

Vérification de la réalisabilité d'une insertion

Les variables de pré-processing permettent d'évaluer le coût et la réalisabilité d'une insertion sans modifier la solution courante. Cette évaluation est basée sur deux tests : en premier lieu on vérifie que l'insertion en question ne crée pas de cycle dans le graphe de précédence. En second lieu, on détermine si la solution résultant de l'insertion est réalisable pour les contraintes temporelles du problème.

Détection d'une création de cycle Comme illustré sur la Figure 3.8, en présence de contraintes de synchronisation, certaines insertions dans une solution réalisable peuvent introduire un cycle dans le graphe de précédence.

Pour détecter efficacement de telles insertions, on suit la méthodologie suivante : si une insertion implique une précédence $u \rightarrow v$ entre deux sommets u et v de la solution courante, on vérifie que la précédence inverse n'existe pas dans la solution actuelle. Si $\Gamma^+(v,u)$ a la valeur *false* pour chaque précédence créée alors l'insertion ne crée pas de cycle.

Pour chaque type d'insertion, on identifie donc un ensemble minimal de précédences ajoutées dans la solution courante pour lesquelles on réalisera ce test.

Pour le PDPT, l'insertion la plus complexe (insertion avec transfert, Figure 3.4) est illustrée en Figure 3.9 sur le graphe temporel. L'insertion crée une précédence entre le sommet j et le sommet $\sigma(i')$. On vérifie donc $\Gamma^+(\sigma(i'), j) = \textit{false}$.

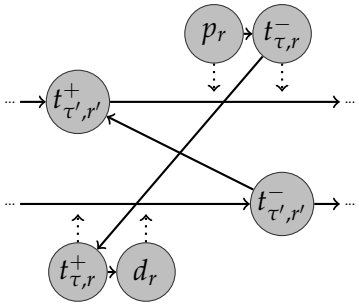


FIGURE 3.8 – *Exemple d'insertion qui produit un cycle dans le PDPT*

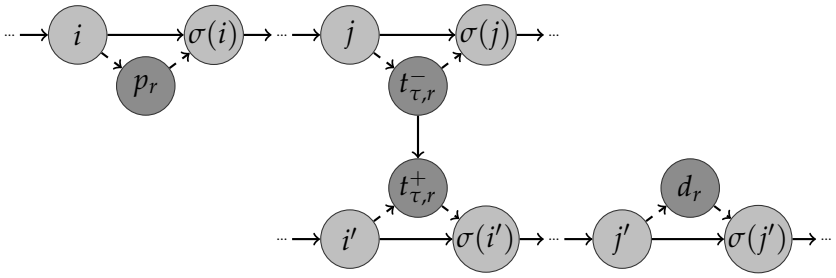


FIGURE 3.9 – *Exemple d'insertion dans le graphe temporel d'une requête r via un point de transfert τ.*

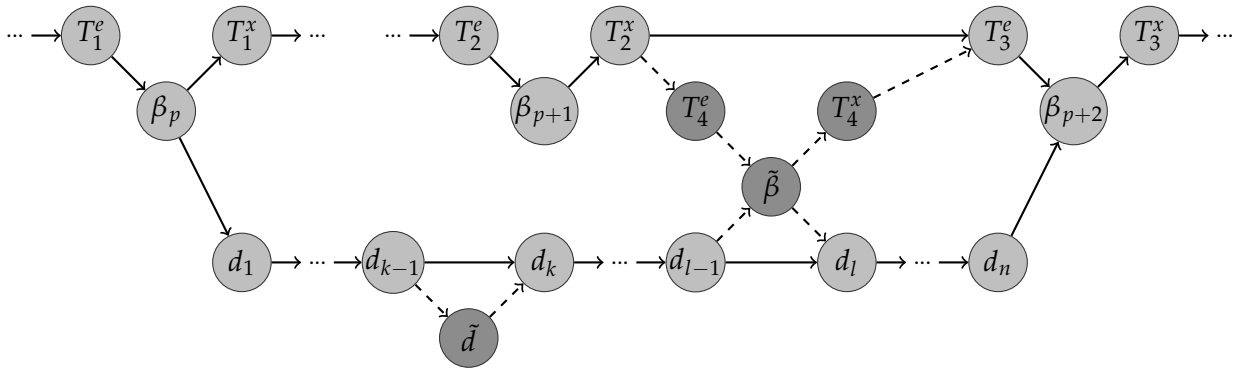


FIGURE 3.10 – *Exemple d'insertion sur le graphe temporel d'un client $\tilde{d}$ avec séparation d'un trip dans le 2E–MTVRP–SS. Le nouveau trip est alimenté au satellite $\tilde{\beta}$, les arcs créés par cette insertion sont représentés en pointillés.*

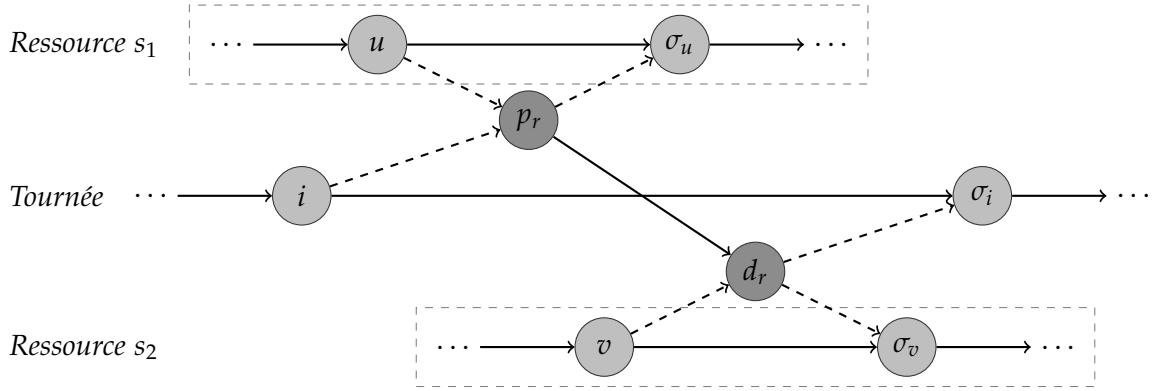


FIGURE 3.11 – *Exemple d'insertion d'une requête (p_r, d_r) dans le FT–PDP–RS*

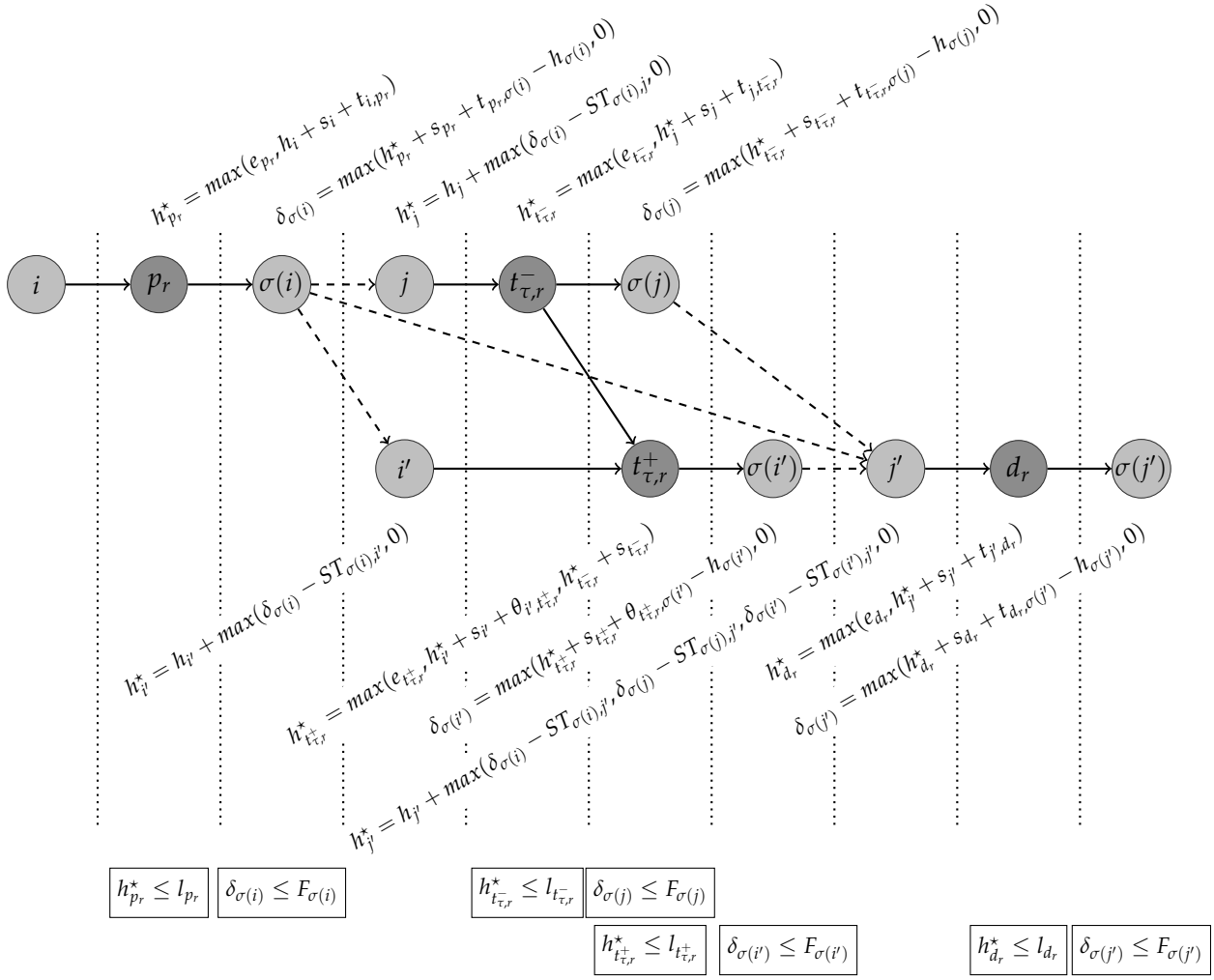


FIGURE 3.12 – *Graphe restreint pour l'estimation des temps de service dans le cadre de l'insertion dans le PDPT présentée en Figure 3.9 (la requête r est insérée via le point de transfert τ)*

Dans le cas du 2E–MTVRP–SS, l'insertion d'un client avec séparation de trip, présentée en Figure 3.6 se modélise dans le graphe temporel comme présenté en Figure 3.10. Cette insertion crée une précédence entre les sommets d_{k-1} et T^x_3. On vérifie donc $\Gamma^+(T^x_3, d_{k-1}) = \mathit{false}$.

Une insertion sur deux ressources synchronisées dans le FT–PDP–RS est illustrée en Figure 3.11. Cette insertion crée les précédences $u \to \sigma_i$, $u \to \sigma_v$, $i \to \sigma_u$, $i \to \sigma_v$ et $v \to \sigma_i$, pour lesquelles on vérifiera qu'il n'existe pas de chemin inverse dans $\mathcal{P}$.

Vérification des contraintes et ordonnancement des tournées En second lieu, on détermine une procédure qui définit l'ordre de calcul des temps de service et de vérification des fenêtres de temps à partir des temps de parcours et des variables de pré-processing.

L'idée générale de ce test est de déterminer un graphe de précédence restreint qu'on décomposera en niveaux pour recalculer les temps de service dans le bon ordre. Pour le PDPT, ce graphe est illustré en Figure 3.12.

Pour un sommet i de ce graphe on note $h^\star_i$ son heure de service dans la solution évaluée.

Pour les sommets déjà insérés, on note δ_i le délai subi par un sommet i en raison de l'insertion évaluée. L'heure de passage au plus tôt, ou le décalage à chaque sommet dans le graphe restreint est calculé par ordre croissant de niveau (le calcul réalisé est présenté a proximité de chaque sommet). Les conditions de réalisabilité vérifiées au cours de ce calcul sont spécifiées sous le graphe.

Les arcs en traits pleins représentent des arcs du graphe temporel. Les arcs en pointillés représentent un chemin dans ce graphe. L'impact d'un délai au début d'un tel chemin est évalué à l'aide des variables ST : par exemple, l'insertion du sommet p_r provoque un retard $\delta_{\sigma(i)}$ sur son successeur. Il doit être inférieur à $F_{\sigma(i)}$. Ce retard peut être : complètement propagé suivant le graphe de précédence aux sommets i', j ou j' ; ou bien absorbé, complètement ou partiellement, par des temps d'attente sur les chemins entre i et ces sommets. Pour le sommet i', la nouvelle heure de passage est $h^\star_{i'} = h_{i'} + max(\delta_{\sigma(i)} - ST_{\sigma(i),i'}, 0)$.

Les variables ST et F, calculées lors du pré-processing, permettent donc d'évaluer en temps constant l'intégralité d'une insertion avec transfert sans réordonnancer tous les successeurs des sommets ayant subis un délai.

Les graphes restreints pour le 2E–MTVRP–SS et le FT–PDP–RS sont donnés respectivement en Figures 3.13 et 3.14. Dans le cas du FT–PDP–RS, les variables ST ne sont pas nécessaires.

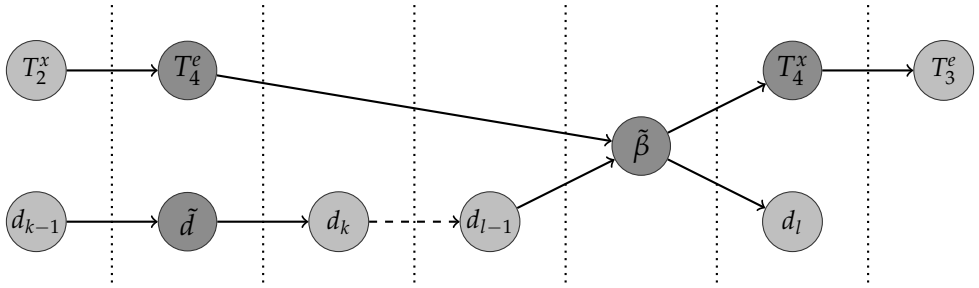


FIGURE 3.13 – *Graphe restreint du 2E–MTVRP–SS correspondant à l'insertion de la Figure 3.10*

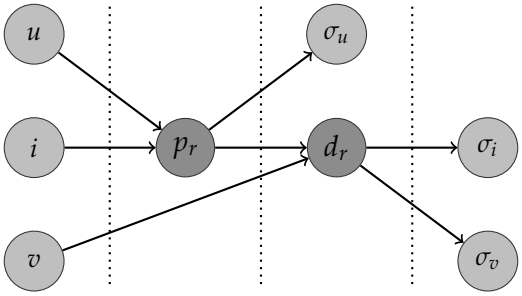


FIGURE 3.14 – *Graphe restreint du FT–PDP–RS modélisant l'insertion présenté en Figure 3.11*

Impact sur les temps de calcul

L'impact du test de réalisabilité en temps constant a été évalué pour le 2E–MTVRP–SS, en le comparant à une implémentation incrémentale d'un algorithme d'ordonnancement au plus tôt dans le graphe temporel. Une synthèse des expérimentations sur 12 instances à 100 clients

est présentée dans le Tableau 3.1. Sur 25 ooo itérations de l'ALNS, les versions implémentant le test de réalisabilité en temps constant sont en moyenne 12 fois plus rapides que les versions implémentant l'algorithme en temps linéaire. Ceci est constaté en dépit des optimisations réalisées pour ne propager que les modifications dues aux insertions dans l'ordonnancement des tournées.

Instance	c101	c102	c201	c202	r101	r102	r201	r202	rc101	rc102	rc201	rc202	average
PERT (in min)	772.4	696.5	1043.2	933.1	483.5	244.0	292.8	543.1	598.4	804.3	456.0	1612.6	706.7
FTS (in min)	**55.5**	**74.1**	**67.6**	**80.7**	**34.7**	**35.7**	**35.4**	**54.6**	**38.4**	**54.9**	**43.1**	**85.7**	**55.0**
Difference (in %)	-92.8	-89.4	-93.5	-90.9	-92.8	-85.4	-87.9	-89.9	-93.6	-93.2	-90.5	-94.7	-92.2

TABLE 3.1 – *Temps de calcul du PERT incrémental versus FTS pour 25 ooo itérations de l'algorithme ALNS pour le 2E–MTVRP–SS*

3.5.2 Prise en compte d'une durée de trajet maximum pour le DARPT

Dans le cas du DARPT, une contrainte de temps de trajet maximum est associée à chaque requête. L'utilisation des précédents algorithmes ne permet alors pas d'évaluer en temps constant les contraintes temporelles dans une solution.

Ce problème de satisfaction est modélisé dans Masson et al. (2012) et Masson et al. (2014a) comme un problème temporel simple (STP : Simple Temporal Problem (Dechter et al. 1991)). Pour le résoudre on modélise le problème à l'aide d'un *graphe de distance*. La Figure 3.15 illustre un exemple de graphe de distance pour un problème à deux tournées, trois requêtes et un point de transfert. La modélisation est relativement proche de celle du graphe temporel du PDPT à l'exception des fenêtres de temps et des contraintes de temps de trajet maximum.

Le sommet 0 est un sommet virtuel qui représente l'heure 0. Les autres sommets sont les points servis par les tournées, avec une duplication du point de transfert identique à celle réalisée pour le PDPT. Dans ce graphe, une fenêtre de temps $[e_i, l_i]$ sur un sommet i est représentée par un arc de 0 vers i de poids e_i et un arc de i vers 0 de poids $-l_i$ (ces arcs, en traits pleins sur la figure, n'ont pas été tous représentés pour des raisons de lisibilité). Les temps de trajets et temps de service sont représentés en arcs pointillés longs. Les arcs en pointillés courts du sommet d_r vers le sommet p_r, pour toute requête r, se voient attribuer un poids $-\bar{L}_r$ égal à moins le temps de trajet maximum autorisé pour cette requête.

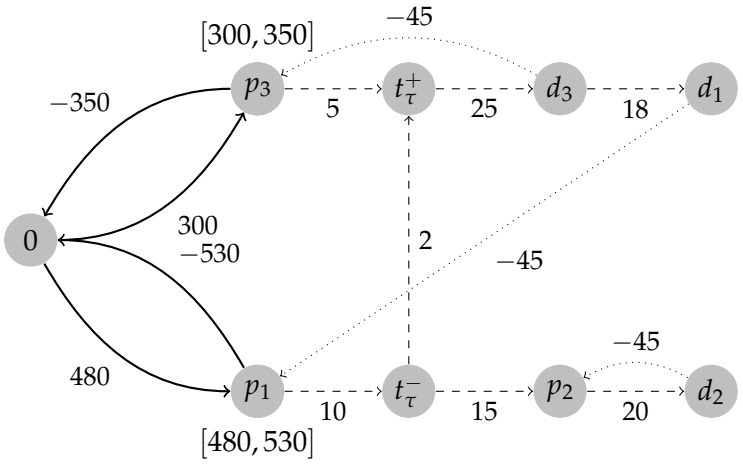


FIGURE 3.15 – *Graphe de distance du problème STP associé à la réalisabilité d'une solution dans le DARPT*

Un cycle de coût positif dans ce graphe, tel qu'illustré en Figure 3.16, indique un ensemble de contraintes non réalisables entre elles. Pour détecter de tels cycles on utilise l'algorithme de plus long chemins BFCT (Tarjan 1981) qui est une variation de l'algorithme de Bellmann-Ford. Pour un graphe à n sommets et m arcs, sa complexité est $\mathcal{O}(n.m)$.

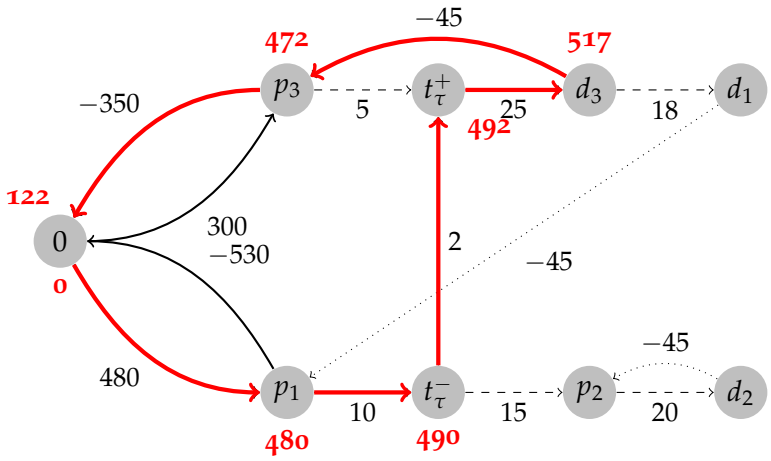


FIGURE 3.16 – *Cycle de longueur positive dans le STP : le cycle* $0 - p_1 - t_\tau^- - t_\tau^+ - d_3 - p_3 - 0$ *à une longueur de* 122 *qui représente une non réalisabilité de la solution. En effet, si le sommet* d_3 *ne peut pas être servi avant l'heure* 517*, il est impossible de servir* p_3 *avant son heure au plus tard* $l_{p_3} = 350$ *tout en assurant un temps de trajet inférieur à* $\bar{L}_3 = 45$ *minutes.*

L'impact de l'algorithme de réalisabilité en $\mathcal{O}(n.m)$ sur le temps de résolution étant relativement important, deux Conditions Nécéssaires (CN) et deux Conditions Suffisantes (CS) sont testées avant la résolution du STP. Une CS vérifiée garantie la réalisabilité de la solution, une CN fausse implique une solution non réalisable. Ces conditions sont évaluées dans l'ordre suivant :

- CN_1 – *relaxation comme un PDPT* : Dans un premier temps, les fenêtres de temps des sommets de la solution courante sont contractées à l'aide des résultats du STP pour cette solution. On utilise ensuite le test de réalisabilité en temps constant du PDPT pour rejeter les insertions non réalisables au sens de ces fenêtres de temps (ceci correspond à une forme de relaxation des contraintes de temps de trajet maximum).

- CS_1 – *insertion sans impact* : Lorsqu'une insertion ne modifie pas les heures de passage dans la solution courante et que les fenêtres de temps des sommets insérés sont respectées, l'insertion est réalisable. Ce test peut être facilement intégré à la vérification de la CN_1.

- CN_2 – *relaxation DARP* : Chaque requête transférée est transformée en deux requêtes indépendantes, avec un calcul d'un temps de trajet maximum pour chaque partie. Le calcul des heures de passage à chaque sommet est ensuite réalisé comme pour un DARP sans transfert avec l'algorithme de Cordeau and Laporte (2003). Une solution non réalisable pour ce problème est rejetée.

- CS_2 – *solution DARP réalisable* : Dans le cas où les heures de passage déterminées dans la CN_2 sont réalisables pour le DARP formulé à cette étape, on vérifie si elles le sont également pour le DARPT. Si c'est le cas, la solution est acceptée.

On montre que l'apport de ces différentes conditions est variable. Pour des instances de l'ADAPEI 44, avec de longues distances, peu de véhicules et de fortes contraintes de temps, les conditions nécessaires permettent de diviser la durée de l'ALNS par un peu plus de 10. Les conditions suffisantes permettent de gagner quelques minutes supplémentaires. Sur des instances issues du conseil général, avec plus de destinations et des limites de temps de trajet moins contraignantes, l'apport des conditions nécessaires et suffisantes est très réduit.

En dépit de ces avancées, le DARPT reste difficile à résoudre, avec des temps de calcul de plus de 6h30 pour 25 000 itérations de l'ALNS sur un problème à 109 requêtes et 2 points de transferts, contre 1h30 pour le PDPT.

3.6 Autres Contributions en Tournées avec Synchronisation

En dehors du cadre de résolution présenté précédemment, j'ai eu l'occasion de travailler sur d'autres problèmes de tournées de véhicules faisant intervenir de la synchronisation. En premier lieu, dans le cadre de la thèse de Renaud Masson sur le PDPT et le DARPT, ont été réalisés deux travaux :

- Une première approche de résolution exacte du problème de transport de personnes avec transferts, résumée en Section 3.6.1. Ici la synchronisation est prise en compte dans une méthode de Branch & Cut & Price grâce à une simplification du problème, inspirée de la pratique des transferts sur le terrain.
- Une extension de l'ALNS pour le PDPT à un cas d'étude en logistique urbaine. Ce cas est résumé en Section 3.6.2. Cette étude a été réalisée en collaboration avec des collègues de La Rochelle sur un système de livraison en ville via une ligne de bus et des tournées de tricycles électriques.

Enfin, une forme intéressante de synchronisation est étudiée dans le cadre de la conception de tournées de véhicules régulières, présenté en Section 3.6.3. Il s'agit ici de proposer des horaires constants à des clients présents sur plusieurs journées de transport alors que la présence des autres varie. On peut voir cette application comme une forme de synchronisation entre tournées se déroulant sur différentes périodes.

3.6.1 Une approche exacte pour le problème de tournées de véhicules avec navettes

La résolution exacte de problèmes d'optimisation de tournées de véhicules avec synchronisation est un sujet extrèmement difficile et complexe. Dans le cas du PDPT, l'algorithme de Branch & Cut proposé par Cortés et al. (2010) résout des problèmes à 6 requêtes. Plus récemment, Rais et al. (2014) présentent un modèle linéraire en nombre entiers permettant de résoudre des problèmes à 7 requêtes. Les temps de résolution augmentent exponentiellement avec le nombre de requêtes et aucune approche ne permet pour l'instant de résoudre des problèmes réels optimalement.

Un algorithme efficace pour résoudre exactement les problèmes de tournées de véhicules, notamment en présence de fenêtres de temps, est l'algorithme de Branch & Price. Cette méthode repose généralement sur un programme maître qui sélectionne des tournées dans un sous-ensemble de l'ensemble des tournées réalisables et sur un sous-problème qui génère des

tournées. Néanmoins, l'utilisation de cette approche est rendue difficile en présence de synchronisation car elle suppose : soit la connaissance des tournées lorsqu'on en génère une autre ; soit la détermination des heures de service dans le problème maître.

Dans le but de mettre en place un algorithme de génération de colonnes pour le calcul de bornes inférieures du problème dans un algorithme de branch & price, nous avons proposé une simplification du PDPT intitulé le problème de tournées de collectes et livraisons avec navettes (PDPS : Pickup and Delivery Problem with Shuttle). Cette simplification du problème est issu de la pratique réelle des établissements de l'ADAPEI 44 qui ont inspiré les travaux sur le PDPT. La Figure 3.17 illustre une solution du PDPS avec trois points de destination (les établissements).

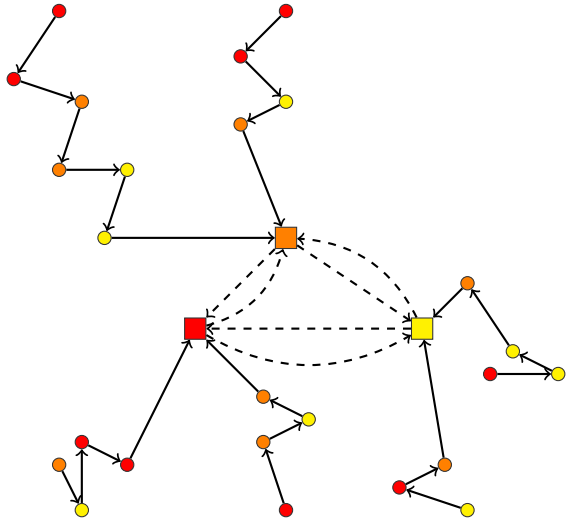


FIGURE 3.17 – *Représentation d'une solution du PDPS : les ronds représentent les domiciles des passagers, les carrés les établissements. Les traits en pointillés représentent les navettes.*

Dans le PDPS on distingue deux types de trajet pour chaque véhicule : les *trajets de collecte* (en traits pleins sur la figure), qui permettent de transporter les personnes jusqu'à un établissement, indépendamment de leur destination ; et les *navettes* entre établissements (en pointillés), qui permettent de transporter les personnes rassemblées par plusieurs véhicules à un établissement intermédiaire vers leur destination finale.

Ici la synchronisation entre tournées peut-être exprimée sous forme de fenêtres de temps, ce qui rend possible la génération de colonnes. Deux modèles ont été proposés et différentes coupes ont été intégrées à l'algorithme de résolution. Des instances générées et réelles comportant jusqu'à 87 requêtes ont été résolue à l'optimalité. Les résultats obtenus confirment les gains évalués par les heuristiques sur l'apport des transferts (PDPS vs PDPTW), en dépit de la simplification introduite. On montre de plus un gain pouvant aller jusqu'à 22 % entre la résolution du PDPS et la résolution d'un VRP par établissement.

3.6.2 Synchronisation de tournées avec une ligne de bus régulière en logistique urbaine

La conception d'un système de transport mixte passagers–marchandises pour la distribution urbaine sur les « *derniers kilomètres* » a fait l'objet de la thèse d'Anna Trentini en Sciences de gestion (Trentini 2012) :

> « Le système de transport proposé assure la distribution de marchandises à partir

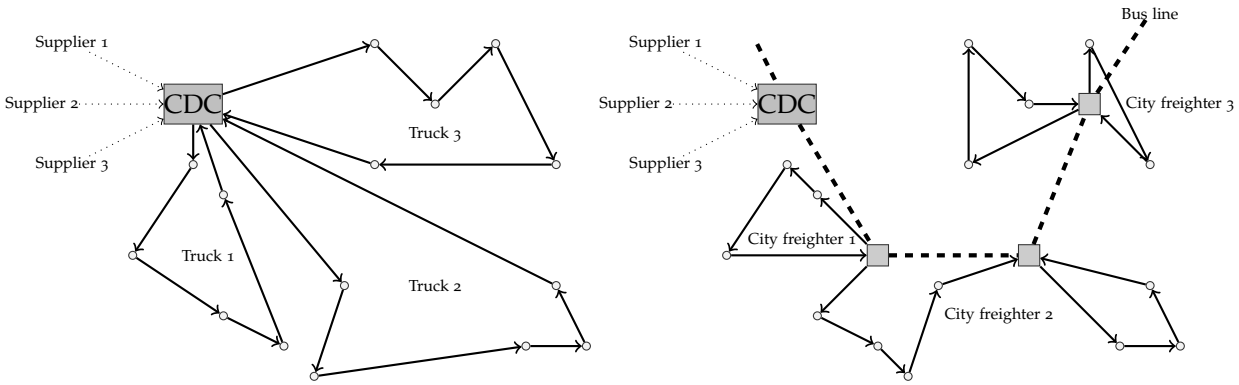


FIGURE 3.18 – *Approche classique vs deux–niveaux pour la distribution en centre ville via une ligne de bus*

> d'un centre de distribution urbaine, en s'appuyant sur une ligne de transport en commun. Les véhicules circulant sur la ligne utilisent leur capacité résiduelle pour transporter les marchandises. Les marchandises sont déchargées aux arrêts de la ligne. Un système de distribution capillaire associé à chaque arrêt assure la livraison aux destinataires finaux par des tournées.

Cette thèse a été réalisée dans le cadre du projet ANR C-GOODS, achevé en 2011. Le système proposé par Anna Trentini est mis en regard du mode standard de distribution en Figure 3.18. Le système repose sur des containers roulants qui peuvent être aisément transférés des bus à des tricycles électriques aux arrêts. Pour chaque bus et chaque trajet, un nombre maximum de containers est estimé en fonction d'une estimation de la capacité disponible. Toujours dans l'optique de préserver la qualité du service aux passagers, le nombre de containers déchargés à chaque arrêt ne peut dépasser une limite donné.

La conception des tournées de distribution dans ce système a fait l'objet d'une collaboration avec Anna Trentini, Nicolas Malhéné et Houda Tlahig dans le cadre de la thèse de Renaud Masson (toujours co-encadré avec Olivier Péton). L'objectif était en priorité d'évaluer le nombre de véhicules nécessaire pour faire fonctionner le système. Le problème de transport urbain mixte (MUTP : Mixed Urban Transportation Problem) a été défini. Il est décomposé en un problème maître et un sous-problème. Le problème maître, résolu par un ALNS, détermine : la répartition des commandes dans les containers ; l'arrêt où est déchargé chaque container ; l'affectation des containers aux véhicules ; l'ordre des clients dans une tournée et le séquencement des tournées de chaque véhicule. Pour chaque solution évaluée, le sous-problème RCAP (Rolling Container Assignment Problem) trouve une affectation réalisable des containers aux bus en intégrant les contraintes de capacité et les contraintes temporelles du problème. Ce problème est formalisé et résolu de manière heuristique.

Ce problème est résolu sur 18 instances issues d'une enquête de terrain qui a permis d'estimer la demande des commerces du centre ville de La Rochelle. L'heuristique est comparée à un calcul d'une borne inférieure du nombre de véhicules réalisé par génération de colonnes. On montre que le nombre minimum de véhicules est trouvé pour 10 instances sur 18. Pour les 8 restantes, la différence entre la borne inférieure et la meilleure solution de l'heuristique est de 1 véhicule.

L'ensemble de ces travaux ont été présentés à la conférence ILS (Trentini et al. 2012), publiés dans les actes de la conférence ainsi que dans la revue EJTL (Masson et al. 2015).

3.6.3 Conception de tournées de véhicules régulières pour le transport hebdomadaire de personnes handicapées

Dans la suite du logiciel Marika, le projet CTouVer (2008-2009) du GdR RO a été réalisé en collaboration avec Dominique Feillet, Thierry Garaix, Olivier Péton et Dominique Quadri. Il porte sur un critère de qualité de service évoqué dans le cadre de Marika : la régularité. En effet, lorsque les mêmes clients sont servis de manière répétitive au cours du temps, maintenir une certaine régularité dans les horaires de service peut être grandement apprécié. Cette préoccupation est particulièrement pertinente pour le transport de personnes sensibles aux changements.

Nous nous sommes intéressés à un problème de VRP où les client émettent une demande intermittente sur une période d'une semaine. Certains clients doivent être transportés tous les jours tandis que d'autres ne doivent être transportés que certains jours de la semaine. Malgré ces variations de la demande, la problématique consiste à maintenir des horaires de service réguliers.

Les contributions ont porté sur une modélisation nouvelle de la régularité en tournées de véhicules et sur la conception d'un algorithme de résolution approché, qui intègre un branch & price dans une heuristique de recherche à voisinage large. Ces travaux ont été publiés dans la revue Networks (Feillet et al. 2014).

Modélisation de la régularité

Un article récent de Groër et al. (2009) définit la notion de *consistent VRP* pour un problème de distribution de courrier. Dans leur modèle mathématique, la notion de régularité des horaires se traduit par une contrainte fixant une différence maximum entre l'horaire maximum et l'horaire minimum de livraison de chaque client sur la période considérée. Ils imposent en outre une règle de précédence qui fige l'ordre de service aux clients.

Dans l'ordonnancement des tournées, des variations de l'heure de service de l'ordre de la minute ne sont pas ressenties par le client. Il fait d'ailleurs peu de sens de les prendre en compte sans considérer qu'il existe une imprécision dans les données et un aléa dans les temps de transport. Des horaires proches peuvent donc être considérés comme appartenant à une même *classe horaire*. Dans le problème considéré, la *régularité* du service est liée au nombre de classes horaires par client, plutôt qu'à une mesure continue des différences entre les horaires de service (Figure 3.19).

Dans ce projet nous avons proposé un modèle de VRP recherchant un ensemble de tournées de longueur minimale, en limitant le nombre de classes acceptables pour tout client. Le modèle est voisin de celui présenté dans Groër et al. (2009). Les principales différences concernent la définition de la régularité, l'interdiction d'introduire des temps d'attente avec des clients à bord des véhicules, et l'abandon de la règle de précédence.

Heuristique et résolution du sous-problème

L'heuristique utilise un voisinage sur les journées qui consiste, à chaque itération, à détruire puis reconstruire l'ensemble des tournées d'une journée. Lors de la reconstruction, on cherche à contrôler le nombre de classes de chaque client. Considérons un client i ayant c_i classes. La reconstruction des tournées de la journée permet d'obtenir entre $c_i - 1$ et $c_i + 1$ classes. Ne pas augmenter le nombre de classes nécessite de servir i dans une fenêtre horaire compatible avec les heures de service des autres journées. Le sous-problème lié à la reconstruction est donc un

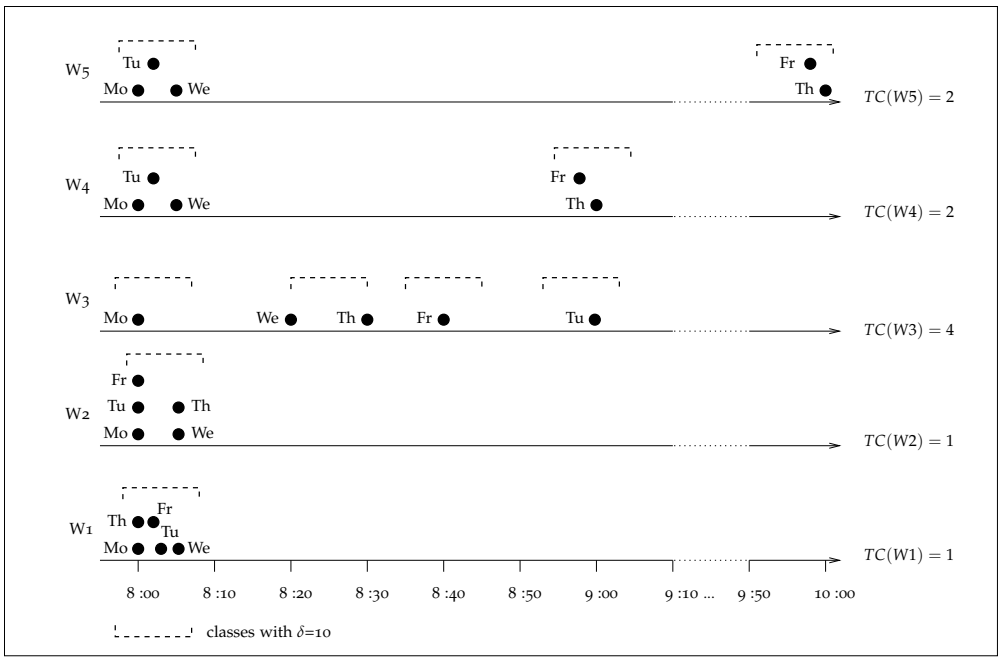


FIGURE 3.19 – *Définition des classes dans le VRP avec régularité*

VRP à fenêtres horaires multiples, et interdisant l'attente des véhicules en cours de tournée. Ce problème, noté VRPmTW-nw, est résolu par une méthode de type branch & price.

À chaque itération, l'heuristique sélectionne une journée, puis un client parmi ceux ayant le plus grand nombre de classes, et définit le VRPmTW-nw visant à réduire le nombre de classes du client sélectionné. La solution fournie du VRPmTW-nw peut augmenter le nombre de classes des clients non sélectionnés. Cette augmentation peut avoir une influence négative sur la régularité de la solution. Une dégradation temporaire peut être acceptée afin de diversifier la recherche.

L'heuristique construit un front de Pareto en faisant varier le nombre k de classes acceptables entre 1 et le nombre de périodes de l'horizon de planification. Pour chaque valeur de k, la solution de distance minimale est sauvegardée.

Expérimentations numériques

L'heuristique proposée a été évaluée sur trois familles d'instances : des petites instances définies par Groër et al. (2009), les instances de Christofides and Eilon (1969) adaptées de manière à introduire une demande intermittente, et des données réelles concernant le transport de personnes handicapées mentales.

Un résultat très intéressant est qu'on montre qu'il est possible, pour différents profils de demande, d'obtenir des solutions très régulières sans pour autant augmenter le coût des tournées de manière excessive. Si l'on accepte deux classes d'horaires, les coûts ne sont augmentés que de 1% en moyenne par rapport à une solution non régulière. Pour avoir une seule classe, l'augmentation moyenne de coût est de 5,9%.

Conclusion et perspectives

Une remarque est que, même si nous n'en avions pas conscience à l'époque, la régularité induit une synchronisation entre tournées de journées différentes. Les perspectives de ce travail incluent la prise en compte de contraintes plus réalistes dans le modèle pour se rapprocher de l'application initiale. La notion de régularité est de plus présente dans d'autres applications. Un problème extrèmement proche a été étudié dans le cadre de Remy Spliet pour la résolution de problèmes de tournées avec demande stochastique (Spliet and Desaulnier 2012, Spliet and Gabor 2015). Le problème est d'affecter une unique fenêtre de temps pour un horizon donnée à des clients, sachant que la demande subit des variations. Ce problème est fréquemment rencontré par les entreprises de transport et encore peu étudié sous cet angle. Un premier état de l'art sur les tournées avec régularité, faisant état de travaux pour la plupart récents, vient juste d'être publié (Kovacs et al. 2014).

La méthode développée pourrait également être étendue pour résoudre des problèmes où la régularité n'est pas souhaitée. C'est le cas dans le cas de transport d'objets de valeur ou d'argent, où l'on souhaite construire des tournées différentes chaque jour (Michallet et al. 2014).

3.7 Conclusion, perspectives de recherche

Les résultats obtenus et suite des travaux sur le PDPT sont encourageants et appellent à de nouveau développements pour généraliser encore le cadre proposé et l'évaluer sur de nouvelles applications. Des suites sont en cours, avec Philippe Grangier par exemple, sur l'optimisation de tournées de transport avec cross-dock, ou Axel Grimault, dont les algorithmes sont actuellement utilisés dans une étude sur l'optimisation de véhicules auto-guidés dans un entrepôt chez Système U.

Néanmoins, comme montré dans le cas du DARPT, le cadre ne répond pas à tous les besoins et de nouveaux développements méthodologiques sont à réaliser. La prise en compte d'un critère de minimisation de la durée des tournées est un premier verrou. Dans le cas du VRPTW, Savelsbergh (1992) présente une évaluation en temps constant du coût d'un voisin dans une recherche locale. En présence de contraintes de synchronisation il n'existe pas, à notre connaissance, d'équivalent. Pour le FT–PDP–RS en particulier, le coût d'une solution opérationnelle dans le projet ORLOGES dépend de la durée des routes. Néanmoins, ce critère correspond à un critère «non régulier» en présence de contraintes de synchronisation. En ordonnancement, un critère est dit régulier s'il est une fonction décroissante des dates de fin des jobs du problème. En d'autres termes, il est toujours optimum de débuter et terminer les tâches aux plus tôt Conway et al. (1967). En effet, une fois les tournées fixées, minimiser la durée des tournées revient à minimiser les temps d'attente, critère peu étudié en ordonnancement. Ce problème peut être posé comme un programme linéaire, mais cette approche convient mal à l'évaluation de multiples insertions. Des travaux sont donc en cours afin de proposer des solutions à ce problème et d'en évaluer la complexité.

Par ailleurs, la présence de contraintes de synchronisation, en particulier dans le cas du DARPT ou du 2E–MTVRP–SS, pose la question de la robustesse des solutions proposées. L'impact des aléas sur les temps de transport, temps de service, est bien connu en transport aérien, et dans les problèmes de construction de rotations d'équipages en particuliers. Un seul retard peut avoir des conséquences fâcheuses sur l'ensemble du réseau. L'intégration de temps de trajet

ou temps de service stochastiques présente un intérêt certain pour la construction de solution moins sensibles aux aléas.

L'étude de la synchronisation avec une ligne de transport régulière pour le transport mixte passagers marchandises ouvre également des perspectives intéressantes. Ce système, étudié en logistique urbaine, est encore plus attractif dans les zones à faibles densité de population où les derniers kilomètres pour la marchandise et les transports publics coûtent très cher. C'est le sujet initié dans le cadre du projet TeMix beta dont la seconde phase est en cours de montage.

Pour terminer, nous avons maintenant acquis une certaine expérience sur l'algorithme (A)LNS, qui, en dépit de sa performance, reste relativement jeune et perfectible. On constate en effet des axes de travail, notamment sur des aspects robustesse (capacité à toujours trouver une bonne solution) et pour la résolution de problèmes de grande taille. L'algorithme comporte un grand nombre de fonctions et paramètres et les différents papiers qui ont suivi les travaux de Ropke and Pisinger (2006a) montrent qu'il n'est pas aisé de déterminer ceux qui sont réellement utiles. Une synthèse et une analyse des différentes contributions réalisées avec cette métaheuristique est à réaliser pour identifier les composants clés et les points d'améliorations à apporter à la méthode.

Chapitre 4

Conception de Réseaux Logistiques Mutualisés

La conception de réseaux de transport est un problème largement étudié dans la littérature. L'objectif est d'optimiser le réseau de distribution de manière à pouvoir satisfaire au mieux un ensemble de demandes de transport. Dans ce type de problème, les principales décisions portent sur la localisation de plateformes logistiques ; sur la détermination des moyens (véhicules) à affecter entre sites de production, stockage, plateformes et clients ; et sur la détermination des chemins suivis par les commandes dans le réseau.

Les problèmes de conception de réseaux de transport sont souvent abordés pour des entreprises qui sont propriétaires de leurs plateformes et de leur flotte de véhicules. Je présente ici une synthèse de travaux récents ou en cours qui abordent plusieurs problèmes d'industriels avec, à chaque fois, deux caractéristiques relativement originales. En premier lieu le transport est sous-traité, ce qui induit des modèles de coûts et des contraintes différents de ceux rencontrés dans la littérature. Deuxièmement, nos travaux interviennent dans un contexte collaboratif : un ensemble de petites et moyennes entreprises se regroupent pour consolider leurs flux et réduire ainsi les coûts de transport.

4.1 Introduction

La planification de réseaux logistiques fait intervenir un grand nombre de décisions Crainic and Laporte (1997). La question des tournées de véhicules, couverte dans le précédent chapitre, appartient aux décisions opérationnelles et concernent généralement des transports sur de courtes distances. Sur le plan stratégique, se pose par ailleurs la question de la localisation de plateformes logistiques (entrepôts, cross-dock). Celles-ci déterminent les points de consolidation sur le réseaux Klose and Drexl (2005). Ces plateformes permettent en général de faire le lien entre transports longue distance et distribution régionale.

Concernant le transport longue distance, sur un plan tactique et opérationnel, il faut déterminer les moyens de transport permettant de faire transiter les marchandises sur le réseau, leurs fréquences et horaires, de manière à permettre l'acheminement des demandes depuis leurs origines jusqu'à leurs destinations. Ce domaine est identifié sous la dénomination *design de réseaux de services* et est plus largement présenté dans l'article de synthèse de Crainic (2000). Une attention croissante est portée ces dernières années sur ce type de problème, avec de nouvelles contributions permettant de mieux en mieux intégrer les contraintes horaires et trajets à vide des véhicules (Erera et al. 2013a;b, Crainic et al. 2014). Ces problématiques apparaissent de plus lorsque plusieurs modes sont utilisés pour transporter les marchandises (Moccia et al. 2011, SteadieSeifi et al. 2014).

Ces problématiques sont été étudiées à travers trois projets : le projet PREDIT Logistique Mutualisée Durable (LMD), le FUI VEGESUPPLY qui suit une pré-étude, et un contrat industriel avec la société 4S Network, qui supporte la thèse de Juliette Médina. Les problèmes étudiés concernent plusieurs producteurs ou fournisseurs désirant collaborer pour diminuer leurs coûts de transport. Dans ces différents cas, les transports sont confiés à des prestataires logistiques pour distribuer des clients en France. Ce contexte a pour conséquence différentes originalités portant sur les coûts de transport, les aspects collaboratifs et la construction de tournées.

4.1.1 Modes et coûts du transport externalisé

On distingue deux types de prestataires qui structurent et font fonctionner le réseau de distribution : les transporteurs, qui réalisent le transport longue distance ; et les distributeurs régionaux, qui possèdent une plateforme de cross-dock et réalisent des tournées de véhicules pour distribuer des clients proches.

Transporteurs

Dans les applications rencontrées, on distingue deux types de transport longue distance qui ont des tarifs différents :

- Le *transport en camions complets* (FTL : Full Truck Load), où l'on achète l'ensemble de la capacité d'un véhicule entre un point de départ et un point d'arrivée. Un coût indépendant de la quantité transportée est négocié avec le transporteur. Celui-ci dépend de multiples facteurs (distance, durée, mais aussi possibilité de trouver un flux retour).
- Le *transport en camion incomplet* (LTL : Less-than Truck Load), où le tarif de la prestation est donné par une matrice de transport. Le coût dépend du département d'origine, du

département d'arrivée et de la quantité transportée. Ce type de coût est illustré sur la Figure 4.1.

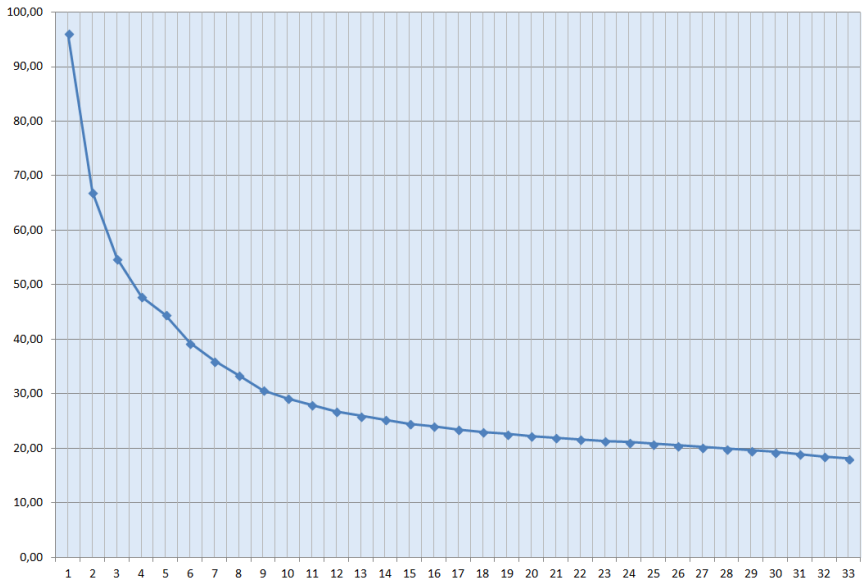


FIGURE 4.1 – *Exemple de coûts LTL issus d'une matrice de transport pour un département d'origine et un département de destination donné. L'axe horizontal représente le nombre d'unités logistiques à transporter (ici des palettes). L'axe vertical représente le coût de transport pour une unité logistique. Le coût unitaire de transport décroit donc fortement lorsque la quantité à transporter augmente.*

Les transports FTL et LTL ont pour objectif de distribuer des clients en France, ce qui constitue peut-être également une originalité par rapport aux approches rencontrées dans la littérature : les transports longue distance peuvent inclure un nombre limité d'arrêts intermédiaires si ceux-ci n'impliquent pas un détour excessif (typiquement, moins de 50 km). Pour chaque arrêt sera facturé un coût dit *d'ouverture de porte*, indépendant du détour réalisé. Concevoir le réseau logistique peut alors impliquer de déterminer des tournées comportant un faible nombre de sommets et respectant la contrainte de détour maximum.

Distributeurs régionaux

Un Centre de Distribution Régional (CDR) est une plate-forme pouvant recevoir des camions complets ou incomplets. Ces marchandises sont ensuite dispersées dans les tournées du distributeur. Les tarifs de distribution sont également basés sur des matrices de transport, avec des coûts unitaires décroissant fortement avec le nombre d'unités logistiques à livrer. Le contrat avec le distributeur ne fait en général pas intervenir de coût fixe mais il peut faire l'objet de négociations sur un nombre minimal annuel d'unités logistiques qui lui sont confiées.

4.1.2 Mutualisation

Les structures de coûts issues de la sous-traitance du transport expliquent déjà fortement l'intérêt d'une collaboration. Si plusieurs fournisseurs s'associent pour livrer un client commun, une

première réduction est obtenue avec l'accroissement de la quantité à transporter. L'aspect temporel peut ici jouer un rôle important : pour deux commandes vers le même client, il faut qu'elles soient livrées en même temps pour que la réduction de coût soit effective. Un second point lié à la massification des flux est la possibilité de démarcher des CDR qui seront approvisionnés en camions complets.

La consolidation des flux de fournisseurs géographiquement proches est réalisée de deux façons, qui peuvent être combinées. La première possibilité est de constituer des tournées de collecte pour les camions complets servant les CDR, ou des camions incomplets à destination des clients.

La seconde possibilité passe par l'ouverture d'un Centre De Consolidation (CDC), qui sert de plateforme locale. Le CDC est approvisionné par des camions pleins depuis les fournisseurs ou des navettes qui réalisent des tournées de collecte de tailles très limitées (deux ou trois arrêts). Le coût du CDC est modélisé par un *coût de passage à quai* : on estime que pour chaque unité logistique transitant sur la plateforme, un coût fixe de traitement est généré.

L'aspect collaboratif implique également de nombreuses questions sur le partage des coûts, la qualité du service et la relation avec les clients et prestataires. Ces questions n'ont néanmoins pas été abordées par l'équipe SLP dans les projets réalisés.

4.1.3 De l'originalité des problèmes abordés

Comme expliqué précédemment, les problématiques de design de réseaux de service sont rarement abordées sous l'angle d'un transport externalisé. Le problème de minimisation de coûts de camions complets dans un réseau à plusieurs échelons est par exemple traité par Gendron and Semet (2009). Les coûts de camions incomplets sont plus rarement intégrés et généralement modélisés par une fonction linéaire par morceaux. Ce type de coût a néanmoins été étudié par Lapierre et al. (2004), dans un problème de conception de réseau avec cross-docks. Croxton et al. (2003), présentent un modèle intéressant de linéarisation de la fonction de coût linéaire par morceaux pour la conception de réseaux de distribution faisant intervenir une consolidation des commandes avant livraison. Cette linéarisation est intégrée dans un modèle de génération de colonnes par Moccia et al. (2011) pour déterminer les itinéraires de marchandises dans un réseau multi-modal.

Ces précédents travaux ne prennent néanmoins pas en compte la possibilité d'avoir plusieurs points de collecte ou de distribution dans le trajet d'un véhicule. Lindsey et al. (2013) combine conception de tournées et conception de réseau avec différents types de coûts. Des coûts LTL sont intégrés pour les trajets directs. Mais les coûts considérés pour les tournées ne dépendent pas de leurs charges respectives.

4.1.4 Plan du chapitre

Ce chapitre résume les travaux réalisés en conception de réseau logistique mutualisé. Le projet LMD est tout d'abord présenté en Section 4.2. Le projet VEGESUPPLY est présenté en Section 4.3 et les travaux de thèse de Juliette Medina, réalisés en collaboration avec l'entreprise 4S-Network, sont introduits en Section 4.4.

4.2 Projet Logistique Mutualisée Durable

Le projet LMD a été financé pour un an en première phase par l'ADEME en 2008 dans le cadre du PREDIT. Ce projet avait pour ambition le développement d'une plateforme logicielle permettant la mise en commun et la gestion mutualisée de ressources logistiques (entrepôts, camions, ...) ainsi que le développement de méthodologies de pilotage de processus de collaboration. Les réflexions sur ce projet ont été nourries par l'expérience de trois industriels (Bénédicta, Nutrimaine et Pastacorp) ayant mutualisé leurs flux pour livrer en commun les plateformes de la grande distribution. Ce cas est dénommé par la suite « cas Bénédicta », en référence au porteur du projet de mutualisation. Il a été présenté au consortium par Xavier Perraudin, directeur logistique de Bénédicta à l'époque et actuel co-dirigeant de la société 4S-Network.

Le problème étudié par les équipes SLP (IRCCyN) et TASC (LINA) dans LMD a porté sur la conception d'un réseau de distribution mutualisé pour des clients communs à un ensemble de producteurs. Ce réseau est inspiré du cas Bénédicta. Dans ce problème on considère un flux annuel de demandes de clients aux fournisseurs. Une fréquence minimale de livraison est fixée en accord avec les clients. Celle-ci est modélisée par un nombre de livraisons annuel de chaque client dans le réseau. La mutualisation est supportée par un entrepôt déporté (CDC) opéré par la société FM Logistique et approvisionné en camions pleins depuis les fournisseurs. Les clients sont ensuite livrés par des tournées de camions incomplets tarifés sur la base de matrices de transport (coûts LTL). Les tournées considérées peuvent comporter jusqu'à trois points de collecte successifs, trois points de livraison successifs, sans que la longueur de la tournée ne soit supérieure de plus de 50 km à la distance entre les deux points les plus éloignés. Chaque arrêt intermédiaire implique un coût d'ouverture de porte. L'entrepôt mutualisé induit un coût de passage à quai, mais également un coût de stockage et de préparation pour une partie des flux.

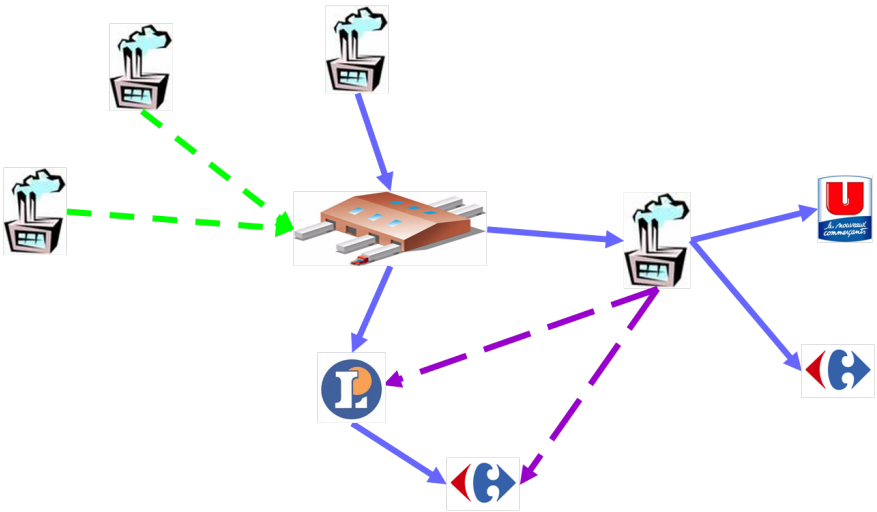


Figure 4.2 – *Illustration d'un réseau mutualisé dans le projet LMD. Les pointillés courts représentent des camions pleins. Les traits points et pointillés longs représentent des tournées de collecte et distribution en camions incomplets.*

L'objectif est de construire un réseau de distribution en déterminant : où sont ouverts les CDC ; quel fournisseur utilise un CDC et lequel ; les tournées de collecte/distributions ; et la

fréquence de chaque tournée. On considère séparément deux fonctions objectif : le nombre total de kilomètres parcourus (indicateur du CO_2 émis) ; et le coût de la solution.

Une heuristique dérivée de l'algorithme DROP (Daskin 1995) de localisation a été réalisée et utilisée pour résoudre le cas Bénédicta (3 fournisseurs, un CDC et 27 clients) ainsi qu'une extension de ce cas à 9 fournisseurs et 4 CDC. L'heuristique est basée sur la résolution d'un problème maître qui part d'une solution où chaque fournisseur et chaque CDC est un point d'expédition potentiel. Ces points sont progressivement fermés un à un et le point de chargement dans les tournées pour chaque fournisseur est déterminé. Le sous-problème est résolu par PLNE et permet de déterminer le coût des transports. Il considère l'ensemble des tournées réalisable et détermine l'ensemble de tournées de coût minimum permettant de satisfaire la demande de transport. On note que les contraintes sur les tournées dans ce problème permet de toutes les énumérer et de les inclure simultanément dans un PLNE.

La méthodologie suivie et les analyses en résultant sont détaillées dans le rapport du projet (PREDIT 3 - Groupe 5 2009). On observe notamment que l'utilisation d'un CDC permet de réduire de beaucoup le nombre total de kilomètres parcourus mais qu'elle est générateur de coûts au CDC (stockage, préparation, passage à quai). Pour minimiser les coûts, le modèle suggère une mutualisation basée sur des tournées de collecte. On peut néanmoins signaler que les coûts internes aux entreprises dans ce cas sont difficiles à évaluer. Les données utilisées dans cette étude sont sujets à discussion. Une solution minimisant le nombre de kilomètres parcourus pour le cas Bénédicta est illustrée en Figure 4.3.

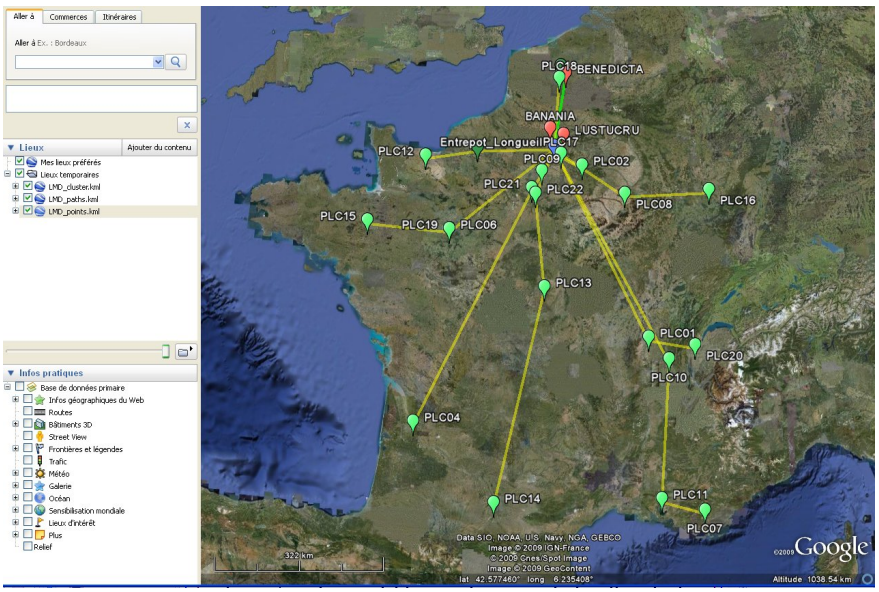


FIGURE 4.3 – *Représentation graphique avec Googe Earth du réseau de distribution mutualisé minimisant le nombre total de kilomètres parcourus pour le cas Bénédicta. Les trois entreprises sont en rouge, le CDC en bleu et les clients en vert. Le CDC est le point de départ de toutes les tournées.*

En dépit d'un prix attribué à LMD lors de la réunion « carrefour à mi-parcours du PREDIT 4 » de Bordeaux le 11 mai 2011, la proposition pour une seconde phase du projet n'a pas été déposée. Les travaux initiés dans le projet LMD sont néanmoins à l'origine des collaborations qui ont suivi avec la société 4S-Network (projet VEGESUPPLY et thèse Juliette Médina).

4.3 Conception d'un Réseau de Distribution de Végétaux Mutualisé

La conception d'un réseau de distribution mutualisé a été étudié sur un cas concret pour un ensemble de producteurs de végétaux de la région angevine, adhérents du pôle de compétitivité VEGEPOLYS. Une première étude d'opportunité a été réalisée début 2012 en collaboration avec la société 4S-Network pour le compte de sept producteurs et de VEGEPOLYS. Suite à cette étude, quatre producteurs ont décidé de mutualiser leurs expéditions et déploient actuellement leur réseau dans le cadre du projet VEGESUPPLY financé par le FUI. Cette expérience est particulièrement enrichissante car elle part d'un cas concret et soutient un initiative ambitieuse. En raison du caractère fragile des végétaux, les producteurs expédient des commandes vers plus de 5 000 clients en France, souvent en très petites quantités (les enseignes ne souhaitent pas manipuler les végétaux sur leurs plateformes). Les coûts de transport sont extrêmement élevés, si bien que des entreprises concurrentes d'une même région sont prêtes à s'associer pour garder leur compétitivité vis à vis de l'étranger.

L'étude d'opportunité a permis d'estimer les réductions de coût pouvant être espérées de la mutualisation. Les conclusions de cette étude sont largement supportées par des simulations réalisées à SLP. Celles-ci sont basées sur une méthode de décomposition réalisée conjointement avec Olivier Péton et présentée en Section 4.3.1. Les perspectives, actuellement à l'étude dans VEGESUPPLY, sont synthétisées en Section 4.3.2

4.3.1 Estimation des coûts de transport dans un réseau de distribution mutualisé

L'étude d'opportunité s'est essentiellement déroulée sur 4 mois début 2012. Elle a porté sur un an d'historique de commandes des sept producteurs impliqués. Une agrégation des données basée sur la saisonnalité de la demande et la fréquence de livraison des clients pour chaque saison a pris d'établir un ensemble de commandes pour 121 périodes. Pour chaque période, en fonction de paramètres modulant le niveau de mutualisation, l'optimisation permet de déterminer le coût d'acheminement des marchandises dans le réseau.

Le réseau le plus général considéré dans cette étude est illustré par la Figure 4.4. On considère un CDC et un ensemble de CDR dont les localisations sont connues. Trois modes de transports peuvent être utilisés :

- Des *expéditions*, qui livrent les clients depuis les producteurs, le CDC ou un CDR. Elles sont sous-traitées à un CDR ou à un transporteur avec un tarif LTL basé sur une matrice de transport.
- Des *tournées de camions complets* (FTL). Chaque tournée FTL livre un unique CDR et possède un à deux points de collecte chez un producteur ou au CDC.
- Des *tournées de navettes*, comportant un ou deux points de collecte chez les producteurs et livrant le CDC.

Les tournées possibles (camions complets et navettes) sont énumérées et leurs coûts respectifs sont déterminés avant la résolution. Un coût de passage à quai est compté pour chaque unité logistique passant par le CDC.

L'objectif est de déterminer le point d'expédition de chaque commande, le nombre de véhicules sur chaque tournée et le trajet des commandes depuis les producteurs vers les points d'expédition.

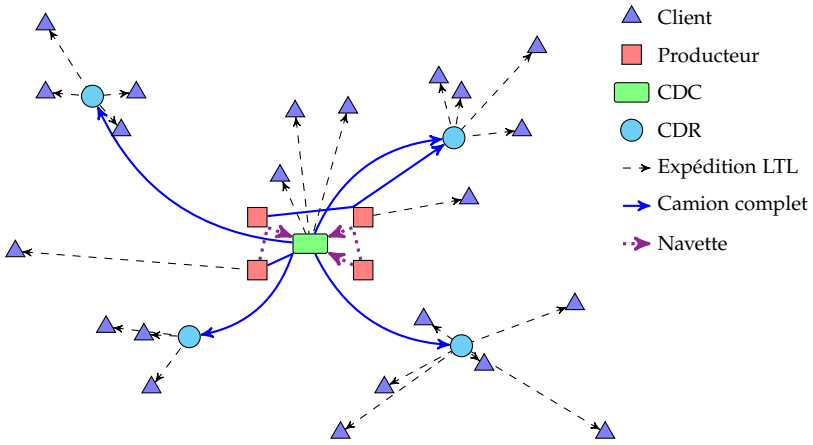


FIGURE 4.4 – *Réseau de distribution mutualisé du projet VEGESUPPLY*

Le problème est décomposé en trois PLNE résolus successivement par un solveur :

1. PLNE 1 : les producteurs et le CDC sont agrégés en un unique point ; le PLNE détermine le point d'expédition de chaque client et le nombre de véhicules complets à destination de chaque CDR.
2. PLNE 2 (une résolution par CDR) : les points d'expédition sont désagrégés ; pour chaque CDR, le PLNE détermine la partie collecte de chaque tournée FTL et affecte les demandes à ces tournées.
3. PLNE 3 : le nombre de véhicules sur chaque tournée de navette est déterminé en fonction des commandes à transporter au CDC.

Cette approche est relativement simple et pragmatique en raison des délais de l'étude. Elle a été comparée a posteriori à la résolution d'un modèle comprenant l'ensemble des variables du problème. Ce modèle global a été résolu sur les 121 instances du problème avec une limite de temps d'une heure. La résolution fournit de bonnes bornes inférieures du coût optimal. On montre que la décomposition, avec une limite de temps de 5 minutes, résout l'ensemble des instances avec un temps moyen inférieur à une minute par instance. La distance entre les solutions obtenues et les bornes inférieures sont de 0,75 % en moyenne. Les solutions de la décomposition ne sont jamais à plus de 1,5% du coût des solutions du modèle global. Pour les grosses instances en revanche, les solutions du modèle global ont un coût jusqu'à 40% plus important que celles de la décomposition.

Si l'on considère le nombre de simulations réalisées pour l'étude, on peut estimer que l'approche proposée s'est avérée extrèmement efficace. Un article la décrivant plus en détail a été soumis à une édition spéciale de la revue OR-Spectrum dédiée à la planification de réseaux de transport collaboratifs (Lehuédé et al. 2014). Ce travail a été réalisé en collaboration avec Xin Tang, post-doctorant sur le projet VEGESUPPLY.

4.3.2 Travaux en cours

La construction et mise en fonction du réseau VEGESUPPLY est actuellement en cours. Quatre producteurs se sont associés et ont monté la SAS VEGESUPPLY, chargé de l'affrètement

et de l'organisation des transports dans le réseau mutualisé. Une première étude en localisation a été réalisée pour supporter la recherche de partenaires logistiques en région (CDR). Un module opérationnel est en cours de réalisation. Il est destiné à déterminer quotidiennement les tournées de transport et la répartition des commandes dans les camions.

4.4 Centres de Routage Collaboratifs

La société 4S-Network a été fondée en 2009 par Xavier Perraudin suite au rachat par Heinz de la société Bénédicta. Il a été rejoint en 2010 par Christian Leroux, ex-responsable transport chez Unilever France. La mission de cette entreprise de service et de conseil en logistique est la suivante [1] :

> *Accompagner ses clients dans la réflexion, la conception et l'organisation de solutions « Supply Chain » novatrices et durables afin de répondre efficacement aux défis de demain.*

Dans le cadre de ses activités, 4S-Network développe le service CRC (Centre de routage collaboratif). Ce service vise à favoriser la coopération horizontale entre fournisseurs et distributeurs de la grande distribution, autour de plateformes de cross-dock mutualisées. Le service CRC propose une collaboration sur plusieurs niveaux :

- La mutualisation des ressources via la mise à disposition de plateformes de cross-dock gérées de façon régionale et appelées "CRC services". Proches des distributeurs, elle peuvent accueillir et router les commandes.
- La mutualisation des transports, facilitée via le regroupement sur ces plateformes des commandes ayant une destination commune.
- Une collaboration avec les transporteurs qui peuvent s'associer au projet via le transport, mais aussi via la mise à disposition de leurs plateformes de routage au service CRC.

La thèse de Juliette Médina a pour objet le développement d'outils décisionnels destinés à aider à la conception, à la vente et au fonctionnement du service.

Les premiers travaux, en cours de finalisation, ont fait l'objet d'un modèle de simulation utilisé pour la prospection des premiers utilisateurs du service. Dans l'esprit de l'étude d'opportunité qui a précédé VEGESUPPLY, le modèle est utilisé pour simuler la mutualisation et analyser son impact.

Le réseau considéré se veut très général. Il comporte plusieurs sites de fournisseurs, des sites de préparation, des CRC et les clients. On intègre des transports en camions complets et incomplets avec possibilité de réaliser des tournées préalablement établies. Les coûts LTL sont approximés dans le modèle par une fonction linéaire par morceaux. Les commandes sont complexes. Une commande peut comporter plusieurs produits provenant de différents sites. Certains doivent passer sur un site de préparation. Les parties d'une commande sont regroupées au CRC avant distribution au client. Cette distribution est réalisée par un ensemble de tournées, également considéré en extension.

1. http://www.4snetwork.com/fondements/

Un article est en cours de rédaction pour décrire le premier modèle réalisé et les analyses qu'il permet. Ce modèle permet de résoudre un problème mono-période. Dans la suite de la thèse, on doit s'intéresser au cas multi-périodes et intégrer la dimension temporelle au problème. Une première approche est à l'étude pour ce dernier problème, beaucoup plus complexe. Elle est inspirée de Moccia et al. (2011), et est basée sur une génération de colonnes pour la détermination des trajets des produits dans le réseau.

4.5 Conclusion

J'ai développé mon activité de recherche en conception de réseaux de services de transport de manière relativement opportuniste et sur la base d'applications concrètes. Ces travaux puisent leur originalité dans la pratique et les besoins des entreprises. Je tiens à souligner que ces travaux sont rendus possibles grâce à l'ouverture d'esprit et l'expertise des dirigeants de 4S-Network, qui nous ont permis à chaque fois de faire le pont entre les pratiques en logistique et les modèles d'optimisation.

Du point de vue scientifique, les travaux achevés sont relativement simples. J'apprécie néanmoins grandement qu'en dépit de sa simplicité, l'approche développée dans l'étude précédant VEGESUPPLY soit redoutablement efficace. Le fait d'avoir pu démontrer l'efficacité de l'approche constitue pour moi un travail de recherche qui, j'espère, pourra servir d'autres intérêts.

Les projets en cours sont enthousiasmants. Tout d'abord, la réalisation d'un module de routage opérationnel pour VEGESUPPLY est un challenge non négligeable, autant du point de vue recherche opérationnelle que du point de vue informatique. Ensuite, le soutient à l'initiative CRC services de 4S-Network est un défi important, source de problèmes d'optimisation complexes. La prise en compte de contraintes temporelles dans le réseau, en combinaison avec les différentes caractéristiques à intégrer, nous pousse à concevoir des méthodes ambitieuses pour produire des outils utilisables en pratique.

Chapitre 5

Conclusion Générale et Perspectives de Recherche

Ce mémoire résume l'ensemble de mes travaux, réalisés depuis mon arrivée dans l'équipe SLP en 2006. Au cours des huit dernières années, je me suis intéressé de manière croissante au domaine passionnant des transports. Mon souhait est de continuer à développer ma recherche dans ce domaine, qui est au cœur des défis que doit relever notre société. Le changement environnemental, nécessaire, passera par des innovations technologiques. Les systèmes d'information et la recherche opérationnelle ont néanmoins leur rôle à jouer. En permettant une meilleure utilisation des moyens bien sûr, mais également en aidant au développement de nouveaux systèmes, mieux intégrés et collaboratifs.

5.1 Perspectives

De nombreuses perspectives ont été discutées dans les trois chapitres de contribution de ce mémoire. Les problèmes de tournées avec synchronisation sont encore peu explorés et je souhaite clairement continuer à m'y intéresser. Les connaissances acquises en AMCD constituent également une sensibilité que souhaite mettre à profit dans mes recherches en transport.

Dans la poursuite de mes recherches, il existe une relation entre intégration des aspects temporels dans les problèmes de conception de réseaux et les tournées avec synchronisation. J'espère que ces deux thèmes pourront se rejoindre et se nourrir pour permettre encore la réalisation de projets ambitieux.

5.2 Des applications à la recherche

Mon expérience m'incite à croire que les projets réalisés en relation avec des associations et entreprises sont d'inépuisables sources d'inspiration pour la recherche. Chaque travail a, à plus ou moins long terme, débouché sur des réalisations innovantes. Ces projets sont à chaque fois le fruit de rencontres avec des partenaires particulièrement ouverts et il convient de saluer leur esprit d'initiative. Les projets en cours avec la société 4S-Network notamment, ou l'entreprise Luc Durand dans le projet ORLOGES, contiennent de multiples sujets encore à explorer.

5.3 De la poursuite d'une recherche de qualité

Néanmoins la recherche partenariale n'est pas nécessairement la plus « efficace » sur le plan purement académique. Des réalisations conséquentes, comme le logiciel Marika, ne débouchent pas nécessairement à court terme sur des publications. La recherche de partenaires est également parfois laborieuse. Un équilibre est donc à trouver pour prendre le temps d'une recherche de qualité. Les financements publics et collaborations nationales et internationale sont indispensables sur ce point. Les thèses de Renaud Masson (bourse MENRT), et Philippe Grangier (sur financement 50% EMN, 50 % CIRRELT), ont permis et permettent d'investiguer des problèmes de fond et de généraliser des résultats pour en accroitre l'impact.

L'intégration à une communauté me semble, par ailleurs, essentielle. Des projets, même avec un support financier modeste, comme MC-DARP (PHC Amadeux) ou CTouVer (GdR RO), permettent de s'enrichir considérablement sur le plan scientifique grâce à la collaboration avec d'autres chercheurs. L'organisation de la conférence VeRoLog à l'école des Mines de Nantes en 2016, conjointement avec plusieurs collègues de l'EMN et d'autres universités, est également un évènement que j'attends avec impatience.

5.4 De la recherche partenariale à la formation

La formation est un élément clé de mon métier, qui puise énormément dans l'expérience acquise sur les applications. Il est bien sûr motivant de former des étudiants de master ou doctorants aux techniques de recherche opérationnelle. Un des défis qui me semble le plus intéressant est de former les futurs logisticiens de l'option GOPL ou du master MOST de l'EMN. J'espère pouvoir, à travers mes cours et l'accompagnement de projets, contribuer à en faire des ingénieurs ouverts à l'innovation, qui pourront pousser nos entreprises vers les côtés les plus intelligents du transport et de la logistique.

Chapitre A
Curriculum Vitae

Le Curriculum Vitae n'a pas été imprimé pour cette version de mon HDR. Merci de vous reporter au dossier de candidature joint.

Chapitre B

Sélection d'articles

Les articles suivants sont annexés à la suite ce manuscrit :

1. Fabien Lehuédé, Michel Grabisch, Christophe Labreuche, Pierre Savéant. Integration and Propagation of a Multicriteria Decision Model in Constraint Programming. *Journal of Heuristics*, 2006, 12 (4-5), pp. 329-346.
2. Renaud Masson, Fabien Lehuédé, Olivier Péton. An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Transfers. *Transportation Science*, 2013, 47 (3), pp. 344-355.
3. Renaud Masson, Fabien Lehuédé, Olivier Péton. Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, 2013, 41, pp. 211-215.
4. Dominique Feillet, Thierry Garaix, Fabien Lehuédé, Olivier Péton, Dominique Quadri. A new consistent vehicle routing problem for the transportation of people with disabilities. *Networks*, Wiley-Blackwell, 2014, 63 (3), pp. 211-224.
5. Philippe Grangier, Michel Gendreau, Fabien Lehuédé, Louis-Martin Rousseau. An Adaptive Large Neighborhood Search for the Two–Echelon Multiple-Trip Vehicle Routing Problem with Satellite Synchronization. *European Journal of Operational Research*, 2016, 254 (1), pp.80-91.

J Heuristics (2006) 12: 329–346
DOI 10.1007/s10732-006-8075-2

Integration and propagation of a multi-criteria decision making model in constraint programming

F. Le Huédé · M. Grabisch · C. Labreuche · P. Savéant

Abstract In this paper we propose a general integration scheme for a Multi-Criteria Decision Making model of the Multi-Attribute Utility Theory in Constraint Programming. We introduce the Choquet integral as a general aggregation function for multi-criteria optimization problems and define the Choquet global constraint that propagates this function during the Branch-and-Bound search. Finally the benefits of the propagation of the Choquet constraint are evaluated on the examination timetabling problem.

Keywords Multi-criteria optimization · Constraint programming · Multi-criteria decision making

1. Introduction

The practice and developments of Constraint Programming (CP) and Operations Research have shown that, in many cases, two complex issues have to be taken into account when addressing real-life optimization problems. First, industrial problems are often highly combinatorial and it is often difficult to solve them fast. Secondly, the preference relation between solutions to a problem is generally complex and depends on several conflicting criteria.

As a result, several formalisms and frameworks have been introduced in CP to handle complex preference relations in optimization problems. For example the Soft Constraints, Preference Based Search, CP-nets frameworks propose various approaches to model a preference relation over several criteria. A common issue when defining a very precise and

F. Le Huédé (✉) · C. Labreuche · P. Savéant
THALES Research and Technology France,
domaine de Corbeville, 91401 Orsay cedex
e-mail: {fabien.lehuede; christophe.labreuche; pierre.saveant}@thalesgroup.com

M. Grabisch
Université Paris I Pantheon-Sorbonne

M. Grabisch
LIP 6, Université Pierre et Marie Curie (UPMC), 8, rue du Capitaine Scott 75015 Paris
e-mail: Michel.Grabisch@lip6.fr

elaborate preference relation, is elicitating the preferences of an expert in order to enable the framework's model to reproduce his choice in a combinatorial search space.

Multi-Criteria Decision Making (MCDM) proposes several methodologies and tools that provide accurate models for the preference of an expert over several criteria. Research in this area focuses mainly on constructing appropriate models for preferences and on processes that produce a correct elicitation of subjective preferences. MCDM models either evaluate or rank a given set of solutions (also called alternatives).

Surprisingly, few approaches integrate an MCDM model in their framework in order to benefit both of the capacities of the model for preference modeling and of the elicitation methodology that has been designed for this model. As a consequence, when a complex preference relation is needed, CP solvers offer few possibilities to handle the problem from the multi-criteria point of view.

1.1. Related work

Many solutions were proposed for integrating multi-criteria preferences in CP. A first category of approaches concentrates on finding a set of Pareto-optimal solutions for different objective functions (Gavanelli, 2002; Barichard and Hao, 2003; Ben Jaâfar et al., 2004). Alternatively, configuration problems and web applications have given rise to several approaches where a formalism allows the user to express partial preferences on criteria in a simple way (Junker, 2004; Pu and Faltings, 2004).

In this section, we focus on methods that allow a sufficiently precise description of the preference relation to be given in order to search for an (approximately) optimal solution. For this concern, most approaches rely on one of the two following frameworks.

The first well known framework is soft-constraints and in particular, Weighted CSPs (Bistarelli et al., 1999). It has been designed to solve over-constrained problems, but modeling multi-criteria problems using soft constraints is often suggested. Expressing preferences as constraints in order to describe a complete preference relation may be however problematic when the number of attributes to be taken into account is large. The method therefore needs a well founded and careful preference elicitation process. Indeed, even for limited partial preference relations, the simple example given in (Pu and Faltings, 2004) shows that one cannot manually modify weights in constraints and expect to keep a clear view of the relative importance of criteria in this relation.

The second framework, called CP-nets (Boutilier et al., 2004), proposes asking the user preference rules such as "if the main meal is fish, then I prefer white wine to red wine". A partial or complete preference relation can then be expressed using this kind of rules over the attributes of a solution. This framework is very expressive and can, in particular, handle context-dependent preferences. However, when the number of attributes and the number of values per attributes gets high, the description of a quasi-complete preference relation implies that a great number of rules are expressed. Recent studies propose integrating CP-nets in CP for optimization (Boutilier et al., 2004; Prestwich et al., 2005). (Prestwich et al., 2005) propose modeling CP-nets with hard constraints thanks to a new semantics. The use of implication constraints in this model suggests that the propagation of a CP-net in a combinatorial problem has few chances to be very efficient during the search for solutions. When the criteria of the problem are clearly different of the decision variables, it is likely that most solutions have to be constructed to ensure that the returned solution is not dominated.

Finally, (Kaymak and Sousa, 2003) proposes a framework for combining a multi-criteria model and a Fuzzy CSP. The constraints proposed in the following of this paper can be used to model the multi-criteria part in this framework.

1.2. Contributions of the paper

In this paper we study the integration of a model of the Multi-Attribute Utility Theory (MAUT) in CP. This model is introduced in Section 2. In Section 3, we propose a general scheme that offers a good flexibility for integrating elaborate multi-criteria models. We then introduce the principles of the propagation of the Aggregation constraint, which models the class of multi-criteria aggregation functions. The case of the Choquet integral is then studied in detail (Section 4). As this aggregation functions is particularly adapted to preference modeling, we apply the principles of Aggregation to define the Choquet constraint. Specific propagation algorithms are then designed and applied in Section 5 to a multi-criteria version of the examination timetabling problem.

2. Modeling multi-criteria preferences

In this paper, we focus on the *Multi-Attribute Utility Theory* framework (Keeney and Raiffa, 1976). This framework models the value of a solution through an overall evaluation, computed by an aggregation function according to its levels of satisfaction on a set of criteria. We introduce the properties of a multi-criteria aggregation function that are the most desirable in order to give a good representation of multi-criteria preferences. For this purpose the Choquet integral is a very general aggregation function which can model a wide range of decisional behaviors according to well founded elicitation processes (Grabisch and Roubens, 2000; Labreuche and Grabisch, 2003).

2.1. Preference modeling in Multi-Criteria Decision Making

Multi-Criteria Decision Making (MCDM) models subjective preferences in order to automate the determination of a preferred solution out of a set of alternatives. Hence, solving a typical multi-criteria decision problem consists in modeling the way an expert ranks a set of potential solutions, described by a set of *attributes* or *points of view*. To achieve this objective, the Multi-Attribute Utility Theory is mainly concerned with the construction of additive utility functions.

Let us denote by $\mathcal{N} = \{1, \ldots, n\}$ the set of criteria. We assume a set of *solutions* or *alternatives* $\mathcal{S}$ among which the decision maker must choose. Each solution is associated with a vector $a \in \Omega$ whose components $a_i \in \Omega_i$, $i \in \{1, \ldots, n\}$ represent the value of the solution for each point of view to be taken into account in the decision making process. A component a_i is called the *attribute* of a solution. Typically, in a multi-objective optimization context, each attribute would correspond to an objective function. According to these attributes, the modeling of the decision maker's preferences $\succeq$ is realized through an *overall utility function* $u : \Omega \to \mathbb{R}$ such that:

$$\forall a, b \in \Omega, a \succeq b \Leftrightarrow u(a) \geq u(b), \tag{1}$$

where $\succeq$ is a complete pre-order.

Classically, this overall evaluation function is split into two parts (Keeney and Raiffa, 1976):

– The *utility functions*, denoted $u_1(a_1), \ldots, u_n(a_n)$, map each attribute to a single satisfaction scale $\mathcal{E} \in \mathbb{R}$. They model the *performance* of a solution on the criteria and ensure

commensurateness between criteria, which is essential when several values have to be aggregated.
– The *aggregation function* aggregates the values returned by $u_1, \ldots, u_n$ and establishes the overall evaluation: $\forall a \in \Omega, u(a) = \mathcal{H}(u_1(a_1), \ldots, u_n(a_n))$.

where $u_i : \Omega_i \to \mathcal{E}$ and $\mathcal{H} : \mathcal{E}^n \to \mathcal{E}$. $u_i(a_i)$ is called the *utility* or *score* of the alternative a on the criterion i.

A common value for the satisfaction scale $\mathcal{E}$ is the [0, 1] interval. Establishing commensurateness between criteria allows us to work with comparable values. For example, this implies that we are able to express that a makespan of 20 days corresponds to the same level of satisfaction as a maximum tardiness of 3 days. Furthermore, when using compensatory aggregators (such as the weighted sum), it is important to work on an *interval scale*. This means that the difference between two values on the same criterion has to make sense. For example, it consists in deciding whether the difference of satisfaction of going from 5 days to 10 days is the same as going from 15 days to 20 days. To construct utility functions in the MAUT framework, we use the MACBETH methodology (Bana e Costa and Vansnick, 1994) (and its associated software), which is based on measurement theory. MACBETH asks the user to give some reference levels for a criterion and some general indications on the difference of satisfaction between values of this attribute.

2.2. Properties of a multi-criteria aggregation function

In decision aid, an aggregation function aggregates commensurate values which model criteria satisfaction levels. Some essential requirements have to be met by the aggregation function $\mathcal{H}$. Among these important properties, which are detailed in (Marichal, 1998), we denote:

– *Monotonicity* (M): $\mathcal{H}$ should be an increasing function. That is to say:

$$x_i > x_i' \Rightarrow \mathcal{H}(x_1, \ldots, x_i, \ldots, x_n) \geq \mathcal{H}(x_1, \ldots, x_i', \ldots, x_n).$$

Indeed, if one solution is better than another on at least one criterion and has equal performances on the other criteria, it cannot be of lesser quality than the other solution. This property is called *Strict Monotonicity* when the right side of the implication is a strict inequality.
– *Continuity* (C): $\mathcal{H}$ should be continuous with respect to its argument as the preferences of the expert generally evolve progressively.

In addition, $\mathcal{H}$ should satisfy (1) when $\succeq$ is defined. Consequently, if we want to use the same parametric model in several applications, the aggregation function has to be very flexible. Additive aggregation functions, usually used in MAUT, suppose *preferential independence* between criteria (Keeney and Raiffa, 1976), which is seldom the case in decision making. The aggregation function has to be able to model the importance of a criterion, but also interaction and compensation effects between criteria. As a result, we propose using an enhanced version of this theory, with a more general aggregation function.

2.3. The Choquet integral

In multi-criteria decision problems, some general aggregation functions such as the *Choquet integral* (Choquet, 1953; Grabisch, 1996) are often necessary in order to take into account not only the importance of each criterion, but also interaction phenomena between the criteria. In

order to generalize the weighted sum, (Sugeno, 1974) proposes assigning weights not only to each criterion separately, but also to any coalition of criteria. This weighting corresponds to a set function called "fuzzy measure".

Definition 1 (Fuzzy measure (Sugeno, 1974)). Let $\mathcal{P}(\mathcal{N})$ be the power set of $\mathcal{N}$. A fuzzy measure μ on $\mathcal{N}$is a function $\mu : \mathcal{P}(\mathcal{N}) \to [0, 1]$, satisfying the following axioms.

(i) $\mu(\emptyset) = 0,\ \mu(\mathcal{N}) = 1.$
(ii) $A \subset B \subset \mathcal{N}$ implies $\mu(A) \leq \mu(B)$.

Here, $\mu(A)$ represents the degree of importance of the subset of criteria $A \subset \mathcal{N}$. In the MCDM methodology, the fuzzy measure is used to model the decision maker preferences (Grabisch and Roubens, 2000). Then, the mono-dimensional utilities $u_1, \ldots, u_n$ are aggregated with the Choquet integral to produce the overall evaluation of an alternative.

Definition 2 (The Choquet integral (Choquet, 1953)). Let μ be a fuzzy measure on $\mathcal{N}$, and $u = (u_1, \ldots, u_n) \in [0, 1]^n$. The Choquet integral of u with respect to μ is defined by:

$$C_\mu(u_1, \ldots, u_n) = \sum_{i=1}^{n} u_{\sigma(i)}[\mu(A_{\sigma(i)}) - \mu(A_{\sigma(i+1)})], \tag{2}$$

where $\sigma(i)$ indicates a permutation on $\mathcal{N}$ such that $u_{\sigma(1)} \leq \cdots \leq u_{\sigma(n)}$, $A_{\sigma(i)} = \{\sigma(i), \ldots, \sigma(n)\}$ and $A_{\sigma(n+1)} = \emptyset$.

Thus, for three criteria, $C_\mu(u_1, u_2, u_3) = \mu(\{\sigma(3)\}) \times u_{\sigma(3)} + [\mu(\{\sigma(2), \sigma(3)\}) - \mu(\{\sigma(3)\})] \times u_{\sigma(2)} + [1 - \mu(\{\sigma(2), \sigma(3)\})] \times u_{\sigma(1)}$. The best score gets its individual weight whereas the others are corrected with respect to better criteria.

The Choquet integral is continuous, increasing, idempotent, stable for positive linear transformation and linear for a given order of its components. An axiomatization of its use for preference modeling has been introduced in Marichal (1998). Efficient indicators are also available for the semantic interpretation of the fuzzy measure (Grabisch, 2000). Ensuring that the Choquet integral is strictly increasing can be easily done setting $\forall A, B \subset \mathcal{N}, A \subset B \Rightarrow \mu(A) < \mu(B)$.

The Choquet integral is a general aggregation function which takes as particular cases the weighted sum, the min and the max. In addition, it allows every linear interpolations between these functions, which makes it very expressive.

Figure 1 shows two representations of the Choquet integral on two criteria. The first curve represents a case where the interaction between the two criteria is positive (they are said to be *complementary*). It models a preference relation where a solution has to be good on both criteria to be considered good. On the contrary, the right hand curve models *substitutive* criteria (i.e., negative interaction). In this case, a solution is considered good by the expert as soon as it is good on one criterion.

Efficient tools and methodologies have been created to establish a good multi-criteria model. In Thales, we use the Myriad© (Labreuche and Le Huédé, 2005) software to help the analyst in building the criteria hierarchy. It calls MACBETH to construct utility functions (§ 2.1), and determines the Choquet integral coefficients according to the decision maker preferences. As these coefficients are complex, they cannot be set by hand. The expert is asked to make pairwise comparisons on various solutions. They are used to build a linear program and deduce the fuzzy measure (Grabisch and Roubens, 2000).

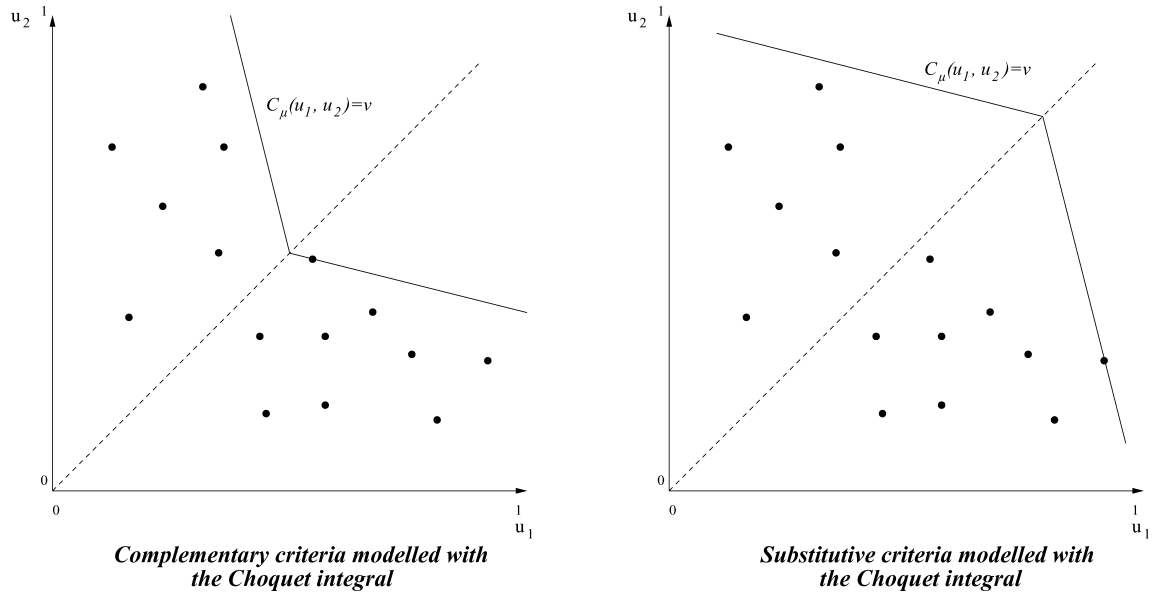


Fig. 1 Level curves of the Choquet integral for the aggregation of two criteria

Example 1 (The Choquet integral for students evaluation). Consider a director who would like to evaluate some students according to scores in mathematics (M), statistics (S) and languages (L). The director has some preferences such as *"for students good in mathematics, a student good in languages is preferred to one good in statistics"* and conversely *"for students bad in mathematics, a student good in statistics is preferred to one good in languages"*. Thus, for scores in [0, 1], this can be expressed as two "learning examples": $(0.8, 0.5, 0.4) \succ (0.8, 0.6, 0.3)$ and $(0.2, 0.6, 0.3) \succ (0.2, 0.5, 0.4)$. Then, to identify trade-offs between criteria, the director is asked to answer questions such as *"which value would you give to score* α *in the equivalence* $(0.5, 0.8, 0.4) \equiv (0.5, 0.7, \alpha)$*?"*. Setting $\alpha = 0.5$ and specifying that (M) is the most important criterion, we obtain the following fuzzy measure: $\mu(\{M\}) = \mu(\{S\}) = 0.5$, $\mu(\{L\}) = 0$, $\mu(\{M, S\}) = \mu(\{S, L\}) = 0.5$, $\mu(\{M, L\}) = 1$ that reflects the director requirements. The reader can note that, according to these preferences, the relative importance of S compared to L is conditional on M being good or bad. This type of decision strategy occurs quite often in preference modeling. They are represented for instance in TCP-nets in a qualitative framework. One can easily check that neither the weighted sum nor the minimum or the maximum can model these preferences.

For a more detailed application of this model in MCDM, the reader can report to (Labreuche and Le Huédé, 2005).

3. Integrating the MAUT model in CP

Our main objective is to capitalize on MCDM models and preferences elicitation methods in order to offer accurate multi-criteria optimization functionalities in CP. Considering that the MAUT model establishes an overall utility for each solution it can be quite naturally integrated in CP as a general objective function for multi-criteria optimization problems.

A simple MAUT model is composed of the following components (Section 2.1): *attributes*, which are generally objective functions in optimization, *utility functions* (one per attribute), which establish the scores of a solution on the criteria, *an aggregation function*, that makes the synthesis of the scores in order to calculate the overall evaluation of a solution. More elaborate preference models generally include several hierarchical levels of aggregation.

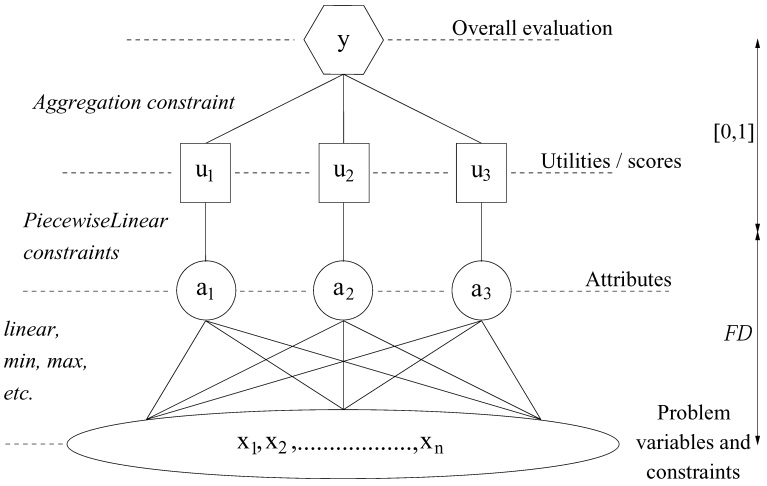


Fig. 2 MAUT integration scheme

For an efficient and flexible integration of MAUT in CP we propose modeling each function with a dedicated global constraint, considering more particularly the Choquet integral case. This leads us to the following integration scheme.

3.1. Integration scheme of the MAUT model

Let us consider a simple model with three criteria and a multi-criteria aggregation function $\mathcal{H}$. Figure 2 describes the proposed integration approach, where each function of the preference model is modeled by a global constraint.

According to this scheme, modeling a multi-criteria optimization problem implies defining several kind of variables: $y \in [0, 1]$ corresponds to the overall evaluation, $u_1, \ldots, u_n \in [0, 1]$ are the scores of a solution over each criterion, $a_1, \ldots, a_n$ are finite domain variables that model the attributes of the problem, $x_1, \ldots, x_m$ are the finite domain variables of the combinatorial problem.

The combinatorial problem is modeled with constraints on variables $x_1, \ldots, x_m$ that are connected to variables $a_1, \ldots, a_n$ by objective functions using constraints such as sum, min or max. The two upper levels of the scheme represent the multi-criteria model. Each attribute $a_i, i \in \{1, \ldots, n\}$ is connected to a score u_i by a PiecewiseLinear constraint that models a utility function. In order to enforce the relation $y = \mathcal{H}(u_1, \ldots, u_n)$, we introduce the Aggregation constraint which connects $u_1, \ldots, u_n$ to the variable $y \in [0, 1]$ that will be maximized.

Hence, two global constraints are needed: PiecewiseLinear models piecewise linear functions and Aggregation models the aggregation function $\mathcal{H}$.

In this scheme, Aggregation represents the generic class of constraints that model MCDM aggregation functions (§ 2.2). Some general properties for the propagation of this class of constraints are introduced in Section 3.2. These properties are then applied in Section 4 to the Choquet integral for the design of the Choquet constraint, which can then be seen as a specialization of Aggregation.

Remark 1 (Discretizing the continuous part of the model). In this integration scheme, variables $x_1, \ldots, x_m$, $a_1, \ldots, a_n$ are finite domain variables and $y, u_1, \ldots, u_n$ belong to the $[0, 1] \subset \mathbb{R}$ interval. To integrate this scheme in a finite domain solver, we discretized the $[0, 1]$ interval into the $\{0, \ldots, 10^6\}$ domain. As for propagation on intervals in CP (Lhomme, 1993), this approach needs "outward rounding" rules (Michel et al., 2001) to guarantee the correctness of algorithms. In particular, using outward rounding on discretized variables does not perturb the results presented in this paper. An interval $[\underline{x}, \overline{x}] \subset [0, 1]$ is represented by

the $\{\lfloor 10^6 \times \underline{x} \rfloor, \ldots, \lceil 10^6 \times \overline{x} \rceil\}$ domain. Discretizing [0, 1] and outward rounding are often considered as implicit in the following of this paper.

Remark 2 (The PiecewiseLinear constraint). Continuous piecewise linear functions are not imposed by MAUT but they correspond to the functions that are designed by the MACBETH software (Bana e Costa and Vansnick, 1994) (§ 2.1). As no a priori shape can be assumed for utility functions, most required shapes can be approximated with piecewise linear functions, which also offer a good understanding of their behavior between two characteristic points.

The design of the PiecewiseLinear constraint has been proposed by P. Refalo using linear relaxations (Refalo, 1999). Here, we consider only strict monotonous utility functions which are very easy to integrate in CP. Hence, to propagate $y = f(x)$ where f is a strictly increasing piecewise linear function, we just have to maintain $\underline{y} \geq \lfloor f(\underline{x}) \rfloor$ and $\overline{y} \leq \lceil f(\overline{x}) \rceil$ for the "discretized" variable y, and $\underline{x} \geq \lceil f^{-1}(\underline{y}) \rceil$ and $\overline{x} \leq \lfloor f^{-1}(\overline{y}) \rfloor$ for the integer variable x.

3.2. The Aggregation constraint

As stated in § 2.2, multi-criteria aggregation functions generally verify the Strict Monotonicity and Continuity properties. Some common principles can be identified for propagating the constraints that model these functions. To introduce these principles we define the Aggregation constraint, which establishes and propagates the equality between a variable y and the aggregation of scores $u_1, \ldots, u_n$ by a function $\mathcal{H}$. Mathematically we want to enforce $y = \mathcal{H}(u_1, \ldots, u_n)$.

Definition 3 (The Aggregation *constraint).* Let $\mathcal{N}$ be a set of n criteria, let $\{y\} \bigcup \{u_1, \ldots, u_n\}$ be a set of variables ranging over [0, 1] and let $\mathcal{H}$ be a continuous and strictly increasing multi-criteria aggregation function. The Aggregation constraint enforces the relation $y = \mathcal{H}(u_1, \ldots, u_n)$ and is denoted aggregation$(\mathcal{H}, y, \{u_1, \ldots, u_n\})$.

According to $\mathcal{H}$, the Aggregation constraint can be specialized to an adequate constraint that inherits its properties. Many studies deal with the modeling of the weighted sum in CP (see e.g. (Apt, 1998; Zhang and Yap, 2000)). Particularly, Apt (Apt, 1998) defines propagation rules for the Linear Equality constraint on $\mathbb{R}$. These results can be generalized to the whole set of multi-criteria aggregation functions.

3.3. Arc-B-Consistency of the Aggregation constraint

As the MAUT model relies on real variables, variables domains are limited to intervals throughout of the paper. We denote $[\underline{x}, \overline{x}]$ the domain of a variable x. Hence, the propagation of *Aggregation* can be achieved by maintaining the arc-B-consistency for this constraint.

Definition 4 (Arc-B-consistency). (Lhomme, 1993) Given a constraint c over q variables $x_1, \ldots, x_q$, and a domain $d_i = [\underline{x_i}, \overline{x_i}]$ for each variable x_i, c is said to be "arc-B-consistent" if and only if for any variable x_i and for each bound $v_i = \underline{x_i}$ and $v_i = \overline{x_i}$, there exist values $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_q$ in $d_1, \ldots, d_{i-1}, d_{i+1}, \ldots, d_q$ such that $c(v_1, \ldots, v_q)$ holds.

Arc-B-consistency is weaker than the *arc-consistency* property. Indeed a constraint is arc-consistent when, for each value in the domain of each of the constraint's variables, there is a set of values in the domain of the other variables that verifies the constraint.

The Strict Monotonicity and Continuity properties of function $\mathcal{H}$ enables us to define four necessary and sufficient conditions that ensure Arc-B-consistency with respect to the Aggregation constraint and avoid searching for a set of value that verify the constraint for each variable bound.

Proposition 1. *(Arc-B-consistency with respect to the Aggregation constraint) Let $\mathcal{H}$ be a strictly increasing and continuous aggregation function and C =Aggregation($\mathcal{H}$, y, $\{u_1, \ldots, u_n\}$) be an Aggregation constraint. C is Arc-B-consistent if and only if the following four conditions hold:*

(AB1) $\underline{y} \geq \mathcal{H}(\underline{u_1}, \ldots, \underline{u_n})$
(AB2) $\overline{\overline{y}} \leq \mathcal{H}(\overline{\overline{u_1}}, \ldots, \overline{\overline{u_n}})$
(AB3) $\forall k \in \{1, \ldots, n\} \ : \ \mathcal{H}(\overline{u_1}, \ldots, \overline{u_{k-1}}, \underline{\underline{u_k}}, \overline{u_{k+1}}, \ldots, \overline{u_n}) \geq \underline{y}$
(AB4) $\forall k \in \{1, \ldots, n\} \ : \ \mathcal{H}(\underline{u_1}, \ldots, \underline{u_{k-1}}, \overline{\overline{u_k}}, \underline{u_{k+1}}, \ldots, \underline{u_n}) \leq \overline{\overline{y}}$

Notation 1. For the sake of concision, we use the following notations: $\mathcal{H}(\underline{u}) = \mathcal{H}(\underline{u_1}, \ldots, \underline{u_n})$, $\mathcal{H}(\overline{u}) = \mathcal{H}(\overline{u_1}, \ldots, \overline{u_n})$, $\mathcal{H}(\overline{u_{-k}}, \underline{u_k}) = \mathcal{H}(\overline{u_1}, \ldots, \overline{u_{k-1}}, \underline{u_k}, \overline{u_{k+1}}, \ \ldots, \ \overline{u_n})$, $\mathcal{H}(\underline{u_{-k}}, \overline{u_k}) = \mathcal{H}(\underline{u_1}, \ldots, \underline{u_{k-1}}, \overline{u_k}, \underline{u_{k+1}}, \ \ldots, \ \underline{u_n})$.

Proof: Let us consider the y variable and assume conditions (AB1) and (AB2) are verified. Since since $\mathcal{H}$ is increasing, (AB1) and (AB2) implies $\underline{y}$ and $\overline{y}$ belong to the interval $[\mathcal{H}(\underline{u}), \mathcal{H}(\overline{u})]$. Function $\mathcal{H}$ being continuous, for each value $v \in \{\underline{y}, \overline{y}\}$, there exist values $v_1, \ldots, v_n$ in $[\underline{u_1}, \overline{u_1}], \ldots, [\underline{u_n}, \overline{u_n}]$ such that $v = \mathcal{H}(v_1, \ldots, v_n)$. Conversely, for any value $v \notin [\mathcal{H}(\underline{u}), \mathcal{H}(\overline{u})]$, $\mathcal{H}(\underline{u})$ and $\mathcal{H}(\overline{u})$ being the lowest and highest values that can be taken by $\mathcal{H}$ respectively, there is no set of values that satisfies C in the domains of $u_1, \ldots, u_n$. The domain $[\underline{y}, \overline{y}]$ is arc-B-consistent w.r.t. the Aggregation constraint C if and only if conditions (AB1) and (AB2) are verified.

Similarly, for a variable $u_k \in \{u_1, \ldots, u_n\}$, (AB3) and (AB4) ensure that $[\underline{u_k}, \overline{u_k}]$ is arc-B-consistent with respect to C. □

Proposition 1 is very straightforward. It should be noted that it is not valid however when variables are integer or when $\mathcal{H}$ is not continuous.

Conditions (AB1) and (AB2) propagate Aggregation on the domain of variable y and for each variable u_k, $k \in \{1, \ldots, n\}$, conditions (AB3) and (AB4) propagate on $\underline{u_k}$ and $\overline{u_k}$ respectively. A first remark is that when (AB1) or (AB2) is not verified, it is easy to compute a new valid domain for y: $[max(\underline{y}, \mathcal{H}(\underline{u})), \ min(\overline{y}, \mathcal{H}(\overline{u}))]$. This deduction may be more difficult for a variable $u_k \in \{u_1, \ldots, u_n\}$. It implies expressing u_k with respect to the other variables, which may be complicated if the inverse function of $\mathcal{H}$ is not easily calculable.

In the following sections we show that these four conditions ensure that the Aggregation constraint is arc-consistent (§ 3.4) and that a propagation algorithm needs to enforce each condition only once to ensure the consistency of the whole constraint. The propagation algorithm is said to be *idempotent* (§ 3.5).

3.4. Arc-consistency of the Aggregation constraint

For some constraints, when the domain of the variables are restricted to intervals enforcing arc-B-consistency allows to ensure that it is also arc-consistent (Lhomme, 1993). As for the Linear Equality constraint (Apt, 1998), this property is verified for Aggregation.

Theorem 1 (Arc-consistency with respect to the Aggregation constraint). *Let* $\{y\} \bigcup \{u_1, \ldots, u_n\}$ *be a set of variables defined on the* $[0, 1]$ *interval, let* $\mathcal{H}$ *be a strictly increasing continuous aggregation function and* $C =$*Aggregation*$(\mathcal{H}, y, \{u_1, \ldots, u_n\})$ *be an Aggregation constraint. C is arc-consistent if and only if c is arc-B-consistent.*

Proof: Function $\mathcal{H}$ being continuous, considering conditions (AB1) and (AB2) and according to the intermediate values theorem, for all value $v \in [\underline{y}, \overline{y}]$, there exists a set of values $v_1, \ldots, v_n$ in $[\underline{u_1}, \overline{u_1}], \ldots, [\underline{u_n}, \overline{u_n}]$ such that $v = \mathcal{H}(v_1, \ldots, v_n)$. Similarly, considering (AB3) and (AB4), for all set of values $v_k \in [\underline{u_k}, \overline{u_k}]$, we have $\mathcal{H}(u_{-k}, v_k) \geq \underline{y}$. There is therefore a set of values $v, v_1, \ldots, v_{k-1}, v_{k+1}, \ldots, v_n$ in $[\underline{y}, \overline{y}], [\underline{u_1}, \overline{u_1}], \ldots, [\underline{u_{k-1}}, \overline{u_{k-1}}], [\underline{u_{k+1}}, \overline{u_{k+1}}], \ldots, [\underline{u_n}, \overline{u_n}]$ such that $v = \mathcal{H}(v_1, \ldots, v_n)$. □

Again,this theorem may seem trivial since variable domains are limited to intervals. However, it may be violated for non-monotonic functions such as the absolute value of a real number.

3.5. Idempotency of the Aggregation constraint

Maintaining the consistency of the Aggregation constraint implies continuously checking the conditions of Proposition 1 and reducing the variables domains to keep them verified during the search. Hence, virtually any event that reduces the variables domains may trigger the filtering algorithm.

An interesting property is that a single calculation of all bound reductions generated by the consistency conditions of Aggregation is enough to maintain its arc-consistency. Without any event that is external to the constraint, calculating all bounds a second time would not reduce any domain. The constraint is said to be *idempotent* (Apt, 1999). To express this property, the propagation mechanism is often defined as a domain reduction function (Apt, 1999) which, from a set of domains $\{d_1, \ldots, d_m\}$ returns a set of domains $\{d'_1, \ldots, d'_m\}$ deduced by the propagation.

Theorem 2. *(Idempotency of the* Aggregation *constraint) Let* $y, u_1, \ldots, u_n$ *be a set of variables defined on domains* $d_y, d_{u_1}, \ldots, d_{u_n}$. *Let* $\mathcal{F}_{Agg}$ *a domain reduction function that propagates the arc-B-consistency conditions of an* Aggregation *constraint on variables* $y, u_1, \ldots, u_n$. *Function* $\mathcal{F}_{Agg}$ *is idempotent. That is to say:* $\mathcal{F}_{Agg}(\mathcal{F}_{Agg}(d_y, d_{u_1}, \ldots, d_{u_n})) = \mathcal{F}_{Agg}(d_y, d_{u_1}, \ldots, d_{u_n})$. *By extension, the* Aggregation *constraint is said to be idempotent.*

Proof: Aggregation is idempotent iff propagating one arc-B-consistency condition cannot provoke the violation of another. A bound reduction can modify the validity of a condition if it appears in this condition. Let us consider the propagations that can be deduced from Proposition 1:

Increasing of $\underline{y}$ *due to the propagation of (AB1):* $\underline{y}$ appears in (AB3). The propagation being $\underline{y} = \mathcal{H}(\underline{u}), \forall k \in \{1, \ldots, n\}, \mathcal{H}(\overline{u_{-k}}, \underline{u_k}) \geq \mathcal{H}(\underline{u}) = \underline{y}$. Condition (AB3) is true for all

$k \in \{1, \ldots, n\}$. Similarly, the propagation of (AB2) does not perturb the validity of other conditions.

Increasing of $\underline{u_i}$ due to the propagation of (AB3): $\underline{u_i}$ appears in (AB1) and (AB4) for all score u_k, $k \neq i$. Propagating (AB3) for $k = i$ ensure $\underline{y} = \mathcal{H}(\overline{u_{-i}}, \underline{u_i}) \geq \mathcal{H}(\underline{u})$, which validates (AB1). In addition, $\forall k \in \{1, \ldots, n\}, k \neq i$, $\overline{y} \geq \underline{y} = \mathcal{H}(\overline{u_{-i}}, \underline{u_i}) \geq \mathcal{H}(\underline{u_{-k}}, \overline{u_k})$. Condition (AB4) is therefore also true. Similarly, the propagation of (AB4) does not perturb the validity of other conditions.

As a results, checking each of the $2n + 2$ conditions once in any order ensures that all conditions are true. □

Hence, a basic propagation algorithm for an Aggregation constraint checks and propagates each of the $2n + 2$ conditions of Proposition 1 every time an external event modifies a bound of its variables during the search. CP solvers such as Eclair© (Museux et al., 2003) generally trigger a suitable propagation algorithm according to the nature of the event and to the variable on which it happens. The propagation is then said to be *incremental*. Such algorithms have been extensively described in Le Huédé (2003) (in French) for the Aggregation constraint.

4. Application to the propagation of the Choquet integral

In this section we apply the propagation principles identified for a general multicriteria aggregation function $\mathcal{H}$ to the Choquet integral. According to the integration scheme proposed in Section 3, we propose specializing Aggregation to the Choquet constraint.

4.1. The Choquet constraint

Considering n variables $u_1, \ldots, u_n \in [0, 1]$ and a variable $y \in [0, 1]$, our aim is to establish and propagate the relation $y = C_\mu(u_1, \ldots, u_n)$ for a given fuzzy measure μ. Hence, we want to enforce $y = \sum_{i=1}^{n} u_{\sigma(i)}[\mu(A_{\sigma(i)}) - \mu(A_{\sigma(i+1)})]$, where $\sigma(i)$ indicate a permutation on $\mathcal{N}$ such that $u_{\sigma(1)} \leq \cdots \leq u_{\sigma(n)}$, $A_{\sigma(i)} = \{\sigma(i), \ldots, \sigma(n)\}$ and $A_{\sigma(n+1)} = \emptyset$. To maintain this relation, we propose defining the Choquet constraint:

Definition 5 (The Choquet *constraint).* Let $\mathcal{N}$ be a set of n criteria, let $\{y\} \bigcup \{u_1, \ldots, u_n\}$ be a set of variables ranging over [0, 1] and let $\mathcal{M} = \{\mu(\emptyset), \mu(\{1\}), \mu(\{2\}), \ldots, \mu(\{1, \ldots, n\})\}$ be the values of a fuzzy measure μ for each element of $\mathcal{P}(\mathcal{N})$. The Choquet constraint enforces the relation $y = C_\mu(u_1, \ldots, u_n)$ and is denoted choquet$(y, \{u_1, \ldots, u_n\}, \mathcal{M})$.

4.2. Propagating the Choquet constraint

The Choquet integral being continuous and strictly increasing (providing adequate conditions on μ (§ 2.3)), the Choquet constraint belongs to the Aggregation constraint class. As a result, maintaining the arc-B-consistency w.r.t. Choquet can be done by checking the consistency conditions of Proposition 1 with $\mathcal{H} = C_\mu$.

As previously observed in Section 3.3, enforcing conditions (AB1) and (AB2) is equivalent to assigning the domain $[max(\underline{y}, C_\mu(\underline{u})), min(\overline{y}, C_\mu(\overline{u}))]$ to variable y. Expressing a score u_k with respect to the other variables in order to compute its new domain is however more complicated. This implies deducing a function $\mathcal{F}$ from equation $y = C_\mu(u_1, \ldots, u_n)$ such that $u_k = \mathcal{F}(y, u_1, \ldots, u_{k-1}, u_{k+1}, \ldots, u_n)$ without knowing the respective order of the scores.

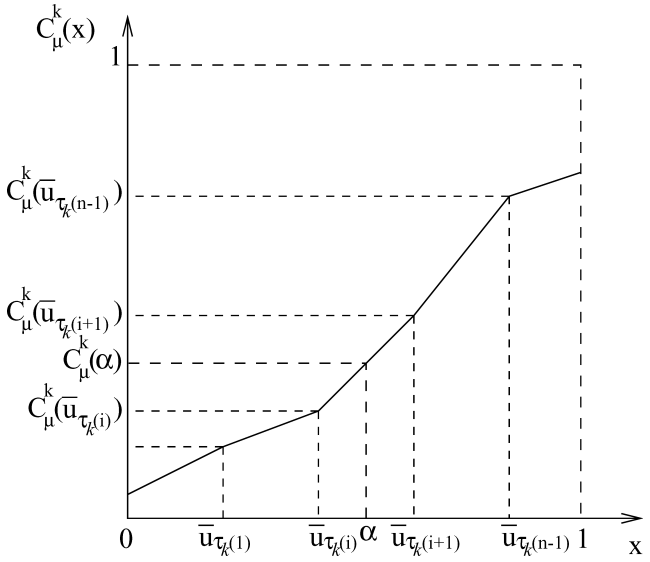


Fig. 3 Graphical representation of $C_\mu^k(x)$

4.3. Calculating the lower bound of a score variable

Assuming condition (AB3) is violated for a given score $u_k, k \in \{1, \ldots, n\}$. We have $C_\mu(\overline{u_1}, \ldots, \overline{u_{k-1}}, \underline{u_k}, \overline{u_{k+1}}, \ldots, \overline{u_n}) < \underline{y}$. To restore the validity of this condition, a new value denoted $\check{u}_k$ must be found for the lower bound $\underline{u_k}$ such that $C_\mu(\overline{u_{-k}}, \check{u}_k) = \underline{y}$. In this equation, the upper bounds $\overline{u_1}, \ldots, \overline{u_{k-1}}, \overline{u_{k+1}}, \ldots, \overline{u_n}$ are actually known, so is their relative order. The only unknown is therefore the relative order of $\check{u}_k$ with respect to these bounds.

In the following paragraph we propose defining algorithm **LB** (i) that first determines the relative order of $\check{u}_k$ with respect to the other bounds and finally returns its value.

Description of function $C_\mu^k(x) = C_\mu(\overline{u_{-k}}, x)$

Notation 2. Let us denote $C_\mu^k : [0, 1] \to [0, 1]$ the function such that $C_\mu^k(x) = C_\mu(\overline{u_{-k}}, x)$. Let τ_k be a permutation on indices $1, \ldots, k-1, k+1, \ldots, n$ such that $\overline{u_{\tau_k(1)}} \leq \cdots \leq \overline{u_{\tau_k(n-1)}}$ and let us denote $\overline{u_{\tau_k(0)}} = 0$ and $\overline{u_{\tau_k(n)}} = 1$. In addition we define the set $A^{-k}_{\tau_k(i)} = \{\tau_k(i), \ldots, \tau_k(n-1)\}$ and we denote $A^{-k}_{\tau_k(n)} = \emptyset$.

Figure 3 shows a possible representation of C_μ^k. It is an increasing piecewise linear function whose vertices correspond to the points where x is equal to $0, \overline{u_{\tau_k(1)}}, \ldots, \overline{u_{\tau_k(n-1)}}, 1$.

Consider a real value $\alpha \in [0, 1]$. We denote $i \in \{0, \ldots, n-1\}$ the index that characterizes the segment on which the point $(\alpha, C_\mu^k(\alpha))$ is located, that is to say, $\alpha \in [\overline{u_{\tau_k(i)}}, \overline{u_{\tau_k(i+1)}}]$. According to the definition of the Choquet integral, function $C_\mu^k(\alpha)$ can be written as follows: $\forall \alpha \in [0, 1], \exists i \in \{0, \ldots, n-1\}$ such that $\alpha \in [\overline{u_{\tau_k(i)}}, \overline{u_{\tau_k(i+1)}}]$:

$$\begin{aligned} C_\mu^k(\alpha) = & \sum_{j=1}^{i} \overline{u_{\tau_k(j)}} \times \left[\mu\left(A^{-k}_{\tau_k(j)} \cup \{k\}\right) - \mu\left(A^{-k}_{\tau_k(j+1)} \cup \{k\}\right)\right] \\ & + \alpha \times \left[\mu(A^{-k}_{\tau_k(i+1)} \cup \{k\}\right) - \mu\left(A^{-k}_{\tau_k(i+1)}\right)\Big] \\ & + \sum_{j=i+2}^{n} \overline{u_{\tau_k(j-1)}} \times \left[\mu\left(A^{-k}_{\tau_k(j-1)}\right) - \mu\left(A^{-k}_{\tau_k(j)}\right)\right] \end{aligned} \tag{3}$$

Hence, to calculate $\check{u}_k$ such that $C_\mu^k(\check{u}_k) = \underline{y}$, we first have to determine on which linear part of the curve the point $(\check{u}_k, \underline{y})$ is located. This is equivalent to finding the index $i^\star$ such that $C_\mu^k(\overline{u_{\tau_k(i^\star)}}) < \underline{y} \leq C_\mu^k(\overline{u_{\tau_k(i^\star+1)}})$. Calculating $\check{u}_k$ can then be done with respect to point $(\overline{u_{\tau_k(i^\star)}}, C_\mu^k(\overline{u_{\tau_k(i^\star)}}))$ and to the slope of this linear part.

Locating lower bound $\check{u}_k$

As stated before, we search for $i^\star \in \{0, \ldots, n-1\}$, such that: $C_\mu^k(\overline{u_{\tau_k(i^\star)}}) < \underline{y} \leq C_\mu^k(\overline{u_{\tau_k(i^\star+1)}})$.

A first remark is that $\check{u}_k \geq \underline{u_k}$ and $\check{u}_k \leq \overline{u_k}$ ($C_\mu^k(\overline{u_k}) \geq \underline{y}$). In addition, $\forall k \in \{1, \ldots, n\}$, $C_\mu(\overline{u}) = C_\mu^k(\overline{u_k})$. Considering that the lower bounds of all scores will be calculated successively, we propose to explore function C_μ^k from edge to edge, starting from the already calculated point $(\overline{u_k}, C_\mu(\overline{u}))$ (which belongs to C_μ^k for all $k \in \{1, \ldots, n\}$), until $i^\star$ is found.

In order to avoid multiple calculations of the Choquet integral during this exploration, we use the following transition formula that allows going from a point with index i to a point with index $i-1$:

Proposition 2. *(Transition formula for function C_μ^k) Consider an index $i \in \{1, \ldots, n\}$. The value of C_μ^k at $\overline{u_{\tau_k(i-1)}}$ can be calculated with respect to $\overline{u_{\tau_k(i)}}$ and $C_\mu^k(\overline{u_{\tau_k(i)}})$ according to the following transition formula:*

$$C_\mu^k(\overline{u_{\tau_k(i-1)}}) = C_\mu^k(\overline{u_{\tau_k(i)}}) - (\overline{u_{\tau_k(i)}} - \overline{u_{\tau_k(i-1)}})\big(\mu\big(A_{\tau_k(i)}^{-k} \cup \{k\}\big) - \mu\big(A_{\tau_k(i)}^{-k}\big)\big) \tag{4}$$

Proof: If we express $C_\mu^k(\overline{u_{\tau_k(i)}})$ and $C_\mu^k(\overline{u_{\tau_k(i-1)}})$ according to Equation (3) and remove common terms, we have:

$$\begin{aligned}
C_\mu^k(\overline{u_{\tau_k(i)}}) - C_\mu^k(\overline{u_{\tau_k(i-1)}}) &= \overline{u_{\tau_k(i)}}\big[\mu\big(A_{\tau_k(i)}^{-k} \cup \{k\}\big) - \mu\big(A_{\tau_k(i+1)}^{-k} \cup \{k\}\big)\\
&\quad + \overline{u_{\tau_k(i)}}\big[\mu\big(A_{\tau_k(i+1)}^{-k} \cup \{k\}\big) - \mu\big(A_{\tau_k(i+1)}^{-k}\big)\big]\\
&\quad - \overline{u_{\tau_k(i-1)}}\big[\mu\big(A_{\tau_k(i)}^{-k} \cup \{k\}\big) - \mu\big(A_{\tau_k(i)}^{-k}\big)\big]\\
&\quad - \overline{u_{\tau_k(i)}}\big[\mu\big(A_{\tau_k(i)}^{-k}\big) - \mu\big(A_{\tau_k(i+1)}^{-k}\big)\big]\\
&= \overline{u_{\tau_k(i)}}\big[\mu\big(A_{\tau_k(i)}^{-k} \cup \{k\}\big) - \mu\big(A_{\tau_k(i)}^{-k}\big)\big]\\
&\quad - \overline{u_{\tau_k(i-1)}}\big[\mu\big(A_{\tau_k(i)}^{-k} \cup \{k\}\big) - \mu\big(A_{\tau_k(i)}^{-k}\big)\big]\\
&= (\overline{u_{\tau_k(i)}} - \overline{u_{\tau_k(i-1)}}) \times \big[\mu\big(A_{\tau_k(i)}^{-k} \cup \{k\}\big) - \mu\big(A_{\tau_k(i)}^{-k}\big)\big]
\end{aligned}$$

□

Remark 3. Let τ be a permutation of indices $1, \ldots, n$ such that $\overline{u_{\tau(1)}} \leq \cdots \leq \overline{u_{\tau(n)}}$. Let $r_k = \tau^{-1}(k)$ be the rank of $\overline{u_k}$ in the sorted set of scores upper bounds. Then we have $\overline{u_k} \in \big[\overline{u_{\tau_k(r_k-1)}}, \overline{u_{\tau_k(r_k)}}\big]$. If the algorithm starts the exploration of $C_\mu^k(x)$ from $(\overline{u_k}, C_\mu(\overline{u}))$, according to (4), the first edge below $(\overline{u_k}, C_\mu(\overline{u}))$ on the curve of function $C_\mu^k(x)$ takes the

following value on the y-axis:

$$C_\mu^k(\overline{u_{\tau_k(r_k-1)}}) = C_\mu(\overline{u}) - (\overline{u_k} - \overline{u_{\tau_k(r_k-1)}}) \times \left[\mu(A^{-k}_{\tau_k(r_k)} \cup \{k\}) - \mu(A^{-k}_{\tau_k(r_k)})\right]$$

In addition, for all $j \in \{1, \ldots, r_k - 1\}$, $\tau_k(j) = \tau(j)$. Knowing $\check{u}_k \leq \overline{u_k}$, in a more general propagation algorithm, permutation τ can be used for all lower bound calculations instead of the n permutations $\tau_1, \ldots, \tau_n$.

Calculating lower bound $\check{u}_k$

Finally, determining $i^\star$ allows us to deduce $\check{u}_k$:

$$\check{u}_k = \overline{u_{\tau_k(i^\star)}} + \frac{\underline{y} - C_\mu^k(\overline{u_{\tau_k(i^\star)}})}{\mu\left(A^{-k}_{\tau_k(i^\star+1)} \cup \{k\}\right) - \mu\left(A^{-k}_{\tau_k(i^\star+1)}\right)} \tag{5}$$

Proof: Since values $\check{u}_k$ and $\overline{u_{\tau_k(i^\star)}}$ both belong to $[\overline{u_{\tau_k(i^\star)}}, \overline{u_{\tau_k(i^\star+1)}}]$, according to (4):

$$C_\mu^k(\check{u}_k) - C_\mu^k(\overline{u_{\tau_k(i^\star)}}) = (\check{u}_k - \overline{u_{\tau_k(i^\star)}}) \times \left[\mu\left(A^{-k}_{\tau_k(i^\star+1)} \cup \{k\}\right) - \mu\left(A^{-k}_{\tau_k(i^\star+1)}\right)\right]$$

Replacing $C_\mu^k(\check{u}_k)$ by $\underline{y}$ in this equation results in equation (5). As the fuzzy measure is assumed to be strictly monotonic, no division by zero can be obtained. □

Algorithm

In the previous paragraph we detailed the steps of the determination of the lower bound of a score u_k, $k \in \{1, \ldots, n\}$ in order to maintain the arc-B-consistency condition (AB3) of Proposition 1. Figure 4 describes the **LB**(k) algorithm, which implements the proposed approach (a similar algorithm **UB**(k) is used to compute the upper bound of u_k).

The main loop of this algorithm locates the lower bound $\check{u}_k$. It applies the transition formula (4) by decreasing variable $i^\star$ until the reached point is either lower or equal to $\underline{y}$ on the y-axis or lower to $\underline{u_k}$ on the x-axis.

Finally, when $\check{u}_k \in [\overline{u_{\tau(i^\star)}}, \overline{u_{\tau(i^\star+1)}}]$ with $\overline{u_{\tau(i^\star+1)}} < \underline{u_k}$, the algorithm returns 0 and does not trigger any propagation. Otherwise, the value $\check{u}_k$ is calculated according to (5).

The main loop in this algorithm represents at most a set of n constant time operations. The complexity of this propagation algorithm is therefore $O(n)$. As a result, propagating every conditions of the constraint has a complexity of $O(n^2)$.

4.4. An example of propagations generated by the Choquet constraint

Let y, u_1, u_2, u_3 be four variables and let $\mathcal{M} = \{\mu_0, \mu_1, \ldots, \mu_{123}\}$ be a fuzzy measure such that: $y \in [0.4, 1]$, $u_1 \in [0, 0.2]$, $u_2 \in [0, 0.8]$, $u_3 \in [0, 0.2]$ and $\mu_0 = 0$, $\mu_1 = 0.1$, $\mu_2 = 0.4$, $\mu_3 = 0.1$, $\mu_{12} = 0.5$, $\mu_{13} = 0.2$, $\mu_{23} = 0.6$, $\mu_{123} = 1$. If we set the constraint choquet(y, $\{u_1, u_2, u_3\}$, $\mathcal{M}$), we obtain the following propagations:

On the y variable, $C_\mu(0, 0, 0) = 0$ and $C_\mu(0.2, 0.8, 0.2) = 0.2 \times (1 - 0.6) + 0.2 \times (0.6 - 0.4) + 0.8 \times 0.4 = 0.44$. Therefore from (AB2) (Proposition 1) we can conclude that $y \in [0.4, 0.44]$.

Fig. 4 Calculation algorithm for the lower bound of a score u_k

```
LB(k) : float
    i* ← τ^{-1}(k)
    u_{i*} ← ū_k
    ŭ_k ← 0
    Cu_{i*} ← C_μ(ū)
    A^{-k}_{τ(i*)} ← {τ(i*+1), ..., τ(n)}
    while ((Cu_{i*} > y)&(u_{i*} > u_k))
        p ← μ(A^{-k}_{τ(i*)} ∪ {k}) − μ(A^{-k}_{τ(i*)})
        Cu_{i*} ← Cu_{i*} − p × (u_{i*} − ū_{τ(i*−1)})
        u_{i*} ← ū_{τ(i*−1)}
        A^{-k}_{τ(i*)} ← A^{-k}_{τ(i*)} ∪ τ(i* − 1)
        i* ← i* − 1
    endwhile
    if (Cu_{i*} = y)
        ŭ_k ← u_{i*}
    else if (Cu_{i*} < y)
            ŭ_k ← u_{i*} + (y − Cu_{i*}) / p
        endif
    endif
    return ŭ_k
end
```

On score u_1: Let $\breve{u}_1$ be the lower bound of u_1 that can be deduced from (AB3), i.e., such that $C_\mu(\breve{u}_1, 0.8, 0.2) = 0.4$. We have already calculated $C_\mu(0.2, 0.8, 0.2) = 0.44$. Therefore we can conclude that $\breve{u}_1 \in [0, 0.2)$ and from Equation (5): $\breve{u}_1 = \frac{0.4 - C_\mu(0,0.8,0.2)}{\mu_{123} - \mu_{23}} = \frac{0.4 - 0.36}{0.4} = 0.1$. Considering the calculation of an upper bound for u_1, we can notice that we have reduced $\overline{y}$ such that $\overline{y} = C_\mu(\overline{u_1}, \overline{u_2}, \overline{u_3})$. It follows that $\overline{y} \geq C_\mu(\overline{u_1}, \underline{u_2}, \underline{u_3})$. Therefore condition (AB4) is verified and we can conclude that $\overline{u_1}$ will not be reduced.

If we follow the same reasoning for u_2 and u_3, we finally obtain: $u_1 \in [0.1, 0.2]$, $u_2 \in [0.7, 0.8]$ and $u_3 \in [0.12, 0.2]$.

5. Experimentation

The propagation of the Choquet constraint has been evaluated on small instances of the multi-criteria examination timetabling problem.

5.1. The examination timetabling problem

Given a set of examinations, a set of students each enrolled for a given list of examinations, a set of rooms of fixed capacities and a set of periods, the examination timetabling problem consists in assigning a period and a room to each examination such that (i) two examinations that are given to a same student cannot be planned on the same period and (ii) the capacity of a room cannot be exceeded. We assume that as long as (i) and (ii) hold, several examinations can take place in the same room at the same time but that the number of students attending an examination cannot be distributed over several rooms.

A simple multi-criteria model has been constructed based on three attributes: the date of the last examination planned (criterion *duration of the examination*), the number of rooms used (criterion *rooms employment*) and the number of times a student has two consecutive examinations (criterion *spreading of the exams*). These criteria are aggregated using the Choquet integral (the whole multi-criteria model is more precisely described in Le Huédé (2003)).

5.2. Instances

Small scenarios have been constructed in order to evaluate the performance of the propagation in complete search. The main characteristics of these scenarios are described in the following table:

	Number of periods	Number of exams	Number of rooms	Number of students
Sc. 12	9	12	2	49
Sc. 15	9	15	3	56
Sc. 20	11	20	2	104
Sc. 30	13	30	2	161

The algorithms are launched for various fuzzy measures that correspond to typical cases of aggregation functions. The μ_{min} measure models an intolerant expert (i.e., complementary criteria), μ_{max} models a tolerant expert and μ_{mean} models the case where criteria are independents. Although μ_{min}, μ_{max} and μ_{mean} are respectively close to the min, max and the mean functions, they are not exactly equal to these functions.

5.3. Results

To evaluate the effect of propagation we use a simple propagation from the scores u_1, u_2, u_3 to the overall evaluation y (basically, only (AB1) and (AB2) are propagated). This strategy is denoted B&B. It is compared to a solving where arc-B-consistency is enforced at each node of the search tree (denoted A-B-C). In order to allow objective comparisons, we use a static chronological labeling strategy. The variables that model the date of the exams are ordered according to the number of disjunctions between exams and the number of enrolments for each exam. The algorithms are compared according to the number of backtracks and the completion time needed to optimally solve each problem.

	μ_{min}				μ_{mean}				μ_{max}			
	B&B		A-B-C		B&B		A-B-C		B&B		A-B-C	
	Btk	s	Btk	s	Btk	s	Btk	s	Btk	s	Btk	s
Sc. 12	1083	0.08	308	0.04	732	0.06	215	0.02	392	0.04	85	0.02
Sc. 15	22271	1.88	3401	0.47	237783	21.87	93289	9.13	67710	5.53	12228	1.45
Sc. 20	738013	72.65	49832	9.45	1357203	135.57	182540	33.45	349852	35.54	35011	4.77
Sc. 30	35.10^6	1:02:13	$7.7.10^6$	0:22:55	190.10^6	6:08:07	53.10^6	1:45:25	-	-	-	-

Time results are indicated either in seconds or in the h:mn:s format. The "-" symbol means that the problem couldn't be solved within 48 hours.

These results clearly show that, for any kind of fuzzy measure, propagating the Choquet constraint considerably reduces the size of the tree that is constructed by the search. Although the time effort is $O(n^2)$ for the Choquet constraint with respect to $O(n)$ for the straightforward calculation, the solving time is also considerably reduced for all measures and instances.

In addition, when the min, mean and max functions are modeled with the Choquet integral, the time used for the propagation of Choquet can be compared to the time taken by the min, sum and max constraints. Times obtained with the solver constraints (S-C) are compared with the others in the following table:

	min			mean			max		
	B&B	A-B-C	S-C	B&B	A-B-C	S-C	B&B	A-B-C	S-C
Sc. 12	0.03	0.01	0.01	0.06	0.03	0.04	0	0	0
Sc. 15	1.4	0.234	0.219	2.37	1.03	0.89	0	0	0
Sc. 20	63.5	5.48	5.17	85.7	21.44	19.45	0	0	0
Sc. 30	0:37:53	0:09:31	0:09:16	1:36:24	0:46:31	0:44:52	-	26:28:42	25:24:41

As previously observed, A-B-C is clearly better than B&B. Comparing A-B-C and S-C, we can see that the difference between the "generic" Choquet algorithms and the dedicated algorithms of each constraint remains quite small. Hence, the expressivity gained with Choquet does not generate a great loss in efficiency.

In addition, in this table we can see that in the max case, an optimal solution is immediately found by the solver, except for scenario 30 which takes very long (B&B failed to solve the problem within 2 days). In this case, the heuristics directly finds a solution that is equal to 1 on criterion "duration" (and thus is optimum for max). As no such solution exists in Sc. 30, a very large part of the search tree is explored before a solution that completely satisfies the "spreading" criterion is found. This illustrates well another difficult problem, which is the definition of search heuristics when the objective depends on several criteria.

6. Conclusion

In this paper we introduced a new approach for the solving of multi-criteria optimization problems in CP. As the elicitation of multi-criteria preferences is a very hard problem on its own, we proposed to integrate a very general model from MCDM and to benefit from the well proven methodology that has already been developed for the elicitation of its parameters.

To achieve this integration we proposed a general scheme based on the definition of a multi-criteria aggregation constraint that establishes the performance of a solution. According to general properties of multi-criteria aggregation functions, we defined some arc-B-consistency conditions with respect to constraints that model these functions. We showed that if the constraint is arc-B-consistent, then it is necessarily arc-consistent and that an algorithm that propagates once each of the consistency conditions is idempotent. These principles were applied to the case of the Choquet integral, which is a very flexible aggregation function that offers good properties for modeling preferences. We defined the Choquet global constraint and proposed propagation algorithms for modeling this function in a CP solver. This constraint has been evaluated on the multi-criteria timetabling problem. The results obtained on the optimal resolution of small instances for different kind of preference models clearly show the interest of propagation.

A first observation is that the aggregation of criteria finally enables reducing the problem to a mono-criterion optimization problem. However, the problem keeps an underlying multi-criteria structure that, in particular, considerably increases the complexity of defining good search heuristics. Some work has been led on this issue and deserves further investigations (Le Huédé et al., 2003).

References

Apt, K.R. (1998). "A Proof Theoretic View of Constraint Programming." *Fundamenta Informaticae* 34(3), 295–321.

Apt, K.R. (1999). "The Essence of Constraint Propagation." *Theoretical Computer Science* 221(1–2), 179–210.

Bana e Costa, C.A. and J.C. Vansnick. (1994). "A Theoretical Framework for Measuring Attractiveness by a Categorical Based Evaluation TecHnique (MACBETH)." In *Proc. XIth Int. Conf. on MCDM*, Coimbra, Portugal, pp. 15–24.

Barichard, V. and J.-K. Hao. (2003). "A Population and Interval Constraint Propagation Algorithm." In *LNCS 2632*, Springer, pp. 88–101.

Bistarelli, S., U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. (1999). "Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison." *CONSTRAINTS* 4, 199–240.

Boutilier, C., R. Brafman, C. Domshlak, H.H. Hoos, and D. Poole. (2004). "CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements." *Journal of Artificial Intelligence Research* 21, 135–191.

Boutilier, C., R. Brafman, C. Domshlak, H.H. Hoos, and D. Poole. (2004). "Preference-Based Constrained Optimization with CP-nets." *Computational Intelligence* 20(2), 137–157.

Choquet, G. (1953). "Theory of Capacities." *Annales de l'Institut Fourier* 5, 131–295.

Gavanelli, M. (2002). "An Algorithm for Multi-Criteria Optimization in CSPs." In Frank van Harmelen, editor, *Proceedings of ECAI-2002*, Lyon, France, IOS Press, pp. 136–140.

Grabisch, M. (1996). "The Application of Fuzzy Integrals in Multicriteria Decision Making." *European J. of Operational Research* 89, 445–456.

Grabisch, M. (2000). "The Interaction and Möbius Representations of Fuzzy Measures on Finite Spaces, k-Additive Measures: A Survey." In M. Grabisch, T. Murofushi, and M. Sugeno, editors, *Fuzzy Measures and Integrals—Theory and Applications*, Physica Verlag, pp. 70–93.

Grabisch, M. and M. Roubens. (2000). "Application of the Choquet Integral in Multicriteria Decision Making." In M. Grabisch, T. Murofushi, and M. Sugeno, editors, *Fuzzy Measures and Integrals—Theory and Applications*, Physica Verlag, pp. 348–374.

Ben Jaâfar, I., N. Khayati, and K. Ghédira. (2004). "Multicriteria Optimization in CSPs: Foundations and Distributed Solving Approach." In Christoph Bussler and Dieter Fensel, editors, *AIMSA*, volume 3192 of *LNCS*, Springer, pp. 459–468.

Junker, U. (2004). "Preference-Based Search and Multi-Criteria Optimization." *Annals of OR* 130, 75–115.

Kaymak, U. and J.M. Sousa. (2003). "Weighted Constraint Aggregation in Fuzzy Optimization." *Constraints* 8(1), 61–78, Jan.

Keeney, R.L. and H. Raiffa. (1976). *Decision with Multiple Objectives*. Wiley, New York.

Labreuche, C. and M. Grabisch. (2003). "The Choquet Integral for the aggregation of Interval Scales in Multicriteria Decision making." *Fuzzy Sets and Systems* 137(1), 11–26.

Labreuche, C. and F. Le Huédé. (2005). "MYRIAD: a Tool Suite for MCDA." In *EUSFLAT'05*, Barcelona, Spain.

Le Huédé, F. (2003). *Intégration d'un modèle d'Aide à la Décision Multicritère en Programmation Par Contraintes*. PhD thesis, Universitée Paris 6.

Le Huédé, F., M. Grabisch, C. Labreuche, and P. Savéant. (2003). "Multicriteria Search in Constraint Programming." In *Proceedings of CP-AI-OR'03*, Montréal, Canada, pp. 291–304.

Lhomme, O. (1993). "Consistency Techniques for Numerical CSPs." In *Proceedings of IJCAI 1993*, Chambery, France, pp. 232–238.

Marichal, J.L. (1998). *Aggregation Operators for Multicriteria Decision Aid*. PhD thesis, University of Liège.

Michel, C., M. Rueher, and Y. Lebbah. (2001). "Solving Constraints Over Floationg-Point Numbers." In *Proceedings of CP-2001*, Springer.

Museux, N., L. Jeannin, P. Savéant, F. Le Huédé, F.-X. Josset, and J. Mattioli. (2003). "Claire/Eclair: Un environnement de modélisation et de résolution pour des applications d'optimisation combinatoires embarquées." In *Proceedings of JFPLC'03*, Amiens, France.

Prestwich, S., F. Rossi, K.B. Venable, and T. Walsh. (2005). "Constraint-Based Preferential Optimization." In *AAAI*, pp. 461–466.

Pearl Pu and Boi Faltings. (2004). "Decision Tradeoff using Example-Critiquing and Constraint Programming." *Constraints* 9(4), 289–310.

Refalo, P. (1999). "Tight Cooperation and its Application in Piecewise Linear Optimisation." In *Proceedings of CP-99*, Springer, pp. 375–389.

Sugeno, M. (1974). *Theory of fuzzy integrals and its applications*, PhD thesis, Tokyo Institute of Technology.

Zhang, Y. and R. H.C. Yap. (2000). "Arc Consistency on n-ary Monotonic and Linear Constraints." In *Proceedings of CP-2000*, Singapore, pp. 470–483.

TRANSPORTATION SCIENCE
Vol. 47, No. 3, August 2013, pp. 344–355
ISSN 0041-1655 (print) | ISSN 1526-5447 (online)

http://dx.doi.org/10.1287/trsc.1120.0432

An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Transfers

Renaud Masson, Fabien Lehuédé, Olivier Péton

LUNAM Université, École des Mines de Nantes, IRCCyN UMR CNRS 6597, F-44307 Nantes cedex 3, France
{renaud.masson@mines-nantes.fr, fabien.lehuede@mines-nantes.fr, olivier.peton@mines-nantes.fr}

The pickup and delivery problem (PDP) consists in defining a set of routes that satisfy transportation requests between a set of pickup points and a set of delivery points. This paper addresses a variant of the PDP where requests can change vehicle during their trip. The transfer is made at specific locations called "transfer points." The corresponding problem is called the pickup and delivery problem with transfers (PDPT). Solving the PDPT leads to new modeling and algorithmic difficulties. We propose new heuristics capable of efficiently inserting requests through transfer points. These heuristics are embedded into an adaptive large neighborhood search. We evaluate the method on generated instances and apply it to the transportation of people with disabilities. On these real-life instances we show that the introduction of transfer points can bring significant improvements (up to 9%) to the value of the objective function.

Key words: pickup and delivery problem; transfers; adaptive large neighborhood search
History: Received: March 2011; revision received: January 2012; accepted: May 2012. Published online in *Articles in Advance* September 5, 2012.

1. Introduction

The pickup and delivery problem (PDP) consists in defining a set of minimum cost routes in order to satisfy a set of transportation requests. Each transportation request has a known origin called the *pickup point* and a destination called the *delivery point*. Several users can share a vehicle as long as its capacity is not exceeded. The PDP is a well-known extension of the vehicle routing problem (VRP). It can be expressed either in a static version, where each request is known in advance, or in a dynamic version, where new requests may be integrated at any time (Berbeglia, Cordeau, and Laporte 2010). If time windows on requests are considered, the problem is called the pickup and delivery problem with time windows (PDPTW). Requests can correspond to the transportation of goods or people. Depending on the context of the application, various objective functions may be considered: number of vehicles used, total ride time, total distance, cost or quality of service-related criteria. In what follows, we consider a static case problem related to the transportation of people with disabilities. The objective function is to minimize the total distance traveled.

This paper focuses on a recent extension of the PDPTW where requests can be transferred from one vehicle to another at intermediate points between the pickup and delivery points. In this case, a first vehicle drives the request to a predetermined location called *transfer point*. A second vehicle picks up the request at the transfer point and drives it to the delivery point. This extension is called the pickup and delivery problem with transfers (PDPT). In the PDPT, the implicit constraint stating that the pickup and delivery points of a given request should be serviced by the same vehicle is relaxed. New precedence constraints state that if a request uses a transfer point, it must be delivered at the transfer point by a first vehicle before being picked up by a second vehicle. Any pickup and delivery problem can be viewed as a PDPT with no transfer point, so the PDPT is NP-hard.

The PDP has been intensively studied over recent decades, Berbeglia et al. (2007) and Parragh, Doerner, and Hartl (2008) present detailed reviews. On the other hand, the PDPT as devised in this article has been the subject of very few works. Mitrović-Minić and Laporte (2006) propose a local search method for the uncapacitated PDPT with a Manhattan distance. They solve generated instances with up to 100 requests. This method was adapted in a tabu search by Masson, Lehuédé, and Péton (2011) for the dial-a-ride problem with transfers. In this study, the authors underline the complexity of enforcing the maximum ride time constraint. A comprehensive mathematical formulation of the PDPT is proposed by Cortés, Matamala, and Contardo (2010). The authors use a branch-and-cut algorithm for solving instances with six requests and two vehicles to optimality.

Regarding related exact approaches, Lin (2008) presents a PDPTW where all requests share the same

delivery location. Nevertheless, the delivery time windows are different. In this problem, it is considered that a transfer can occur on the last pickup before a delivery. Instances with up to 100 requests are solved to optimality using a commercial solver. Kerivin et al. (2008) consider a PDP where every request can be split as well as transferred from one vehicle to another at every node of the problem. This problem has no time window and is solved using branch and cut. Some instances with up to 15 requests are solved to optimality. Nakao and Nagamochi (2010) calculate a lower bound for the PDPT with a single transfer point. Transfers also appear in the school bus model of Fugenschuh (2009). In this work, the routes are already designed and the objective is to minimize the number of buses needed. Some passengers can be transferred from one bus to another but the problem is very different.

The use of transfer points has also been considered, for solving pickup and delivery problems with particular restrictions. This possibility was first raised by Shang and Cuff (1996). These authors consider that any point can be used as a transfer point. Transfers are only considered to insert a request that cannot be inserted in the current solution without resorting to an extra vehicle. A heuristic construction relies on the construction of miniroutes that are assigned to vehicles. Thangiah, Fergany, and Awam (2007) use the same principles in a real-time version of the problem. Oertel (2000) considers two variants where all the points or only a subset of locations can be considered as transfer points. A heuristic method and a column generation heuristic are presented. Gørtz, Nagarajan, and Ravi (2009) consider a version of the PDPT where the objective is to minimize the makespan. Heuristic methods are proposed to solve the uncapacitated and capacitated cases. A heuristic column generation method is proposed by Mues and Pickl (2005). They consider a problem with a single transfer point through which requests are systematically routed. The set of routes is composed of pickup routes ending at the transfer point and delivery routes starting at this transfer point. Instances with up to 70 requests are considered for experiments. Cortés and Jayakrishnan (2002) present a feasibility simulation for a demand-responsive transit system allowing a single transfer per passenger. Petersen and Ropke (2011) present another ALNS to handle freight transportation problems considering one transfer point and up to 982 requests.

The application that motivates this work is a demand responsive transport problem for people with disabilities. This concerns people who require daily trips from their home to schools, social centers, or vocational rehabilitation centers. We minimize the total distance traveled, which may deteriorate the quality of service provided to users. Therefore, time windows on pickup and delivery points have to be carefully chosen in order to ensure the desired quality of service. Without loss of generality, we consider outbound trips from home to a destination. In this case, the number of pickup points is significantly larger than the number of delivery points. Transfers correspond to a practice followed in some vocational rehabilitation centers that aims to reduce transportation costs. People that live in the same geographical area are picked up by one vehicle, independently of their destination. The passengers are delivered at the closest school or center where they spread in other vehicles that drive them to their final destination. However, the transfer operation may also have a negative impact on the quality of service because direct trips are generally preferred by passengers. Thus, the decision makers have to balance the expected benefits and drawbacks of resorting to transfers.

In this article, we focus on designing a method capable of efficiently exploring the search space of the problem. Therefore, a few practical constraints are relaxed. We consider a homogeneous fleet of vehicles and ignore the maximum ride-time constraint. In the instances considered, ride time is limited because of adequate time windows. We assume that the passengers take two vehicles at most and thus only one transfer per trip is authorized. Moreover, no fixed cost is associated with the use of a vehicle or with the transfer operation.

We propose new heuristics capable of efficiently inserting requests through transfer points. These heuristics are embedded into an adaptive large neighborhood search (ALNS). We evaluate the method on generated instances and apply it to the transportation of people with disabilities. One purpose of this study is to evaluate the potential output of transfers. To do so we compare solutions of the ALNS with and without considering transfer points. To provide a lower bound on the savings that can be achieved thanks to the use of transfer, we also compare upper bounds of the PDPT solutions with lower bounds of PDPTW solutions.

The remainder of this paper is organized as follows. Section 2 presents a formulation of the PDPT. Sections 3 and 4 are devoted to the solution method for the problem without and with transfer points, respectively. Computational results are presented in §5, followed by the conclusion in §6.

2. Problem Formulation

In this section, we give a description of the PDPT. We consider a set of n requests, $R = \{1, \ldots, n\}$, a set T of transfer points, and a set K of homogeneous vehicles of capacity Q. The pickup and delivery nodes of the request $i \in R$ are designated by $p(i)$

and $d(i)$. The sets of all pickup and delivery points are denoted P and D; $o(k)$ and $o'(k)$ represent the starting and the ending depot of route $k \in K$, and the set of all depot locations is represented by O. The PDPT is defined on a complete directed graph $G = (V, A)$, where $V = P \cup D \cup O \cup T$ represents the set of all vertices and $A = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$ the set of all arcs. With each arc $(v_i, v_j) \in A$ is associated a nonnegative travel time $l_{i,j}$. A time window $[e_i, l_i]$ is associated with each vertex $v_i \in V$, where e_i and l_i represent the earliest and the latest time at which the service can begin. Each vertex has a known service time s_i representing the time needed to get users in or out of the vehicle. We consider that a request i concerns q_i passengers who have the same pickup and delivery nodes. At any moment, the number of users carried simultaneously by a vehicle cannot exceed Q. In addition, the service at each vertex j should begin in the interval $[e_j, l_j]$. A vehicle is allowed to wait at a vertex in order to service it within its time windows.

A solution of the PDPT is a set of $|K|$ routes that satisfies all the requests and such that route k starts at $o(k)$ and ends at $o'(k)$. For every request $i \in R$, vertices $p(i)$ and $d(i)$ can be served by the same route, $d(i)$ being serviced after $p(i)$. Vertices $p(i)$ and $d(i)$ can also be served by distinct routes $k_1 \in K$ and $k_2 \in K$. In this case, k_1 and k_2 both have to service one transfer point v_j such that

(i) v_j is serviced after $p(i)$ in route k_1;

(ii) v_j is serviced before $d(i)$ in route k_2; and

(iii) v_j is serviced in route k_1 before being serviced in route k_2.

A mixed integer linear program of the PDPT has been proposed by Cortés, Matamala, and Contardo (2010). Because this model contains 31 sets of constraints, we do not reproduce it here.

3. ALNS for the PDPTW

In this section, we describe the ALNS developed for this paper. The ALNS has been described extensively by Pisinger and Ropke (2007) and applied to the PDPTW by Ropke and Pisinger (2006). We use the same approach to solve the PDPTW as a basis for comparison with the PDPT. It also provides a basic framework in which the specific developments for the PDPT, described in §4, are integrated.

3.1. Main Scheme of the Large Neighborhood Search (LNS)

The large neighborhood search (LNS) was introduced by Shaw (1998) in a constraint programming framework to solve the vehicle routing problem with time windows (VRPTW). The LNS is similar to the *ruin and recreate* introduced by Schrimpf et al. (2000). An extensive description of the method and of its application to combinatorial optimization problems can be found in the recent review by Ropke and Pisinger (2010). The underlying principle of the LNS is to destroy and repair a solution iteratively in order to improve it. The general functioning of the LNS is depicted in Algorithm 1.

Algorithm 1 (LNS)

Require: *InitialSolution*

1: *BestSolution* ← *InitialSolution*
2: *CurrentSolution* ← *InitialSolution*
3: **while** the termination criterion is not satisfied **do**
4: Selection of *Destroy* and *Repair* heuristics
5: $S \leftarrow$ *CurrentSolution*
6: $S \leftarrow Destroy(S)$
7: $S \leftarrow Repair(S)$
8: **if** $S <$ *BestSolution* **then**
9: *BestSolution* ← S
10: *CurrentSolution* ← S
11: **else**
12: **if** *TransitionAccepted*(S, *CurrentSolution*) **then**
13: *CurrentSolution* ← S
14: **end if**
15: **end if**
16:
17: **end while**
18: **return** *BestSolution*

The latest LNS methods integrate several heuristics for destroying and repairing the current solution (lines 6 and 7); they accept deterioration of the objective during the search (line 12), using, e.g., a simulated annealing acceptance criterion (Kirkpatrick, Gelatt, and Vecchi 1983) or a record-to-record travel (RRT) algorithm (Laporte, Musmanno, and Vocaturo 2010). Ropke and Pisinger (2006) present an ALNS for the PDPTW. The search is called adaptive because the probability of choosing each destroy and repair method is reevaluated periodically depending on their efficiency in the past iterations. Its main components are reviewed in the following subsections.

At each iteration, r% of the requests are removed from the current solution. Requests are reinserted in the new solution using repair heuristics until all requests have been planned or no feasible insertion can be found. Partial solutions can be used as intermediate unfeasible solutions in the algorithm, unrouted requests are penalized in the objective function or placed in a so-called request bank.

3.2. Destruction Neighborhoods

We implemented four destruction heuristics:

1. *Worst removal*. This heuristic first computes the cost savings produced by the removal of each request; r% of the requests are then selected randomly to be removed. The probability of being selected increases with the savings.

2. *Random removal*. This heuristic randomly selects r% of the requests to be removed from the current solution.

3. *Related removal*. This heuristic aims at removing related nodes. The relatedness measure of two requests (Shaw 1998) depends on the distance between their pickups and their deliveries, the difference between the time of service at their nodes and the difference in load of the two requests. Complete description of this operator is given by Ropke and Pisinger (2006).

4. *History removal*. This heuristic is inspired from the two history removal heuristics presented by Pisinger and Ropke (2007). It aims to remove the requests that seem poorly placed in the current solution with regard to the best-known solutions. For each node j and j' of the problem, $\xi_{j,j'}$ is the number of solutions among the 50 best-known solutions in which the node j is directly followed by the node j' in a route. Let us denote $j-1$ and $j+1$ the predecessor and successor of a node j in the current solution. For each request i of the problem we define a score ϕ_i as follows:

$$\phi_i = \xi_{p(i)-1,p(i)} + \xi_{p(i),p(i)+1} + \xi_{d(i)-1,d(i)} + \xi_{d(i),d(i)+1}. \quad (1)$$

The m requests with the lowest ϕ_i are removed.

3.3. Repair Neighborhoods

We use two insertion heuristics based on best insertion and regret principles.

1. *Best insertion*. At each iteration, the best insertion cost is computed for each destroyed request and the request with the lowest insertion cost is inserted at its best position. The heuristic stops when all requests are routed or none can be inserted (Ropke and Pisinger 2006).

2. *Regret heuristics*. This heuristic is based on the notion of regret used, for example, by Potvin and Rousseau (1993) for the vehicle routing problem with time windows (VRPTW), and proposed in a parametrized version by Ropke and Pisinger (2006). For each request i in the set U of unplanned requests, Δf_i^j designates the insertion cost of the request i in the jth best route at its best position. At each iteration, the request $i^\star$ selected for insertion at its best position is chosen such that $i^\star = \arg\max_{i\in U}(\sum_{j=2}^{k} \Delta f_i^j - \Delta f_i^1)$. The heuristic stops when no more destroyed requests can be inserted in a route or if all requests are inserted. In this paper, we consider four regret-k heuristics with values of k between two and five.

3.4. Acceptance and Stopping Criteria

The decision to accept a new solution is taken according to a simulated annealing criterion. At each iteration, the temperature used by the simulated annealing is multiplied by u. The parameter u is chosen in $]0, 1[$ such that the temperature at the last iteration is equal to 0.2% of the starting temperature. Like Ropke and Pisinger (2006), we set the starting temperature in such a way that a solution 5% worse than the initial solution has a 50% chance of being accepted. In addition, as proposed by Pisinger and Ropke (2007), noise is applied to the objective function as a diversification operator. The stopping criterion for the whole process is the maximum number of iterations.

3.5. Adaptive Aspects

The solution is modified by a destruction and a repair neighborhood. With each neighborhood i is associated a weight w_i and a score π_i. The neighborhoods are selected using a roulette-wheel selection procedure. On the first iteration, all neighborhoods have the same weight. Each time a neighborhood is called, its score is updated depending on its performance. The interested reader can refer to Ropke and Pisinger (2006) for more detail. If the use of a neighborhood leads to a new best solution, its score is incremented by some value σ_1. If the solution generated is better than the current solution, the score is augmented by σ_2. Finally, if the new solution is accepted as a new current solution and has never been encountered before, the score of the neighborhood is increased by σ_3. The entire search is divided into time segments. Here we consider time segments of 100 iterations. At the end of each time segment, the weights of the neighborhoods are updated using the expression $w_i \leftarrow (1-\alpha)w_i + \alpha\pi_i/\theta_i$, where θ_i is the number of times the neighborhood has been called during the last time segment and $\alpha \in [0, 1]$ is a reaction factor that controls the inertia of the weight adjustment. In our implementation, we use the same values as Ropke and Pisinger (2006) for σ_1, σ_2, σ_3. We set $\alpha = 0.5$ instead of $\alpha = 0.1$ to ensure a greater reactivity when a small number of iterations is performed.

3.6. Efficiency of the Algorithm

This algorithm has been implemented and evaluated on the benchmark instances proposed by Li and Lim (2002) for the PDPTW. In these instances, the objective is first to minimize the number of vehicles used in the solution and then the total distance driven. Among the 354 instances, the number of vehicles has been improved in 18 cases and the distance traveled in 55 cases. For 36% of the instances, the implemented method returns the best-known solution. When we obtain the minimum number of vehicles but a suboptimal distance traveled, this distance is within 2% of the best-known result in 73% of the cases. In only 8% of the instances, we do not succeed in finding a solution with the minimum number of vehicles. According to these results we consider that the implemented ALNS gives satisfactory reference results for the PDP without transfer.

4. ALNS for the PDP with Transfers

In this section, we describe the new developments introduced to consider the specificities of the PDPT. This includes not only destruction and repair neighborhoods but also heuristics for selecting appropriate transfer points in neighborhoods (§4.3). Section 4.4 presents the feasibility test used in insertion procedures.

4.1. Destruction Neighborhoods

4.1.1. Transfer Point Removal Heuristic. This heuristic is inspired from a neighborhood for a hub routing problem (Lapierre, Ruiz, and Soriano 2004). The original idea consists of rerouting all requests from one hub to another hub. Requests that use a given transfer point are removed simultaneously to give them a chance to be rerouted through another transfer point.

We first select one active transfer point randomly. If the number of requests going through this transfer point is less than or equal to the number of requests to be destroyed, all these requests are removed. Otherwise, we randomly select a subset of requests to remove. If the number of removed requests is less than r, we remove requests that minimize the distance of their pickup and the closest pickup of an unplanned request plus the distance of their delivery and the closest delivery of an unplanned request. This can be seen as an extension of a relatedness measure based only on the distance between a request and the set of unplanned requests.

Figure 1 illustrates a case where this neighborhood leads to a more profitable solution. If all requests using t_2 are reinserted with a transfer first heuristic (see 4.2.2) on a subset of transfers limited to $\{t_1\}$ (see 4.3), a better solution is obtained.

4.1.2. Pickup/Delivery Cluster Removal Heuristic. This heuristic aims to remove simultaneously a given number of requests that can be efficiently routed through a common transfer point. Two requests, or more, may benefit from using one transfer point if their pickup or their delivery locations form a cluster. If the pickups of these requests are close, they can be serviced together by a first vehicle and then carried to a transfer point in order to be assigned to distinct vehicles. Symmetrically, if their deliveries are located in the same area, they can be serviced by various vehicles for the first part of their journey and then gathered into a single vehicle at a transfer point. The root node of the cluster is selected randomly. All the pickups (or deliveries) are listed from the closest to the farthest to the root node. We iteratively remove the request in position $\lfloor y^p \times (|P| - |U|) \rfloor$ in this list. In this expression, U is the set of unplanned requests, p is a deterministic parameter equal to nine in our implementation, and y is randomly chosen in $[0, 1[$.

4.1.3. History Removal. This heuristic is an extension of the PDPTW history removal presented in §3.2. Let us consider that a transfer point t is decomposed into two nodes for each transferred request j at this node: t_j^- corresponds to the delivery of j at t and t_j^+ is the pickup node of j at t. The score ϕ_i of a transferred request i is computed as follows:

$$\phi_i = \tfrac{1}{2}\big(\xi_{t_i^- - 1,\, t_i^-} + \xi_{t_i^-,\, t_i^- + 1} + \xi_{t_i^+ - 1,\, t_i^+} + \xi_{t_i^+,\, t_i^+ + 1} + \xi_{p(i)-1,\, p(i)} + \xi_{p(i),\, p(i)+1} + \xi_{d(i)-1,\, d(i)} + \xi_{d(i),\, d(i)+1}\big). \quad (2)$$

The 1/2 coefficient makes this score comparable with the score defined by Equation (1) in the case of nontransferred requests.

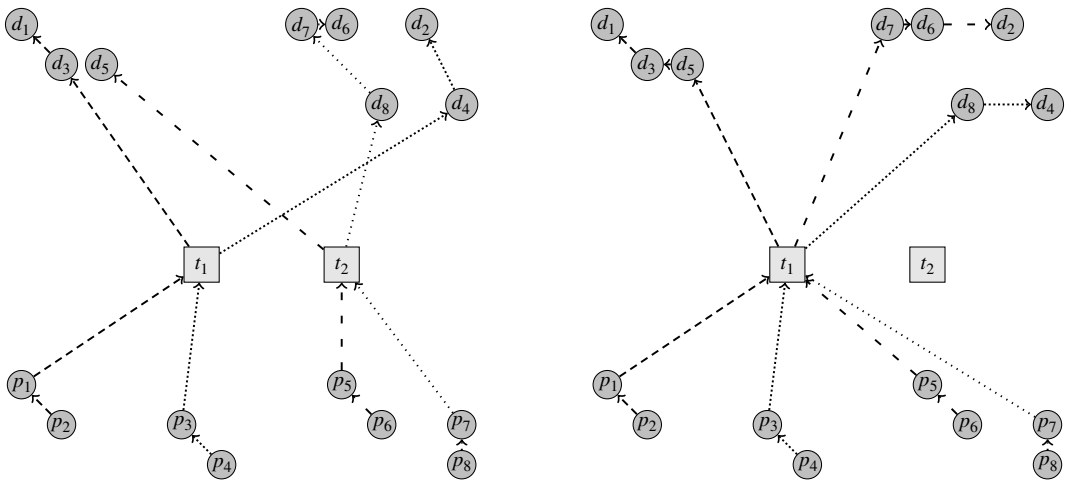


Figure 1 Example of Transfer Point Removal

4.2. Repair Neighborhoods

The heuristics presented in the next three sections are insertion heuristics specifically designed for the PDPT.

4.2.1. Best Insertion with Transfer. This neighborhood is an adaptation of the neighborhood of Mitrović-Minić and Laporte (2006) for the PDPT. For each unplanned request the best insertion without considering any transfer opportunity is first evaluated. Then for each transfer point t and each unplanned request $(p(i), d(i))$, we evaluate the insertion cost as follows:

(i) The insertion cost of the pair $(p(i), t)$ is evaluated and considered as inserted in its best position. Then, an evaluation of the insertion cost of the pair $(t, d(i))$ is done.

(ii) The insertion cost of the pair $(t, d(i))$ is evaluated and considered as inserted in its best position. Then, an evaluation of the insertion cost of the pair $(p(i), t)$ is done for every route.

Finally, the best insertion among all evaluated insertions is performed.

4.2.2. Transfer First. The best insertion with transfer evaluates the insertion of each request sequentially and may miss good opportunities to use transfer points. Figure 2 presents such a case.

It seems locally better to route requests $(p1, d1)$ or $(p2, d2)$ directly (cost = 10) rather than through the transfer point t (cost = 12). However regrouping the requests at transfer point t (using arcs $p1 \to t$, $p2 \to t$ and $t \to d1$) has a cost of 18 whereas the solution without transfers ($p1 \to d1$ and $p2 \to d2$) has a cost of 20.

The transfer first neighborhood gives priority to the use of transfer points. While unplanned requests remain, best insertions with transfers. If no feasible insertion with transfer can be found, the best insertion without transfer is considered. When all requests have been inserted, a post-processing step consists in iteratively removing each transferred request and trying to reinsert it without transfer. This step aims at detecting forced transfers that reduce the quality of the current solution.

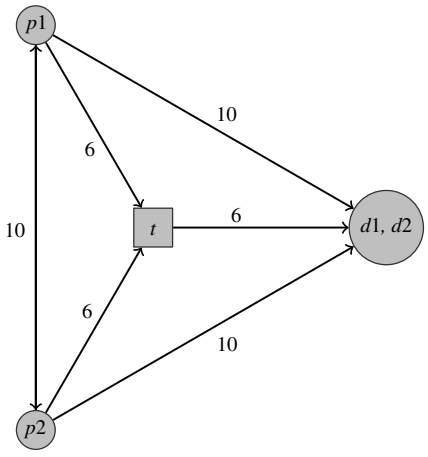


Figure 2 **Example Where Forcing the Use of Transfer Points Improves the Quality of the Solution**

4.2.3. Regret Insertion with Transfer. This heuristic facilitates the insertion of requests for which using a transfer point is cheaper than the insertion without transfer. For each destroyed request we compute the difference between the insertion cost of this request using a transfer point and the best insertion cost without transferring the request. The request that has the largest difference is inserted first in its best position.

4.3. Selecting a Subset of Transfer Points

When a problem has a large number of transfer points, considering each of these locations in insertion heuristics is time consuming. Moreover, in some cases, the use of a transfer point is only interesting if a large enough number of requests use it. Considering a small number of transfer points can help to find good solutions. Therefore, we propose to consider only a subset of transfer points in our heuristics.

We propose five methods to determine subsets of transfer points:

1. The first method consists in randomly selecting a subset of m potential transfer points that will be considered for each request.

2. We define the distance between a request and a transfer point as the length of the following trip: pickup point → transfer point → delivery point. For all requests, the second method consists in selecting only the m closest transfer points.

3. The third method exploits the idea that transfer points are advantageous only when several transshipments are performed at the same point. It considers the m transfer points in which the greatest number of requests is transferred after the destruction operation.

4. The fourth method uses a simplification of the PDPT, known as the k-star hub problem Blasum et al. (2007), which is solved using a commercial solver. It is defined as follows: we consider a set of requests and k transfer points. Each request can be delivered directly, with a given cost (equal to the distance between the pickup and the delivery), or it can be routed through a transfer point. In the latter case, a cost is associated with an arc between the pickup point and the transfer point and with the arc between the transfer point and the delivery point. If an arc between two nodes is already used, all requests that have these nodes in common can use it at cost zero. This problem is solved to determine which transfer points the unplanned requests should use. In our

problem, every pickup and delivery node are distinct. Therefore, we cannot apply this approach. In the first step, clusters of pickups and clusters of deliveries are built and considered as single nodes. In the existing routes, we consider that pickups and deliveries that are serviced in a row form a cluster. Once every node of the current solution is assigned to a cluster, we add each node of unplanned requests to their closest cluster. We solve the k-star hub problem where some arcs are already used (according to the destroyed solution). The cost of an arc between two clusters is set to the smallest distance between two nodes of these clusters. For each unplanned request, the transfer point to consider during the reconstruction is the one, if any, used for this request in the optimal solution of the k-star hub problem.

5. The last method is based on the historical performance of the transfer points. For each request i and each transfer point t, we define $h_{i,t}$ as the cost of the best-known solution where i is transferred through t. If the transfer point t has never been used by request i, then we set $h_{i,t} = \infty$. For each request, a roulette wheel is used to select the transfer point to be considered. The weight associated with a transfer point t for a request i is $\min_{t' \in T} h_{i,t'}/h_{i,t}$.

In practice, we considered transfer selection in inserting heuristics with $m = 1$. For each inserting heuristic with transfer, one neighborhood is created for each of the five transfer selection methods, plus one neighborhood that considers all transfer points. The adaptive aspect of the ALNS gives priority to the most efficient neighborhoods.

4.4. Neighborhood Evaluation

The consideration of transfer points raises new difficulties, for example, checking if a modification of the solution (e.g., inserting or removing a request from a route) is feasible or not. Inserting a request may impact on the feasibility of the time window constraints on more than one route. This is highlighted in Figure 3.

The insertion of p_1 in the dashed route can cause a delay on the dotted route, propagated up to t_2, and make the insertion of d_1 unfeasible. The feasibility of an insertion can be checked by computing the new service times at nodes that follow the insertion position. However, it is computationally expensive. We consider an extension of forward time slacks (Savelsbergh 1985) based on a square matrix Ω containing the sum of waiting times between each connected point of the problem. Each time a request is removed or inserted, this matrix has to be updated. This update has a worst-case complexity of $O(|V|^2)$. Each time a request is inserted or removed, the latest possible service time at each inserted node has to be updated. However, using Ω, an insertion can be checked in constant time. Let us consider the insertion of request i. Let us denote $p(i)+1$ the node following the insertion position of $p(i)$, $d(i)-1$ the node preceding the insertion position of $d(i)$, and $d(i)+1$ the node following the insertion position of $d(i)$. The delay at a node j_2 generated by the insertion of a node j_1 is denoted δ_{j_1,j_2}. First, we check that inserting $p(i)$ does not violate the latest possible service time at $p(i)+1$. Secondly, we compute the new service time at $d(i)-1$, which is equal to its current service time plus $\max(0, \delta_{p(i),p(i)+1} - \Omega_{p(i)+1,d(i)-1})$. Hence, considering the delay at $d(i)-1$ when inserting $p(i)$, the service time at $d(i)$ and $d(i)+1$ can be computed in constant time.

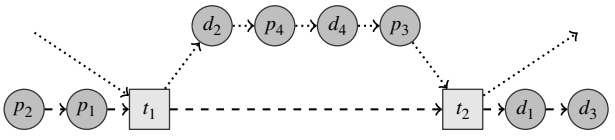


Figure 3 Difficulties in Evaluating the Feasibility of an Insertion

5. Computational Experiments

The method was coded in C++ and experiments were run on an i3-540 computer operated by Windows. We first discuss ALNS settings and use real-life data to evaluate the benefits of routing with transfer points. Considering the large run time obtained for the ALNS, the impact of reducing the number of iterations is then studied. Finally, the proposed ALNS is evaluated on related benchmark instances from the literature. For some of these generated instances, a lower bound on the savings achieved thanks to the use of transfer is computed.

5.1. Instances

We evaluate the algorithm on a set of 10 real-life instances arising from distinct sources: specialized centers for disabled children, vocational rehabilitation centers for adults, and schools receiving both able-bodied and disabled children. In the first two sources, each delivery location is a common destination for dozens of requests. The instances considered correspond to a few centers that wish to pool the organization of their transport. In the following, the names of these instances start with "centers." In the case of schools for disabled children, each school is the destination for a small number of people. Thus, the instances generally include all the schools of a given geographical area, possibly several dozen. The names of these instances start with "schools."

The considered instances include between 55 and 193 requests. They only consider outbound trips from home to the centers. Thus, the pickup points are the personal addresses of the transported people and the delivery and transfer points are the centers or schools.

All requests have distinct pickup points and correspond to individual demands ($q_i = 1$). The number of delivery points is comprised between two and five for specialized schools or vocational rehabilitation centers and between 13 and 33 for all other schools. All delivery points can be used as transfer points.

Instances are named "category-$|P|$-$|D|$" such that the number of requests and delivery locations is reflected in their name. For example, instance centers-193-5 is of the type "centers" and consists of 193 requests and 5 delivery locations.

The delivery points may have different opening hours. The time window $[e_i, l_i]$ for vehicle arrival is defined such that l_i corresponds to the beginning of the daily activity. The time window opening e_i is set at 15 to 30 minutes prior to l_i, depending on the center. We assume that the vehicles start from a dummy location located at distance zero from every pickup location. The traveling time matrices between locations are asymmetric and satisfy the triangular inequality.

We also use the instance set proposed by Mitrović-Minić and Laporte (2006). The 4,320 instances contain either 50 or 100 requests and are divided into 144 classes. The average results are given by Mitrović-Minić and Laporte (2006) for 64 classes of instances. The nodes of the problems are located in a 60 km × 60 km square. A constant speed of 60 km/h is considered for the vehicles. The Manhattan distance is used. A 10-hour planning horizon is considered and the capacity of the vehicles is not binding. Several types of node distribution are considered. "*Uniform*" represents instances where the nodes are positioned randomly in the square and "*Border* $(a \times a)$" represents instances where the nodes are located in four clusters of side length a located near the middle of each side of the 60 km × 60 km square. Finally, "*Random* $(a \times a)$" represents instances where the nodes are located in four clusters of side length a located randomly in the square. Various distributions of the time windows are considered; "10" represents the case where the requests have no time windows, "2–4–8" represents the case where 30% of the requests are two-hour requests, 50 % of the requests are four-hour requests and 20% of the requests are eight-hour requests.

Following Mitrović-Minić and Laporte (2006), we tested several configurations of transfer points layout. Type T0 means that no transfer point is considered. In *Uniform* instances, T4 means that four transfer points are located at the positions (20, 20), (40, 20), (20, 40), (40, 40). In *Border* instances, the scheme T2 designates the case with a single transfer point located in the middle of the area. In *Random* instances, T2 designates the case with four transfer points located at the intersection of parallel horizontal and vertical lines drawn from the centers of the clusters. Finally, for all cluster instances, the scheme T3 considers the transfer points of scheme T2 plus four transfer points located at the center of the clusters. The service time is considered to be five minutes at each pickup and delivery node and zero minutes at the transfer points.

5.2. ALNS Parameter Settings

For most of the parameters of the ALNS algorithm, we validate the values proposed by Ropke and Pisinger (2006). It turns out that for the PDPT, the percentage r of requests removed in each iteration can be reduced to improve the method. We investigated four settings where r is chosen in the intervals [10, 50], [10, 35], [10, 20], and [5, 10], respectively, and run the ALNS for 25,000 iterations. The results obtained for five representative instances are summarized in Table 1.

Unsurprisingly, when the number of requests removed decreases, the run time is reduced. We also observe that better or equivalent costs are obtained when the percentage r is taken in the interval [10, 20]. The counterpart is a slight increase in the standard deviation. Thus, in the following tests, the percentage r is taken in the interval [10, 20].

5.3. Evaluation of the Benefits of Transfers

Each instance is solved five times during 25,000 iterations. We consider that the number of vehicles is not limited and minimize the total distance traveled by the vehicles. The objective is to compare the cost of

Table 1 Impact of the Percentage of Requests Removed

Instances	% Req. removed	Avg.	Min.	Std. dev.	Time (sec.)
Centers-84-2	[10, 50]	778.61	773.15	3.16	3,187
	[10, 35]	775.52	767.20	6.64	1,766
	[10, 20]	**769.83**	**762.15**	6.10	1,167
	[5, 10]	777.30	769.38	5.14	755
Centers-109-2	[10, 50]	620.98	617.82	4.68	18,297
	[10, 35]	615.76	609.98	3.68	10,693
	[10, 20]	**610.74**	**606.85**	3.36	5,418
	[5, 10]	620.97	616.73	4.27	2,328
Centers-193-5	[10, 50]	1,159.10	1,151.90	6.85	123,434
	[10, 35]	1,160.20	1,151.55	6.98	59,309
	[10, 20]	**1,152.44**	**1,134.90**	10.40	37,617
	[5, 10]	1,153.69	1,142.27	7.20	16,085
Schools-66-13	[10, 50]	800.51	800.27	0.33	10,379
	[10, 35]	800.26	799.05	0.74	5,504
	[10, 20]	800.51	800.27	0.33	2,538
	[5, 10]	**798.60**	**792.82**	4.55	1,060
Schools-84-21	[10, 50]	**1,053.64**	1,051.32	2.35	24,690
	[10, 35]	1,054.15	**1,050.42**	2.57	10,839
	[10, 20]	1,055.67	1,052.17	3.43	4,818
	[5, 10]	1,057.41	1,056.03	1.47	2,513

Notes. The first column designates the instance. The second column indicates the interval for r. Columns three to five report the average and minimum values of the objective function and the standard deviation found over five runs, respectively. The last column reports the average run time in seconds.

Table 2 Real-Life Instances: Minimization of the Distance Traveled—ALNS 25,000 Iterations

Instances	Transfer points	Avg.	Min.	Std. dev.	Time (sec.)	Gap (%)
Centers-84-2	0	782.04	782.03	0.02	157	2.54
	2	769.83	762.15	6.10	1,167	
Centers-81-2	0	453.09	452.72	0.56	205	3.23
	2	438.64	438.08	0.58	3,599	
Centers-87-2	0	478.06	477.25	0.77	206	2.96
	2	463.92	463.12	0.48	4,215	
Centers-109-2	0	650.20	647.88	2.22	280	6.33
	2	610.74	606.85	3.36	5,418	
Centers-193-5	0	1,253.63	1,250.30	3.48	750	9.23
	5	1,152.44	1,134.90	10.40	37,617	
Schools-55-16	0	654.72	654.72	0.00	135	0.75
	16	649.78	649.78	0.00	413	
Schools-66-13	0	824.82	824.82	0.00	159	2.98
	13	800.51	800.27	0.33	2,538	
Schools-84-21	0	1,121.37	1,121.37	0.00	215	6.17
	21	1,055.67	1,052.17	3.43	4,818	
Schools-84-33	0	1,239.65	1,239.32	0.30	209	6.23
	33	1,164.48	1,162.08	2.22	6,239	
Schools-106-24	0	978.64	978.15	0.73	285	4.12
	24	940.02	937.85	2.95	9,867	

Notes. The first column contains the instance names. The second column indicates the number of transfer points considered. Columns three to five report the average and best results and the standard deviation over five runs, respectively. The sixth column reports the average run time and the last column shows the gap between the best-known solutions with and without transfer points.

optimized solutions for the PDPTW (using the ALNS algorithm described in §3) with the best costs found for the PDPT. Table 2 summarizes the results on real-life instances.

The table shows that the savings because of transfers can vary a lot from one instance to another. In practice, it is observed that transfers are useful when several people from the same geographical area make transportation requests to distinct destinations.

Table 3 presents the savings achieved on Mitrović-Minić and Laporte (2006) instances. We compare solutions obtained using the ALNS heuristics on instances with various patterns of transfer points.

The gap found by the ALNS between solutions using or not using transfer points is similar to the gap reported by Mitrović-Minić and Laporte (2006) for each class of instances.

To evaluate the minimal savings achieved by transfers, we wish to compare upper bounds of the PDPT found by the ALNS algorithm with optimal solutions or lower bounds for the PDP (without transfer). Given one instance, let us denote UB_T the best upper bound for the PDPT and LB a lower bound for the PDP. The savings because of the use of transfer points are greater than or equal to $gap_T = (LB - UB_T)/UB_T$. We were able to compute LB with a column generation approach on instances with a uniform distribution of the nodes and time windows with the pattern "2–4–8" (see Table 3). For this class of instances, the average value of δ_T is 5.08%. In 60% of the cases, gap_T is greater than 5%. Let us also denote UB an upper bound found for the PDP and $gap^* = (UB - LB)/LB$ the optimality gap. The value of gap^* is 1.1 on average and is below 2% in 90% of the cases. The minimal, average and maximal observed values of these gaps are reported in Table 4.

Note that we also performed some tests on PDPTW instances with known optimal solutions. The instances of Li and Lim (2002) are constructed in such a way that transfers did not produce an interesting

Table 3 Evaluation of Savings Achieved Using Transfers on Mitrović-Minić and Laporte (2006) Instances

Node distribution	TW	Transfer points	Cost	Gap (%)
Uniform	10	T0	800.29	
		T4	738.02	7.78
	2–4–8	T0	1,217.03	
		T4	1,145.47	5.88
Border (6 × 6)	10	T0	439.68	
		T2	366.26	16.70
		T3	364.68	17.06
	2–4–8	T0	956.47	
		T2	696.46	27.18
		T3	686.60	28.22
Border (10 × 10)	10	T0	479.49	
		T2	439.11	8.42
		T3	432.23	9.86
	2–4–8	T0	972.54	
		T2	767.57	21.08
		T3	753.20	22.55
Border (20 × 20)	10	T0	593.28	
		T2	584.61	1.46
		T3	555.75	6.32
	2–4–8	T0	991.22	
		T2	905.53	8.65
		T3	882.51	10.97
Random (6 × 6)	10	T0	323.31	
		T2	297.84	7.88
	2–4–8	T0	665.03	
		T2	568.74	14.48

Notes. The first three columns designate the node distribution, the time windows distribution, and the transfer points configuration, respectively. The fourth column reports the cost found by the ALNS. The fifth column shows the gap with the solution without transfers (T0).

Table 4 Comparison Between Solution With and Without Transfers

	Min (%)	Average (%)	Max (%)
$gap_T = \frac{LB - UB_T}{UB_T}$	2.05	5.08	8.70
$gap^* = \frac{UB - LB}{LB}$	0.00	1.10	2.44

output. The instances of Ropke and Cordeau (2009) have very tight capacity constraints. Therefore only few requests can share vehicles, which does not allow much consolidation of the flows at transfer points. Thus transfers did not yield significant savings on these instances either.

In conclusion, the comparison based on Mitrović-Minić and Laporte (2006) instances shows that introducing transfer points can result in savings of up to 28%. On instances where a lower bound on the cost without transfer has been computed, the average savings achieved thanks to the use of the transfer is larger than 5.08%.

5.4. Run-Time Reduction

The experiments of §5.3 show that the time needed to perform 25,000 iterations is prohibitive for some instances. This is due to the time needed to explore some neighborhoods considering transfer. Thus, we choose to decrease the number of iterations. Four settings with 5,000 and 10,000 iterations and various values for r are compared with the 25,000 iteration runs. Because the cooling schedule of the simulated annealing depends on the number of iterations, the runs with 5,000, 10,000, and 25,000 iterations are performed separately. Table 5 summarizes the results.

When the number of iterations performed is smaller, the results of the algorithm are less stable and it has more difficulty in finding very good solutions. It can also be noted that, even if the running times are equivalent, it is more useful to perform 5,000 iterations with $r \in [10, 20]$ than 10,000 with $r \in [5, 10]$. However, the difference between the 5,000 iteration runs and the 25,000 iteration runs is not dramatic, on average 0.68%. From this perspective, performing 5,000 iterations may provide a good trade-off between the quality of the solution and the running time. Preliminary tests with a faster version of the ALNS running for 30 seconds provide results that compete with those of Mitrović-Minić and Laporte (2006).

5.5. Evaluation on Related Instances

We evaluate the ALNS on 26 representative classes grouping 780 instances from Mitrović-Minić and Laporte (2006). For each class of instances, the results presented in Table 6 are average values over the 30 instances of the class. Concerning the ALNS, each instance is executed five times and the cost of the best solution is reported. The ALNS is run over 25,000 iterations with $r \in [10, 20]$.

These tests show that the ALNS method presented in this article is better overall than the local search on the instances with loose time windows (the TW 10 instances). However, unstable results are observed. We assume that the use of the Manhattan distance makes these instances difficult. In the presence of harder time windows (TW 2–4–8 instances), the ALNS clearly outperforms the local search. As the ALNS is designed to solve problems with quite hard time windows, this result is not too surprising.

Table 5 Impact of the Number of Iterations

Instances	No. of iter.	% Req.	Avg.	Min.	Time (sec.)
Centers-84-2	5,000	[10, 20]	780.63	767.27	205
	10,000	[10, 20]	776.44	765.60	482
	10,000	[5, 10]	784.73	780.83	244
	25,000	[10, 20]	769.83	762.15	1,167
Centers-81-2	5,000	[10, 20]	442.64	437.13	599
	10,000	[10, 20]	441.71	440.78	1,333
	10,000	[5, 10]	442.03	440.12	872
	25,000	[10, 20]	438.64	438.08	3,599
Centers-87-2	5,000	[10, 20]	466.68	463.08	660
	10,000	[10, 20]	467.11	463.77	1,651
	10,000	[5, 10]	466.85	464.33	684
	25,000	[10, 20]	463.92	463.12	4,215
Centers-109-2	5,000	[10, 20]	620.05	614.52	1,055
	10,000	[10, 20]	618.51	615.27	2,047
	10,000	[5, 10]	626.98	617.82	833
	25,000	[10, 20]	615.96	609.27	5,804
Centers-193-5	5,000	[10, 20]	1,163.06	1,145.68	7,447
	10,000	[10, 20]	1,166.82	1,165.53	12,635
	10,000	[5, 10]	1,157.47	1,141.80	6,763
	25,000	[10, 20]	1,152.44	1,134.90	37,617
Schools-55-16	5,000	[10, 20]	649.78	649.78	413
	10,000	[10, 20]	649.78	649.78	864
	10,000	[5, 10]	649.87	649.78	931
	25,000	[10, 20]	649.78	649.78	2,190
Schools-66-13	5,000	[10, 20]	800.39	800.27	472
	10,000	[10, 20]	799.66	798.45	932
	10,000	[5, 10]	801.65	800.87	414
	25,000	[10, 20]	800.51	800.27	2,538
Schools-84-21	5,000	[10, 20]	1,060.36	1,057.78	792
	10,000	[10, 20]	1,055.60	1,052.32	1,754
	10,000	[5, 10]	1,059.10	1,055.27	1,160
	25,000	[10, 20]	1,055.67	1,052.17	4,818
Schools-84-33	5,000	[10, 20]	1,168.61	1,163.35	1,846
	10,000	[10, 20]	1,166.75	1,165.32	3,492
	10,000	[5, 10]	1,168.37	1,164.18	1,291
	25,000	[10, 20]	1,164.48	1,162.08	6,239
Schools-106-24	5,000	[10, 20]	945.41	940.82	2,001
	10,000	[10, 20]	943.01	939.02	4,156
	10,000	[5, 10]	944.21	941.1	1,836
	25,000	[10, 20]	940.02	937.85	9867
Average	5,000	[10, 20]	809.76	803.97	1,549
	10,000	[10, 20]	808.54	805.58	2,935
	10,000	[5, 10]	810.13	805.61	1,503
	25,000	[10, 20]	805.13	800.97	7,805

Notes. The second column indicates the number of iterations performed. The third column is the interval in which r is chosen. The fourth and fifth columns are the average and minimum values found over five runs, respectively. The last column reports the average run time in seconds.

6. Summary and Further Research

In this article, we proposed an adaptive large neighborhood search for solving the pickup and delivery

Table 6 Comparison with the Local Search of Mitrović-Minić and Laporte (2006)

Node distribution	TW	Transfer points	M-M & L	ALNS obj.	ALNS time (sec.)
Uniform	10	T0	845.86	**800.29** (5.39%)	122
		T4	791.53	**738.02** (6.76%)	1,423
	2–4–8	T0	1,324.63	**1,217.03** (8.12%)	34
		T4	1,297.84	**1,145.47** (11.74%)	328
Border (6 × 6)	10	T0	473.46	**439.68** (7.13%)	592
		T2	**360.47**	366.26 (−1.61%)	1,490
		T3	409.97	**364.68** (13.41%)	2,566
	2–4–8	T0	1,093.38	**956.47** (12.52%)	39
		T2	809.61	**696.46** (13.98%)	869
		T3	811.35	**686.60** (15.38%)	1,098
Border (10 × 10)	10	T0	526.66	**479.49** (8.96%)	560
		T2	440.86	**439.11** (0.40%)	1,350
		T3	526.18	**432.23** (17.86%)	2,714
	2–4–8	T0	1,120.78	**972.54** (13.23%)	37
		T2	892.38	**767.57** (13.99%)	846
		T3	912.47	**753.20** (17.46%)	1,070
Border (20 × 20)	10	T0	657.67	**593.28** (9.79%)	389
		T2	633.91	**584.61** (7.78%)	668
		T3	647.07	**555.75** (14.11%)	2,340
	2–4–8	T0	1,137.78	**991.22** (12.88%)	38
		T2	1,053.25	**905.53** (14.03%)	591
		T3	1,073.21	**882.51** (17.77%)	787
Random (6 × 6)	10	T0	**321.84**	323.31 (−0.46%)	927
		T2	311.39	**297.84** (4.35%)	2,596
	2–4–8	T0	777.98	**665.03** (14.52%)	44
		T2	726.34	**568.74** (21.70%)	789
Average	10	All	534.37	**493.43** (7.04%)	1,364
	2–4–8	All	1,002.38	**862.18** (14.41%)	505

Notes. The first three columns designate the node distribution, the time windows distribution, and the transfer points configuration, respectively. The fourth column indicates the cost found by Mitrović-Minić and Laporte (2006), denoted M-M & L. The fifth column reports the cost found by the ALNS, as well as the gap with column M-M & L. The last column is the average run time of the ALNS in seconds.

problem with transfers. Our approach relies on the ALNS algorithm described by Pisinger and Ropke (2007), which implementation is evaluated on the instances proposed by Li and Lim (2002).

The algorithm is then extended to handle transfer points. This requires introducing new destruction and repairing heuristics dedicated to the use of transfer points. The algorithm improves the results of Mitrović-Minić and Laporte (2006) for the PDPT. We also apply the method to real-life instances concerning the transportation of mentally or physically disabled people. In these cases, the introduction of transfer points generally brings significant improvements (up to 9%), which differ greatly from one geographical area to another.

This work suggests several improvements and further research perspectives. Solving the PDPT clearly shows the potential benefits of using transfer points but requires much more computation time than solving the PDPTW. Most of the time is spent on the calculation of insertion positions. Indeed, every insertion of a request in a route with a transfer point can impact the passengers not only along this route but also those along adjacent routes (routes that include this transfer point). The feasibility check is then more demanding and leads to a long computation time. From a strategic point of view, the decision makers are likely to accept long running times provided the heuristic proposes very good solutions. On the other hand, in an operational context, the users would like to test various scenarios (fleet configurations, special days with various demand profile, etc.). This requires faster computation. As the ALNS relies on an adaptive mechanism, it needs to run during a large enough number of iterations. Providing a method capable of quickly giving good results for the PDPT is then an important practical issue.

An important criterion for measuring the quality of service to passengers is the ride time, i.e., the time spent by each passenger in the vehicles. Associating each request with a maximum ride time transforms the problem into a dial-a-ride-problem (DARP). Applying the ALNS to the DARP with transfers leads to the additional difficulty of checking the ride-time constraints when repairing a destroyed solution. Another challenge is to compute tight lower bounds for the PDPT and to develop exact methods (branch and price) for solving instances of the PDPT of realistic size.

References

Berbeglia G, Cordeau J-F, Laporte G (2010) Dynamic pickup and delivery problems. *Eur. J. Oper. Res.* 202(1):8–15.

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

Blasum U, Hochstattler W, Oertel P, Woeginger G (2007) Steiner diagrams and *k*-star hubs. *J. Discrete Algorithms* 5(3):622–634.

Cortés CE, Jayakrishnan R (2002) Design and operational concepts of high-coverage point-to-point transit system. *Transportation Res. Record* 1783:178–187.

Cortés CE, Matamala M, Contardo C (2010) The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *Eur. J. Oper. Res.* 200(3):711–724.

Fugenschuh A (2009) Solving a school bus scheduling problem with integer programming. *Eur. J. Oper. Res.* 193(3):867–884.

Gørtz IL, Nagarajan V, Ravi R (2009) Minimum makespan multi-vehicle dial-a-ride. Fiat A, Sanders P, eds. *Algorithms—ESA 2009: 17th Annual European Symposium*, Lecture Notes in Computer Science, Vol. 5757 (Springer-Verlag, Berlin, Heidelberg), 540–552.

Kerivin HLM, Lacroix M, Mahjoub AR, Quilliot A (2008) The splittable pickup and delivery problem with reloads. *Eur. J. Indust. Engrg.* 2(2):112–133.

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220(4598):671–680.

Lapierre SD, Ruiz AB, Soriano P (2004) Designing distribution networks: Formulations and solution heuristic. *Transportation Sci.* 38(2):174–187.

Laporte G, Musmanno R, Vocaturo F (2010) An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Sci.* 44(1):125–135.

Li H, Lim A (2002) A metaheuristic for the pickup and delivery problem with time windows. Bilof R, Palagi L, eds. *Proc. 13th Internat. Conf. Tools with Artificial Intelligence, ICTAI-2001* (IEEE, Dallas), 160–167.

Lin CKY (2008) A cooperative strategy for a vehicle routing problem with pickup and delivery time windows. *Comput. Indust. Engrg.* 55(4):766–782.

Masson R, Lehuédé F, Péton O (2011) A tabu search algorithm for the dial-a-ride problem with transfers. Benyoucef L, Trentesaux D, Artiba A, Rezg N, eds. *Proc. Internat. Conf. Indust. Engrg.* (Systems Management, Metz, France), 1224–1232.

Mitrović-Minić S, Laporte G (2006) The pickup and delivery problem with time windows and transshipment. *INFOR* 44(3):217–228.

Mues C, Pickl S (2005) Transshipment and time windows in vehicle routing. Bein W, Chin FYL, Hsu DF, Palis ML, eds. *ISPAN'05: Proc. 8th Internat. Sympos. Parallel Architectures, Algorithms and Networks* (IEEE Computer Society, Washington, DC), 113–119.

Nakao Y, Nagamochi H (2010) Worst case analysis for pickup and delivery problems with transfer. *IEICE Trans. Fundamentals of Electronics, Comm. Comput. Sci.* E91-A(9):2328–2334.

Oertel P (2000) Routing with reloads. Ph.D. thesis, University of Cologne, Germany.

Parragh SN, Doerner KF, Hartl RF (2008) A survey on pickup and delivery problems, part 2: Transportation between pickup and delivery locations. *J. für Betriebswirtschaft* 58(2):81–117.

Petersen H, Ropke S (2011) The pickup and delivery problem with cross-docking opportunity. Böse J, Hu H, Jahn C, Shi X, Stahlbock R, Voß S, eds. *Computational Logistics*, Lecture Notes in Computer Science, Vol. 6971 (Springer, Berlin/Heidelberg), 101–113.

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. *Comput. Oper. Res.* 34(8):2403–2435.

Potvin J-Y, Rousseau J-M (1993) A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *Eur. J. Oper. Res.* 66(3):331–340.

Ropke S, Cordeau J-F (2009) Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Sci.* 43(3):267–286.

Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Sci.* 40(4):455–472.

Ropke S, Pisinger D (2010) Large neighborhood search. Gendreau M, Potvin J-Y, eds. *Handbook of Metaheuristics*, 2nd ed. (Springer, New York), 399–419.

Savelsbergh MWP (1985) Local search in routing problems with time windows. *Ann. Oper. Res.* 4(1):285–305.

Schrimpf G, Schneider J, Stamm-Wilbrandt H, Dueck G (2000) Record breaking optimization results using the ruin and recreate principle. *J. Comput. Physics* 159:139–171.

Shang JS, Cuff CK (1996) Multicriteria pickup and delivery problem with transfer opportunity. *Comput. Indust. Engrg.* 30(4): 631–645.

Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. Maher M, Puget JF, eds. *Proc. 4th Internat. Conf. Principles and Practice of Constraint Programming*, (Springer, Berlin, Heidelberg), 417–431.

Thangiah SR, Fergany A, Awam S (2007) Real-time split-delivery pickup and delivery time window problems with transfers. *Central Eur. J. Oper. Res.* 15(4):329–349.

Operations Research Letters 41 (2013) 211–215

Contents lists available at SciVerse ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers

Renaud Masson, Fabien Lehuédé, Olivier Péton *

LUNAM Université, Ecole des Mines de Nantes, IRCCyN UMR CNRS 6597 (Institut de Recherche en Communications et Cybernétique de Nantes), Nantes, France

ARTICLE INFO

Article history:
Received 24 September 2012
Received in revised form
15 January 2013
Accepted 16 January 2013
Available online 23 January 2013

Keywords:
Pickup and delivery problem
Transfers
Feasibility check
Forward time slack

ABSTRACT

The Pickup and Delivery Problem with Transfers (PDPT) consists of defining a set of minimum cost routes in order to satisfy a set of transportation requests, allowing them to change vehicles at specific locations. In this problem, routes are strongly interdependent due to request transfers. Then it is critical to efficiently check if inserting a request into a partial solution is feasible or not. In this article, we present a method to perform this check in constant time.

1. Introduction

The Pickup and Delivery Problem with Transfers (PDPT) [3] consists of defining a set of routes that minimize the distance traveled in order to satisfy a set of transportation requests. Each transportation request has a known origin called the *pickup point* and a destination called the *delivery point*. The PDPT allows for transfers (also referred to as preemption in other papers): a request may be picked up by one vehicle, dropped at a transfer point and later be moved to its delivery point by another vehicle. Several requests can share a vehicle as long as its capacity is not exceeded. The service at pickup and delivery points must occur during time windows.

Neighborhood-based heuristics are common methods for solving vehicle routing problems with time-related constraints. It is well known that a critical factor of performance is the ability of the method to quickly check the feasibility of a solution. For example, Hunsaker and Savelsbergh discussed a sufficient condition to assess the feasibility of a route in the dial-a-ride problem under maximum wait time and maximum ride time constraints [6]. They show that this sufficient condition can be checked in linear time. Firat and Woeginger show that the whole feasibility check can be performed in linear time [5]. Feasibility testing raises an additional difficulty in the case of the PDPT where routes are *interdependent*. Transferring a request from one vehicle to another introduces precedence constraints at the transfer point. Since in many heuristic methods, millions of insertions are evaluated during the search, their feasibility must be checked as efficiently as possible.

The previous heuristic algorithms developed for the PDPT did not discuss the complexity of their feasibility tests [7,8].

Contributions of this note. In the PDPT, checking the feasibility of a request insertion contains two parts. The first part checks the temporal constraints of the problem. We propose an incremental method which enables us to check in constant time if a given insertion is consistent with the temporal constraints of the problem. This method is based on the *Forward Time Slack* principle of Savelsbergh [9]. The second part checks the vehicle capacity constraints. Since this test is similar to those performed in the pickup and delivery problem with no transfer, we do not discuss it. The present paper covers only the PDPT, but we believe this work is relevant for a variety of vehicle routing problems with precedence or synchronization constraints (see [4] for a survey).

In many pickup and delivery applications, the time of departure or arrival of vehicles may be imposed. Without loss of generality, we assume that all vehicles leave the depot at time 0 and all requests are serviced as early as possible. Our approach can be easily adapted to the symmetric case, with imposed arrival date at the depot and requests serviced as late as possible.

The remainder of the paper is organized as follows: The PDPT is formally stated in Section 2. In Section 3 we recall the Forward Time Slack principle. Section 4 defines the feasibility problem. Section 5 extends the Forward Time Slack principle to the PDPT. Section 6 presents the feasibility test for an insertion without transfer, while Section 7 concerns the insertion of a

* Correspondence to: Ecole des Mines de Nantes, 4 rue Alfred Kastler, F-44307 Nantes Cedex 3, France. Tel.: +33 251 858 313; fax: +33 251 838 349.
E-mail addresses: renaud.masson@mines-nantes.fr (R. Masson), fabien.lehuede@mines-nantes.fr (F. Lehuédé), olivier.peton@mines-nantes.fr (O. Péton).

0167-6377/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
doi:10.1016/j.orl.2013.01.007

request through a transfer point. The feasibility check described in this article requires some preprocessing which is described in Section 8: each time a request is removed from or inserted into a current solution, the service times of the requests and a matrix of waiting times must be updated. This update has a quadratic complexity but it is much less frequent than the constant time feasibility check described here.

2. The pickup and delivery problem with transfers

The PDPT can be formally described as follows. Let R be the set of transportation requests, T the set of transfer points and K the set of homogeneous vehicles of capacity Q. A request $r \in R$ corresponds to the transportation of q_r load units from a pickup point p_r to a delivery point d_r. The sets of all pickup and delivery points are denoted by P and D respectively. Each vehicle performs a single route. The starting and ending depots of vehicles are designated by o and o' respectively. The PDPT is defined on a complete directed graph $G = (V, U)$ where $V = P \cup D \cup T \cup \{o\} \cup \{o'\}$ and $U = \{(u, v) | u, v \in V, u \neq v\}$. With each arc $(u, v) \in U$ is associated a non-negative travel time $\theta_{u,v}$. We assume that all travel times respect the triangle inequality. A time window $[e_v, l_v]$ is associated with each vertex $v \in V$, where e_v and l_v represent the earliest and the latest time at which the service can begin at vertex v. The service at each vertex $v \in P \cup D \cup T$ has a known duration s_v, modeling the time needed to get requests in or off the vehicle. The number of load units carried simultaneously by a vehicle cannot exceed its capacity Q.

A vehicle can wait at a vertex only if it arrives there before the opening of the corresponding time window. If two requests have a common pickup or delivery point, the corresponding vertex is duplicated. According to this modeling, a vertex cannot be associated with more than one request.

A solution of the PDPT is a set of $|K|$ routes serving all requests such that each vehicle starts at o and ends at o'. For every request $r \in R$, vertices p_r and d_r can be served by the same route, provided that p_r is served before d_r. Vertices p_r and d_r can also be served by distinct routes $k \in K$ and $k' \in K$. In this case k and k' have to visit a common transfer point $\tau \in T$, such that τ is visited after p_r in route k and before d_r in route k'. Moreover, r must get off vehicle k at τ before being reloaded into k'. The objective of the PDPT is to serve all requests at minimal cost.

3. Forward time slack

We recall the principle of Forward Time Slack proposed by Savelsbergh [9] for the Traveling Salesman Problem with Time Windows (TSPTW) and the Vehicle Routing Problem with Time Windows (VRPTW).

Let us first introduce some notation that is used all along the paper: let u and v be two vertices serviced by a route k and $\psi_{u,v}$ the ordered set of vertices on the path $(u, \ldots, v)$. The waiting time at a vertex $i \in \psi_{u,v}$ is denoted by w_i. The direct predecessor and successor of vertex i in route k are denoted by $\rho(i)$ and $\sigma(i)$ respectively. The set of all successors of i is denoted by $\Gamma(i)$.

We assume that the service times are scheduled as early as possible. Thus the service time h_v at v is equal to the service time h_u, plus the travel and service durations along the path, plus the waiting times at the vertices of the path. If we denote $TWT_{\psi_{u,v}} = \sum_{i \in \psi_{\sigma(u),v}} w_i$, the total waiting time on path $(u, \ldots, v)$, then

$$h_v = h_u + \sum_{i \in \psi_{u,\rho(v)}} (s_i + \theta_{i,\sigma(i)}) + TWT_{\psi_{u,v}}. \quad (1)$$

For each vertex u on a route k, Savelsbergh introduces the notion of *forward time slack* F_u to represent the maximal amount of time the service at vertex u can be postponed without violating any time window at its successors:

$$F_u = \min_{i \in \{u\} \cup \Gamma(u)} \left\{ l_i - h_u - \sum_{j \in \psi_{u,\rho(i)}} (s_j + \theta_{j,\sigma(j)}) \right\}.$$

Using forward time slacks, the feasibility of a client insertion in a route can be checked in constant time. As pointed out by Cordeau et al. [1], based on Eq. (1), this equation can be rewritten as

$$F_u = \min_{i \in \{u\} \cup \Gamma(u)} \left\{ TWT_{\psi_{\sigma(u),i}} + l_i - h_i \right\}. \quad (2)$$

According to [9], the feasibility of 2-exchanges and or-exchanges for the TSPTW and the VRPTW can be checked in constant time if the variable $TWT_{\psi_{u,v}}$ is known for any pair of vertices (u, v) in the considered route.

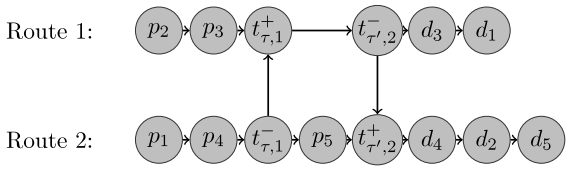


Fig. 1. Example of a precedence graph for two routes interconnected at transfer points τ and τ'.

4. Insertion feasibility for the PDPT

We consider a partial solution s of the PDPT that satisfies the temporal constraints of the problem. Let us denote by $R_s \subset R$ the subset of requests served in solution s. The sets of pickup and delivery vertices in this solution are denoted by P_s and D_s respectively. For each request $r \in R_s$ transferred at a transfer point $\tau \in T$, this point is split into an inbound transfer vertex $t^-_{\tau,r}$, where the request gets off the vehicle, and an outbound transfer vertex $t^+_{\tau,r}$, where the request gets in another vehicle. The sets of inbound and outbound transfer vertices used in s are denoted by T^-_s and T^+_s respectively. A partial solution of the PDPT can be associated with a *precedence graph*, defined below.

Definition 1 (*Precedence Graph*)**.** The precedence graph associated with a solution s of the PDPT is defined by $G_s = (V_s, A_s)$, where

$$V_s = P_s \cup D_s \cup T^-_s \cup T^+_s \cup \{o\} \cup \{o'\}$$

and A_s contains arcs (u, v) such that $v = \sigma(u)$ or u and v are the inbound and outbound transfer points of the same request.

An example of precedence graph is represented in Fig. 1 that models a partial solution with two routes and five requests. In this example, requests 1 and 2 are transferred at points τ and τ' respectively. This is modeled by four vertices: $t^-_{\tau,1}$, $t^+_{\tau,1}$, $t^-_{\tau',2}$ and $t^+_{\tau',2}$.

A *schedule* on G_s defines a service time h_i for each vertex $i \in V_s$. The schedule is *feasible* if the temporal constraints of the PDPT are satisfied for s. As stated before, in this paper we consider the vertices to be scheduled as early as possible and we denote by w_i the waiting time at vertex $i \in V_s$.

Let $r \in R \setminus R_s$ be a request to insert in this partial solution. Inserting r in s at a given position consists of inserting the corresponding vertices and arcs in the precedence graph G_s. This is feasible if the new solution resulting from this insertion has a feasible schedule.

A request r can be inserted without transfer or through a specified transfer point τ. As inserting without transfer is similar to and simpler than inserting with transfers, we provide the feasibility condition for the latter case only. Therefore we consider

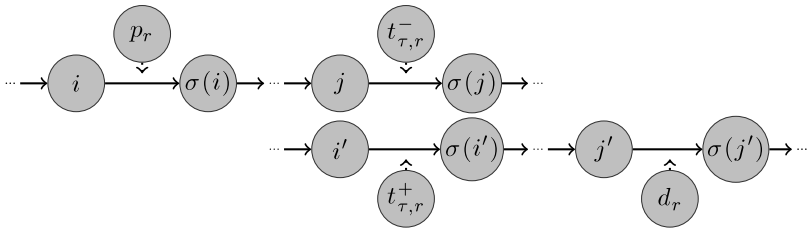


Fig. 2. Example of the attempt to insert a request r through a transfer point τ.

the insertion of four vertices p_r, $t^-_{\tau,r}$, $t^+_{\tau,r}$ and d_r in the precedence graph G_s. Let us assume that these vertices are inserted after some vertices called $i, j, i'j'$ respectively, with i and j being in the route in which p_r is to be inserted and i' and j' being in the route in which d_r is to be inserted. This insertion attempt is illustrated in Fig. 2.

We define $\Omega_{u,v}$ as the set of paths from $u \in V_s$ to $v \in V_s$ in G_s. The set $\Gamma(u)$ is redefined as the set of successors of a vertex u in G_s, that is $\Gamma(u) = \left\{v \in V | \Omega_{u,v} \neq \emptyset\right\}$. Since vertices in G_s are scheduled as early as possible, $\Gamma(u)$ contains all vertices which service time can be impacted by in the service time at vertex u.

Proposition 1. *Let r be a request to insert through a transfer point τ in a partial solution s with precedence graph $G_s = (V_s, A_s)$. The insertion of vertices p_r, $t^-_{\tau,r}$, $t^+_{\tau,r}$, d_r after vertices i, j, i' and $j' \in V'$ is feasible if and only if the service time of all vertices in $\Gamma(i) \cup \Gamma(i') \cup \left\{p_r, t^-_{\tau,r}, t^+_{\tau,r}, d_r\right\}$ – after the insertion – satisfies their time window, precedence constraints at transfer points and travel time constraints.*

5. Forward time slack for the PDPT

We propose to extend the definition of forward time slacks to the PDPT and to redefine F_u for each vertex u in a precedence graph G_s: F_u indicates how far the service at vertex u can be postponed without generating any time window violation.

Note that, in the precedence graph of a solution, there may be several paths between two vertices. In the example shown in Fig. 1, there exist two possible paths between vertices p_4 and d_4 : $(p_4, t^-_{\tau,1}, p_5, t^+_{\tau,2}, d_4)$ and $(p_4, t^-_{\tau,1}, t^+_{\tau,1}, t^-_{\tau,2}, t^+_{\tau,2}, d_4)$. Hence, a postponement of the service time at vertex p_4 can be propagated up to d_4 along both paths.

For every path $\omega \in \Omega_{u,v}$, let us denote by TWT_ω the sum of waiting times along path ω in an "as early as possible schedule" on G_s. The value TWT_ω measures the slack time on path ω which can be extended to the notion of slack time between any pair of vertices in the precedence graph as follows.

Definition 2 (*Slack Time Between Two Vertices*)**.** Let $G_s = (V_s, A_s)$ be a precedence graph and u and v two vertices. If $\Omega_{u,v} \neq \emptyset$, the slack time $ST_{u,v}$ is defined by

$$ST_{u,v} = \min_{\omega \in \Omega_{u,v}} TWT_\omega.$$

Otherwise, $ST_{u,v} = +\infty$.

According to this definition, the maximal time shift, F_u, allowed for service at u without causing any time window violation in the solution can be computed as follows.

Proposition 2 (*DPT Forward Time Slack*)**.** *Let s be a solution of the PDPT; then the forward time slack F_u of a vertex $u \in V_s$ is*

$$F_u = \min_{i \in \{u\} \cup \Gamma(u)} \left\{ST_{u,i} + l_i - h_i\right\}.$$

Proof. Let us consider a vertex $i \in \Gamma(u)$ and a path $\omega \in \Omega_{u,i}$. The set of vertices in the path ω is written as Ψ_ω. The maximal time shift F_ω at u that does not generate time window violation on path ω can be deduced from (2):

$$F_\omega = \min_{j \in \Psi_\omega} \left\{TWT_\omega + l_j - h_j\right\}.$$

Clearly if there are several paths between u and i, the maximal time shift at u that avoids time window violation at i is

$$F_{u,i} = \min_{\omega \in \Omega_{u,i}} \min_{j \in \Psi_\omega} \left\{TWT_\omega + l_j - h_j\right\}.$$

Let $\Psi_{u,i} = \bigcup_{\omega \in \Omega_{u,i}} \Psi_\omega$ denote the set of nodes in a path between u and i in G_s. Then, from Definition 2,

$$\begin{aligned} F_{u,i} &= \min_{j \in \Psi_{u,i}} \min_{\omega \in \Omega_{u,j}} \left\{TWT_\omega + l_j - h_j\right\} \\ &= \min_{j \in \Psi_{u,i}} \left\{ST_{u,j} + l_j - h_j\right\}. \end{aligned}$$

Knowing that a time shift at u can only cause a violation at u or at one of its successors in G_s, we have

$$F_u = \min_{i \in \{u\} \cup \Gamma(u)} F_{u,i}$$

which is equivalent to (2). □

6. Insertion of a request without transfers

Inserting a vertex in G_s is likely to postpone the service at its successors. When two or more vertices are to be inserted in G_s, the new service times at successor vertices should be computed. Let us first evaluate the insertion of a request r with no transfer.

Based on the definition of slack times (Definition 2), the new service time at some vertex $v \in V_s$ resulting from a postponement of $u \in V_s$ can be computed according to Proposition 3.

Proposition 3. *Let $G_s = (V_s, A_s)$ be a precedence graph and two vertices $u \in V_s$ and $v \in V_s$ with service times h_u and h_v respectively. If h_u is postponed for δ, then the new service time $\bar{h}_v$ is*

$$\bar{h}_v = h_v + \max(\delta - ST_{u,v}, 0). \tag{3}$$

Proof. If $v \notin \Gamma(u)$, the result is trivial. Otherwise, the proof follows a similar reasoning to the proof of Proposition 2. □

Based on Propositions 2 and 3, we define the function *EvaluateInsertion* described in Algorithm 1. This function checks in constant time if the insertion of a vertex v after a vertex i serviced at time h_i is feasible. If the insertion is feasible, it returns the value $\delta_{\sigma(i)}$ of the time shift at vertex $\sigma(i)$, and $+\infty$ otherwise.

As a result, inserting vertices p_r and d_r without transfer in a partial solution s can be evaluated in constant time using Algorithm 2. This algorithm evaluates the insertion of p_r after some vertex $i \in V_s$ and the insertion of d_r after $j \in V_s$, i and j being serviced in this order in the same route. It returns *true* if the insertion is feasible, and *false* otherwise.

7. Insertion of a request with transfer

Let us consider the insertion depicted in Fig. 2. Inserting a request r through a transfer point τ consists of inserting four nodes in the precedence graph. This destroys at most four arcs (where p_r, $t^-_{\tau,r}$, $t^+_{\tau,r}$ and d_r are inserted) and creates at most nine arcs, modeling precedence relations in the two modified routes as well as the precedence relation between transfer points.

Algorithm 1: EvaluateInsertion(v, i, h_i)

Result: return $\delta_{\sigma(i)}$ if inserting vertex v after vertex i serviced at time h_i is feasible, $+\infty$ otherwise

/* Evaluate the service time at v */
$\bar{h}_v = max(e_v, h_i + s_i + \theta_{i,v})$;
if $\bar{h}_v > l_v$ **then**
  return $+\infty$
/* Compute the time shift at $\sigma(i)$ */
$\delta_{\sigma(i)} = max(\bar{h}_v + s_v + \theta_{v,\sigma(i)} - h_{\sigma(i)}, 0)$;
if $\delta_{\sigma(i)} > F_{\sigma(i)}$ **then**
  return $+\infty$
else
  return $\delta_{\sigma(i)}$.

Algorithm 2: Insertion without transfer

Result: return **true** if inserting vertex p_r after vertex i and inserting vertex d_r after vertex j is feasible, **false** otherwise

/* Evaluate the insertion of p_r */
$\delta_{\sigma(i)} = EvaluateInsertion(p_r, i, h_i)$;
if $\delta_{\sigma(i)} = +\infty$ **then**
  return false
/* Compute the service time at j */
$\bar{h}_j = h_j + max(\delta_{\sigma(i)} - ST_{\sigma(i),j}, 0)$;
/* Evaluate the insertion of d_r */
$\delta_{\sigma(j)} = EvaluateInsertion(d_r, j, \bar{h}_j)$;
if $\delta_{\sigma(j)} = +\infty$ **then**
  return false
else
  return true.

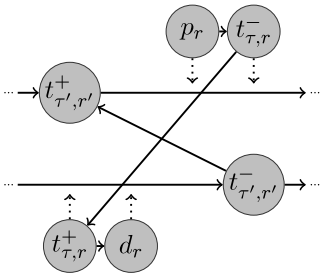


Fig. 3. Example of an insertion introducing a cycle in the precedence graph.

7.1. Detection of cycles in the precedence graph

As illustrated in Fig. 3, the presence of multiple transfers may introduce cycles in the precedence graph. Cycles imply a precedence relation between a vertex and itself, which is clearly inconsistent.

The feasibility test has to efficiently detect such cycles. Proposition 4 gives a necessary and sufficient condition for the creation of a cycle in a precedence graph.

Proposition 4. *The insertion of a request r composed of vertices $p_r, t^-_{\tau,r}, t^+_{\tau,r}$ and d_r, after vertices i, j, i' and j' respectively in a precedence graph G_s, introduces a cycle in G_s if and only if $j \in \Gamma(\sigma(i'))$.*

Proof. If $j \in \Gamma(\sigma(i'))$ and $t^-_{\tau,r}$ is inserted after j, then $t^-_{\tau,r} \in \Gamma(\sigma(i'))$. If $t^+_{\tau,r}$ is inserted after i', then $i' \in \Gamma(t^+_{\tau,r})$. Hence, $t^-_{\tau,r} \in \Gamma(t^+_{\tau,r})$. By definition of $t^-_{\tau,r}$ and $t^+_{\tau,r}$, $t^+_{\tau,r} \in \Gamma(t^-_{\tau,r})$. This introduces a cycle in G_s.

Conversely, suppose that $j \notin \Gamma(\sigma(i'))$, then $t^-_{\tau,r} \notin \Gamma(\sigma(i'))$. Therefore $t^-_{\tau,r} \notin \Gamma(t^+_{\tau,r})$ which proves that no cycle has been introduced in G_s. □

From Proposition 4, checking if the insertion of request r introduces a cycle is equivalent to checking if $j \in \Gamma(\sigma(i'))$. This information can be accessed in $O(1)$ provided that it is stored in an appropriate data structure indexed by j and $\sigma(i')$, for example, a table of dimension $|V| \times |V|$.

7.2. Calculation of service times

We now detail the process for calculating the service times while checking the feasibility of request insertion. This calculation requires constant time, by using Eq. (3). The process is decomposed in four stages, consisting of sequentially updating the service times after insertion of vertices p_r, $t^-_{\tau,r}$, $t^+_{\tau,r}$ and d_r respectively. This process is detailed in Algorithm 3, which returns *true* if the insertion of p_r after i, $t^-_{\tau,r}$ after j, $t^+_{\tau,r}$ after i' and d_r after j' is feasible, and *false* otherwise.

Algorithm 3: Insertion with transfer

Result: return **true** if the insertions of p_r after i, $t^-_{\tau,r}$ after j, $t^+_{\tau,r}$ after i' and d_r after j' are feasible, **false** otherwise

/* Evaluate the insertion of p_r */
$\delta_{\sigma(i)} = EvaluateInsertion(p_r, i, h_i)$;
if $\delta_{\sigma(i)} = +\infty$ **then**
  return false
/* Evaluate the service time at j */
$\bar{h}_j = h_j + max(\delta_{\sigma(i)} - ST_{\sigma(i),j}, 0)$;
/* Evaluate the insertion of $t^-_{\tau,r}$ */
$\delta_{\sigma(j)} = EvaluateInsertion(t^-_{\tau,r}, j, \bar{h}_j)$;
if $\delta_{\sigma(j)} = +\infty$ **then**
  return false
/* Evaluate the service time at i' */
$\bar{h}_{i'} = h_{i'} + max(\delta_{\sigma(i)} - ST_{\sigma(i),i'}, \delta_{\sigma(j)} - ST_{\sigma(j),i'}, 0)$;
/* Evaluate the service time at $t^+_{\tau,r}$ */
$\bar{h}_{t^+_{\tau,r}} = max(e_{t^+_{\tau,r}}, \bar{h}_{i'} + s_{i'} + \theta_{i',t^+_{\tau,r}}, \bar{h}_{t^-_{\tau,r}} + s_{t^-_{\tau,r}})$;
if $\bar{h}_{t^+_{\tau,r}} > l_{t^+_{\tau,r}}$ **then**
  return false
/* Compute the time shift at $\sigma(i')$ */
$\delta_{\sigma(i')} = max(\bar{h}_{t^+_{\tau,r}} + s_{t^+_{\tau,r}} + \theta_{t^+_{\tau,r},\sigma(i')} - h_{\sigma(i')}, 0)$;
if $\delta_{\sigma(i')} > F_{\sigma(i')}$ **then**
  return false
/* Evaluate the service time at j' */
$\bar{h}_{j'} = h_{j'} + max(\delta_{\sigma(i)} - ST_{\sigma(i),j'}, \delta_{\sigma(j)} - ST_{\sigma(j),j'}, \delta_{\sigma(i')} - ST_{\sigma(i'),j'}, 0)$;
/* Evaluate the insertion of d_r */
$\delta_{\sigma(j')} = EvaluateInsertion(d_r, j', \bar{h}_{j'})$;
if $\delta_{\sigma(j')} = +\infty$ **then**
  return false
else
  return true

The 11 service time calculations required to test an insertion are performed in constant time. Theorem 1 formulates the main result of this article.

Theorem 1. *The feasibility of a request insertion in a partial solution of the PDPT can be evaluated in constant time.*

8. Preprocessing

In the preceding sections, we showed that the feasibility check can be performed in constant time under three conditions: (i) all service times are scheduled as early as possible; (ii) the values F_v for all $v \in V$ and the set of successors $\Gamma(v)$ are updated after each request insertion or removal; (iii) the value of $ST_{u,v}$ are updated after each request insertion or removal, for all vertices $u \in V_s$ and $v \in V_s$ in the precedence graph.

If these conditions are not met, preprocessing steps must be performed. The calculation of values F_v can be seen as computing the shortest path between one vertex and all other vertices in a direct acyclic graph. This can be done in $O(|V|)$ [2]. Scheduling times of service as early as possible and recomputing the set of successors can also be performed in linear time.

Value $ST_{u,v}$ is the length of a shortest path between u and v with weight on each arc of the precedence graph G_s defined as the waiting time at its terminal extremity. Since the precedence graph is a directed acyclic graph, a shortest path between a vertex and all other vertices can be computed in a linear time. Therefore, the complexity of computing all values $ST_{u,v}$ is quadratic in $|V|$.

9. Conclusion

In this article, we presented a method for testing in constant time the feasibility of a request insertion in a solution of the PDPT. This method extends the concept of Forward Time Slack [9]. The constant time complexity requires some preprocessing each time a current solution of the problem is actually updated. This preprocessing has a quadratic complexity in the number of vertices of the precedence graph, which is efficient provided that the number of actual updates is substantially smaller than the number of feasibility checks.

References

[1] J.-F. Cordeau, G. Laporte, A. Mercier, Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows, Journal of the Operational Research Society 55 (2004) 542–546.
[2] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, MIT Press, 2001.
[3] C.E. Cortés, M. Matamala, C. Contardo, The pickup and delivery problem with transfers: formulation and a branch-and-cut solution method, European Journal of Operational Research 200 (3) (2010) 711–724.
[4] M. Drexl, Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints, Transportation Science 46 (3) (2012) 297–316.
[5] M. Firat, G.J. Woeginger, Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh, Operations Research Letters 39 (1) (2011) 32–35.
[6] B. Hunsaker, M. Savelsbergh, Efficient feasibility testing for dial-a-ride problems, Operations Research Letters 30 (2002) 169–173.
[7] S. Mitrović-Minić, G. Laporte, The pickup and delivery problem with time windows and transshipment, INFOR 44 (3) (2006) 217–228.
[8] Y. Qu, J.F. Bard, A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment, Computers & Operations Research 39 (10) (2012) 2439–2456.
[9] M.W.P. Savelsbergh, The vehicle routing problem with time windows: minimizing route duration, ORSA Journal on Computing 4 (2) (1992) 146–154.

A New Consistent Vehicle Routing Problem for the Transportation of People with Disabilities

Dominique Feillet and Thierry Garaix
Ecole des Mines de Saint-Etienne, CMP Georges Charpak, F-13541, Gardanne, France

Fabien Lehuédé and Olivier Péton
IRCCyN, Ecole des Mines de Nantes, 4 rue Alfred Kastler, F-44307, Nantes, France

Dominique Quadri
Université d'Avignon et des Pays du Vaucluse, Laboratoire Informatique d'Avignon, F-84911, Avignon, France

In this article, we address a problem of the transportation of people with disabilities where customers are served on an almost daily basis and expect some consistency in the service. We introduce an original model for the time-consistency of the service, based on so-called time-classes. We then define a new multiday vehicle routing problem (VRP) that we call the Time-Consistent VRP. We address the solution of this new problem with a large neighborhood search heuristic. Each iteration of the heuristic requires solving a complex VRP with multiple time windows and no waiting time which we tackle with a heuristic branch-and-price method. Computational tests are conducted on benchmark sets and modified real-life instances. Results demonstrate the efficiency of the method and highlight the impact of time-consistency on travel costs. **© 2013 Wiley Periodicals, Inc. NETWORKS, Vol. 000(00), 000–000 2013**

Keywords: consistent vehicle routing problem; transportation of people with disabilities; multiple time windows; heuristic branch-and-price

1. INTRODUCTION

Among the diverse categories of problems that have been investigated in operations research, vehicle routing problems (VRPs) probably constitute one of the most emblematic class. In practice, their purpose is generally to propose optimized distribution plans while delivering goods to customers (e.g., be they individuals, firms, or retailers). In most of the many models developed for the routing of vehicles, a single period of time is considered (say, a day). This modeling, however, corresponds to two very different practical situations: either the routes are intended to be performed once and recomputed with new data the next day, or the routes are repeated identically every single day over a long time horizon. However, there is an intermediate situation in which demands are almost identical from day to day. In addition to the usual cost minimization objective, a new criterion might then make sense: limiting the variability of the routes between the different periods of the time horizon. The problem then becomes a multiday VRP where routes are expected to show some regularity from day to day.

An important illustration of this problem was given by Groër et al. [14] and concerns fast parcel delivery (involving big companies such as UPS). The ability to assure consistent schedules and to limit the number of drivers that a customer has to deal with can significantly improve the satisfaction of these customers: facilitating receipt operations, improving trust between the delivery companies and their customers, and so forth. A major advantage can thus be provided in comparison with competing companies.

Another interesting case, which motivated this article, concerns the transportation of people with mental disabilities from and to day care centers [19]. Because of their lack of autonomy, most of the persons concerned have no vehicle and are not capable of traveling by public transport. Consequently, medicosocial centers usually resort to specialized transportation companies to organize and perform the daily trips. Customers need to be transported regularly but sometimes with some small variations during the week. For example, a person may not ask for transportation or may not have the same address every day. In addition, these users

Received March 2013; accepted August 2013
Correspondence to: D. Feillet and T. Garaix; e-mail: feillet,garaix@emse.fr
Contract grant sponsor: GdR RO (Operations Research Group of the Scientific Research National Center)
DOI 10.1002/net.21538
Published online in Wiley Online Library (wileyonlinelibrary.com).

are particularly sensitive to changes. The issue of keeping consistent time schedules is therefore fundamental.

In this article, we investigate how consistency can be considered in VRPs. Because the expectations of stability are context-dependent, as will be detailed below, we essentially address the transportation of people with disabilities and the stability of the time of services. Because there is a paucity of works on this subject, we first discuss the notion and the modeling of consistency, before introducing a new model (section 2). We then introduce the time-consistent VRP (TCVRP) and propose a solution approach based on a large neighborhood search (LNS) (section 3). Computational experiments evaluating the behavior of this algorithm are then described (section 4), before the article is concluded (section 5).

2. MODELING THE TIME-CONSISTENCY OF ONE CUSTOMER

The issue of constructing consistent vehicle routes is related to many important lines of research in operations research such as robustness or stochasticity [3, 13]. However, consistency is generally sought in relation to unexpected events that disrupt the transport.

In contrast, we consider here the consistency of services with respect to deterministic data with slight variations from day to day, which has rarely been investigated in a routing context. Our study mainly compares with the recent paper of Groër et al. [14] which introduces the consistent VRP. Since its first online publication, this problem has been investigated in [18, 22, 27]. The consistent vehicle problem clearly shares some similarities with the courier delivery problem [26] which, in addition, considers time windows, uncertain service times, uncapacitated vehicles and that frequent customers should not necessarily be served by a single vehicle. A few other studies show some connections with this subject [1, 2, 28]. Finally, the consistent VRP has some relationship with multiperiod VRPs [12], where the assignment of visit days is also considered.

From the customer point of view, consistency improves the quality of service. Having consistent schedules helps the customer to get prepared for the service, to manage material flows, to schedule upstream and downstream tasks, and so forth. Assigning the same driver to the different visits of a customer facilitates the delivery process and strengthens the relationship between suppliers and customers [6, 16, 24]. Such criteria can play an essential role in the choice of a supplier [5].

For drivers, having consistent sequences of visits also presents many advantages. It helps develop a good knowledge of the usual itineraries and traffic conditions, thus enabling drivers to better estimate distribution times and potential delays, reducing transportation times [6, 15, 24, 25]. Driver familiarity with the customer also helps in applying security and administrative procedures [16]. With all these improvements, the transportation process is generally considered more comfortable by the drivers [25].

In the remainder of the article, we essentially restrict the study to the consistency of schedules. Our assumption is that once a time-consistent solution is computed, finding a reasonably consistent assignment of drivers should be possible. This assumption is checked experimentally in section 4. The issue of simultaneously optimizing time- and driver-consistency is left for future research.

To discuss the modeling of time-consistency, let us consider a customer i, a time horizon $D = \{1, \ldots, |D|\}$, and times of service (or schedule) $W = (w_1, \ldots, w_{|D|})$ for customer i over this horizon: the service of customer i starts at time w_d on day d. For the sake of simplicity, note that we assume here that customer i is visited every day of the time horizon. Note also that W is assumed to be known. The method to optimize W will be the subject of section 3.

We first review the model proposed by Groër et al. [14] and then describe our model which generalizes the former.

Note that we do not consider explicitly customers with several addresses. Actually, time-consistency is important for a given address. A person with two distinct addresses is modeled as two people (one per address).

2.1. Modeling Time-Consistency Through Time Slacks

The time-consistency for customer i is ensured in [14] by limiting the difference between the latest and earliest service dates. More formally, a time of service variation is measured as:

$$\mathrm{TC}_{\mathrm{slack}}(W) = \max_{1 \le d \le |D|} w_d - \min_{1 \le d \le |D|} w_d \qquad (1)$$

This definition is relevant in the context of the small package shipping industry. Figure 1 illustrates for five different schedules $W_1, \ldots, W_5$, the measure of time-consistency following Groër et al. for a horizon ranging from Monday to Friday.

However, considering other application contexts, the preceding modeling needs to be questioned. A first remark is that it is particularly suitable for small variations in service times. Thus, schedules $W_1 = (8{:}00, 8{:}03, 8{:}05, 8{:}00, 8{:}02)$ and $W_2 = (8{:}00, 8{:}00, 8{:}05, 8{:}05, 8{:}00)$, such that $\mathrm{TC}_{\mathrm{slack}}(W_1) = \mathrm{TC}_{\mathrm{slack}}(W_2) = 5$, can be considered equivalent in terms of time-consistency. However, it is less clear for $W_3 = (8{:}00, 9{:}00, 8{:}20, 8{:}30, 8{:}40)$ and $W_4 = (8{:}00, 8{:}02, 8{:}05, 8{:}58, 9{:}00)$. Although $\mathrm{TC}_{\mathrm{slack}}(W_3) = \mathrm{TC}_{\mathrm{slack}}(W_4) = 60$, W_4 shows much more regularity, with two sets of almost identical times of service around 8:00 and 9:00.

Following this observation, a second important remark is that while W_1 or W_2 undoubtedly define more consistent schedules than W_4, should the two schedules W_4 and $W_5 = (8{:}00, 8{:}02, 8{:}05, 9{:}58, 10{:}00)$, with $\mathrm{TC}_{\mathrm{slack}}(W_4) = 60$ and $\mathrm{TC}_{\mathrm{slack}}(W_5) = 120$ respectively, be considered so different? In the following section and in the remainder of this article, we introduce a new modeling framework for time-consistency which assumes that such schedules are equivalent.

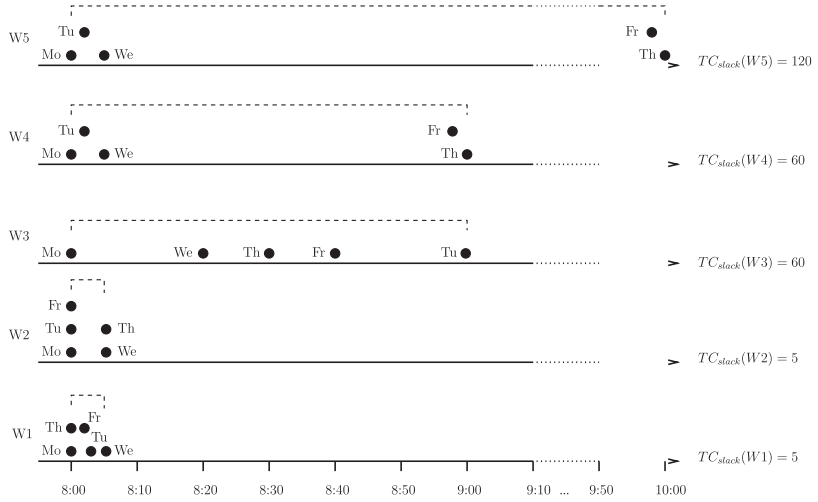


FIG. 1. Time-consistency measure of Groër et al. [14].

As could be expected, Groër et al. [14] attain a very high level of consistency in their solutions, thus justifying their modeling. Two factors can be considered determinant in this behavior. First, vehicles deliver 100 to 150 customers everyday. Second, many of these customers are visited only once during the time horizon, thus giving many opportunities for substitutions in the routes. As a consequence, consistent schedules can be preserved for the relevant customers, giving rise to a strong time-consistency and mostly avoiding being concerned by the above remarks. From an algorithmic perspective, the chosen approach was to define template routes that respect the so-called precedence principle: "If customers i and j are both served by the same vehicle on a specific day and i is serviced before j, then customer i must receive service before customer j from the same vehicle on all days where they both require service." Note that this principle also deals with driver-consistency.

For the transportation of people with disabilities, the situation is very different. Vehicles only visit a very small number of people, who make transportation requests almost everyday, thus rendering substitution between customers much more complicated and increasing the differences between the service times. This explains why we introduced the modeling developed below.

2.2. *Modeling Consistency Through Time-Classes*

From the above discussion, we introduce a new consistency measure that follows these principles: a small gap between service times is not significant; a large gap induces a disturbance, whatever the gap.

Let us consider the parameter $\delta > 0$ reflecting the sensitivity of the customers to schedule variations. To evaluate the time-consistency for customer i, we define the notion of time-classes as follows: the times of service w_d and $w_{d'}$ are allowed to be assigned to the same time-class if $|w_d - w_{d'}| \leq \delta$. Hence, the times of service $\{w_1, \ldots, w_{|D|}\}$ can be divided into time-classes.

It can easily be seen from the above definition that there is no unique way of assigning a set of times of service to time-classes. Let us go back to the example of schedule $W_3 = (8{:}00,\ 9{:}00,\ 8{:}20,\ 8{:}30,\ 8{:}40)$. With parameter $\delta = 10$ one can, for example, construct time-classes {{Mo}, {We,Th},{Fr},{Tu}} or {{Mo},{We},{Th,Fr},{Tu}}.

We define the time-consistency $\mathrm{TC}(W)$ of a schedule $W = (w_1, \ldots, w_{|D|})$ as the minimal number of time-classes into which $w_1, \ldots, w_{|D|}$ can be ranked:

$$\mathrm{TC}\,(W) = \min C \tag{2}$$

subject to

$$|w_d - w_{d'}| > \delta \Rightarrow C_d \neq C_{d'} \quad (d, d' \in D), \tag{3}$$

$$C_d \leq C \quad (d \in D), \tag{4}$$

$$C_d \in \{1, \ldots, |D|\} \quad (d \in D), \tag{5}$$

$$C \in \{1, \ldots, |D|\}. \tag{6}$$

Constraints (3) impose the assignment of different classes for service times whose difference is greater than δ. Constraints (4) with objective function (2) compute the minimal number of classes required C.

Figure 2 outlines the proposed consistency through time-classes considering schedules W_1 to W_5 introduced in subsection 2.1, with parameter $\delta = 10$. Note that with this new measure, schedules W_1 and W_2 are still the most consistent, schedules W_4 and W_5 are considered equivalent, and schedule W_3 is strongly penalized. The method to compute values $TC(W)$ and to assign visits to time-classes is presented in the following subsection.

2.3. *Computation of Time-Classes*

It can easily be seen that model (2)–(6) defines a graph coloring problem. Thus, undirected graph $G = (D, E)$ could be introduced, where edge $(d, d') \in E$ if and only if $|w_d - w_{d'}| > \delta$. Computing the minimal number of time-classes for

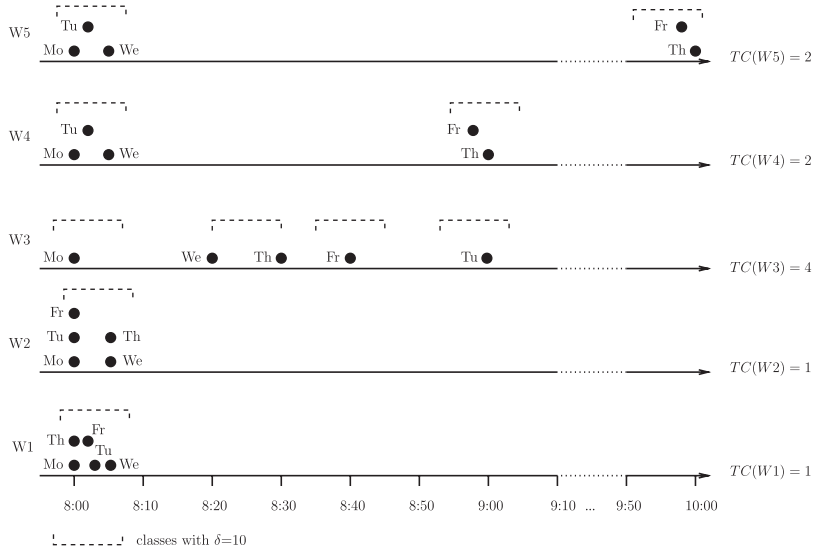


FIG. 2. Time-class consistency.

schedule W then amounts to coloring the vertices in G, such that two vertices connected with an edge are colored with a different color.

Although graph coloring problems are generally NP-hard, graph G exhibits a very special structure which implies that the problem is polynomial. This is highlighted by the subsequent equivalent definition of set E: edge $(d, d') \in E$ if and only if $[w_d - \delta/2, w_d + \delta/2] \cap [w_{d'} - \delta/2, w_{d'} + \delta/2] = \emptyset$. As such, the coloring problem can be solved using a greedy algorithm (Algorithm 1). In Algorithm 1, vertices are colored in the increasing order of times of service w_d , with the first available color. Let d represent the first vertex colored with the current color. A sufficient condition for a vertex to be colorable using the current color is to be compatible with vertex d.

The time-classes that would be constructed for schedules W_1 to W_5 with $\delta = 10$, applying Algorithm 1, can be observed in Fig. 2.

Algorithm 1 Computation of time-classes

$\{d_1, \ldots, d_{|D|}\} \leftarrow$ vertices $\{1, \ldots, |D|\}$ sorted in increasing order of times of servicecolor vertex d_1 with color 1
$d \leftarrow d_1$
for $i = 2$ to $|D|$ **do**
  if edge $(d, d_i) \notin E$ **then**
    color vertex d_i with the same color as d
  else
    color vertex d_i with the next color
    $d \leftarrow d_i$
  end if
end for

3. THE TCVRP

In the remainder of the article, we first assess how the proposed definition of time-consistency can be handled in a solution approach. Second, we evaluate the impact of considering time-consistency on vehicle routes.

The original application includes various specific constraints and criteria that can complicate the readability of the subsequent results. We thus introduce and address a simplified problem. This is called the TCVRP and is basically a multiday VRP with the additional constraint of optimizing time-consistency over days.

An important difference to be noted compared to previous sections is that customers are initially located at the depot and request to be transported back home with regular arrival times of services, with a set of vehicles all departing at the same time (time 0) from the depot. In the real-application that motivates this article, the people with mental disabilities all had to arrive at a given time and expected to be picked up at regular times. From a computational point of view, however, the two situations are completely symmetric and equivalent.

We first define the TCVRP in section 3.1. Then, the proposed solution algorithm is outlined in section 3.2. Finally, sections 3.3, 3.4, and 3.5 provide details on the different steps of the algorithm.

3.1. *Definition of the TCVRP*

Let us denote D the set of days in the time horizon and consider a complete graph $G = (V, A)$, where $V = \{0, \ldots, n\}$ represents a depot (vertex 0) and the destinations of n customers (vertices 1 to n). Customers need to be transported from the depot to their destination on some days of the time horizon, defined by $p_i^d = 1$ if customer i has to be transported on day d, $p_i^d = 0$ otherwise. Set K represents a fleet of identical vehicles available at the depot every day of the time horizon. The capacity of a vehicle is limited to Q customers. A travel time $t_{ij} > 0$ is defined for every arc $(i, j) \in A$. This travel time corresponds equivalently, without loss of generality, to a travel cost. A parameter δ is given to represent the

maximal difference of times of service that can be ranked in the same time-class.

Two objectives are pursued. The first is to minimize travel costs, that is, the sum of the route costs throughout the horizon. The second is to maximize time-consistency, that is, to minimize the maximal number of time-classes over the n customers. The number of time-classes for a customer is defined in equations (2)–(6), where times of service are related to routing decisions.

Note that no time windows are defined for deliveries. In addition, waiting times are not allowed. Without this constraint, waiting could improve time-consistency.

The TCVRP then consists in optimizing these two objectives by finding a set of at most $|K|$ routes every day of the time horizon D such that (i) all the requested deliveries are performed, (ii) all routes satisfy vehicle capacities. Seeing that D is assumed to remain quite small (typically a week), the problem can rather be stated as the problem of solving $|D|$ monoobjective subproblems, where the maximal number of time-classes is limited and successively defined as $C = 1, \ldots, |D|$. The model (7)–(25) presented below shows a mathematical formulation for one of these monoobjective subproblems, with a limited number of time-classes, namely at most C time-classes per customer. Note, however, that the solution method and experiments presented later deal with the complete (multiobjective) TCVRP. The model is as follows:

$$\text{Min } z = \sum_{d\in D}\sum_{k\in K}\sum_{(i,j)\in A} t_{ij}x_{ijk}^{d} \tag{7}$$

s.t.

$$\sum_{k\in K}\sum_{j\in V} x_{ijk}^{d} = p_i^{d} \quad (i \in V\backslash\{0\}, d \in D), \tag{8}$$

$$\sum_{j\in V} x_{jik}^{d} - \sum_{j\in V} x_{ijk}^{d} = 0 \quad (i \in V\backslash\{0\}, k \in K, d \in D), \tag{9}$$

$$\sum_{i\in V\backslash\{0\}} x_{0ik}^{d} = 1 \quad (k \in K, d \in D), \tag{10}$$

$$\sum_{i\in V\backslash\{0\}} x_{i0k}^{d} = 1 \quad (k \in K, d \in D), \tag{11}$$

$$\sum_{i\in V\backslash\{0\}}\sum_{j\in V} x_{ijk}^{d} \le Q \quad (k \in K, d \in D) \tag{12}$$

$$w_{jk}^{d} \le w_{ik}^{d} + t_{ij} + (1 - x_{ijk}^{d})M \quad ((i,j) \in A, k \in K, d \in D), \tag{13}$$

$$w_{jk}^{d} \ge w_{ik}^{d} + t_{ij} - (1 - x_{ijk}^{d})M \quad ((i,j) \in A, k \in K, d \in D), \tag{14}$$

$$w_{i}^{d} \le w_{ik}^{d} + (1 - \sum_{j\in V} x_{ijk}^{d})M \quad (i \in V\backslash\{0\}, k \in K, d \in D), \tag{15}$$

$$w_{i}^{d} \ge w_{ik}^{d} - (1 - \sum_{j\in V} x_{ijk}^{d})M \quad (i \in V\backslash\{0\}, k \in K, d \in D), \tag{16}$$

$$w_{0k}^{d} = 0 \quad (k \in K, d \in D), \tag{17}$$

$$w_{i}^{d_2} - w_{i}^{d_1} \le \delta + My_{i}^{d_1 d_2} \quad (i \in V\backslash\{0\}, d_1, d_2 \in D), \tag{18}$$

$$c_{i}^{d_1} + 1 \le c_{i}^{d_2} + M(1 - y_{i}^{d_1 d_2}) \quad (i \in V\backslash\{0\}, d_1, d_2 \in D), \tag{19}$$

$$y_{i}^{d_1 d_2} \in \{0,1\} \quad (i \in V\backslash\{0\}, d_1, d_2 \in D), \tag{20}$$

$$c_{i}^{d} \le C \quad (i \in V, d \in D), \tag{21}$$

$$x_{ijk}^{d} \in \{0,1\} \quad ((i,j) \in A, k \in K, d \in D), \tag{22}$$

$$w_{ik}^{d} \ge 0 \quad (i \in V\backslash\{0\}, k \in K, d \in D), \tag{23}$$

$$w_{i}^{d} \ge 0 \quad (i \in V\backslash\{0\}, d \in D), \tag{24}$$

$$c_{i}^{d} \in \{1, \ldots, C\} \quad (i \in V, d \in D). \tag{25}$$

The decision variables have the following interpretation:

x_{ijk}^{d}: 1 if arc (i,j) is used by vehicle k on day d, 0 otherwise,
w_{i}^{d}: time of service for customer i on day d,
w_{ik}^{d}: time of service for customer i on day d if it is carried out by vehicle k,
c_{i}^{d}: number of the time-class assigned to the delivery of customer i on day d.

The intermediate binary variables $y_{i}^{d_1 d_2}$ are set to 1 if the times of service of customer i on days d_1 and d_2 are increasing in this order and they are not allowed to belong to the same time-class ($w_{i}^{d_2} - w_{i}^{d_1} > \delta$), and 0 otherwise.

Constraints (8) state that customer i is serviced on day d if and only if he or she needs to be transported on that day. Constraints (9) are classical flow conservation constraints. Constraints (10) and (11) enable vehicles to leave the depot and return to the depot once a day at most, respectively. Constraints (12) present vehicle capacities. Constraints (13) and (14) define time variables. They linearize the logical expression

$$\forall (i,j) \in A, \forall k \in K, \forall d \in D, x_{ijk}^{d} = 1 \Rightarrow w_{jk}^{d} = w_{ik}^{d} + t_{ij}.$$

Note that vehicles are not allowed to wait to improve time-consistency. Constraints (15) and (16) define times of service for customers, independently of the vehicle used. They linearize the logical expression

$$\forall i \in V\backslash\{0\}, \forall k \in K, \forall d \in D, \sum_{j\in V} x_{ijk}^{d} = 1 \Rightarrow w_{i}^{d} = w_{ik}^{d}.$$

Constraints (17) set vehicle starting times at the depot. For every customer, constraints (18)–(20) enforce the assignment of distinct time-classes to service times that differ from more than δ. They linearize the logical expression

$$\forall i \in V\backslash\{0\}, \forall d_1, d_2 \in D, w_{i}^{d_2} - w_{i}^{d_1} > \delta \Rightarrow c_{i}^{d_1} + 1 \le c_{i}^{d_2}.$$

Constraints (21) limit the maximum number of time-classes allowed for every customer.

Finally, we note that in the case of $C \geq |D|$, constraints (18)–(20), (21), and (25) can be removed since every day can be solved independently. Furthermore, the case $C = 1$ can also be simplified by replacing those constraints by

$$|w_i^{d_2} - w_i^{d_1}| \leq \delta \quad (i \in V \backslash \{0\}, d_1, d_2 \in D). \tag{26}$$

3.2. Solution of the TCVRP with a LNS Approach

In this section, we describe the framework of the algorithm proposed for the solution of the TCVRP. Subsequent sections explain the different procedures introduced here.

Time-consistency constraints significantly complicate the problem as they introduce temporal dependences between routes. Very few VRPs in the literature exhibit similar features. The most closely related problems we were able to find are those with synchronization constraints [4, 17]. As pointed out by Rousseau et al. [21], synchronization constraints considerably limit the applicability of traditional local search techniques. Assessing the feasibility or impact of a very small move can be a large and complex problem. From this observation, we decided to address the solution of the TCVRP with a LNS approach.

The LNS heuristic, first introduced by Shaw [23] has been successfully applied to VRPs in recent years [8, 20]. Although our implementation differs from the initial algorithm of Shaw, the proposed algorithm follows the fundamental principle of LNS: the heuristic starts from a feasible solution and alternates between the destruction of a large part of the solution and the reconstruction of the resulting partial solution.

The main principle of the proposed heuristic is to start from a solution that is good from the cost point of view and to reduce the number of time-classes progressively, recording the best found solution for each level of time-consistency (thus solving the bi-objective version of the TCVRP). To diversify the search, a perturbation mechanism and a restart procedure are performed periodically.

The initial solution is computed by considering each day independently and solving (heuristically) the corresponding VRPs. The solution obtained tends to minimize the total distance traveled during the time horizon. As time-consistency is completely ignored, it is likely that some customers will have a large number of time-classes.

At every iteration, the destruction phase selects one day for which all routes are destroyed. The reconstruction phase relies on the definition of time constraints that enable the maximal number of time-classes for the customers to be controlled. Our approach is to select a customer whom we attempt to reduce the number of time-classes and to limit the number of time-classes for other customers, to varying degrees depending on their distance from the current maximal number of time-classes. These constraints can be expressed in the form of multiple time windows associated with customers. The problem to solve is then a variant of the VRP that we call the VRP with multiple time windows and no-wait constraints (VRPmTW-nw). This problem is tackled with a heuristic branch-and-price solution method.

Algorithm 2 outlines the main steps of the LNS heuristic.

Algorithm 2 General scheme of the LNS algorithm

1: $sol \leftarrow$ initial feasible solution
2: $D' \leftarrow D$
3: **repeat**
4: every p_1 useless iterations without improvement: diversify or restart
5: select a day d at random in D'
6: select a customer i with a maximum number of time-classes among those served on day d
7: define an instance of the VRPmTW-nw for day d according to sol and i
8: $s \leftarrow$ solution of the VRPmTW-nw or **null** if infeasible
9: **if** $s \neq$ **null** then
10: $sol \leftarrow$ replace routes of day d in sol by the routes of s
11: **end** if
12: **if** $D' \backslash \{d\} \neq \emptyset$ **then** $D' \leftarrow D' \backslash \{d\}$ **else** $D' \leftarrow D$
13: **until** a computing time limit is met

The LNS algorithm is initialized (line 1) by solving the VRP defined every day. For this purpose, we apply the heuristic branch-and-price algorithm that is used for reconstruction (see section 3.4), initialized with routes obtained by applying the Clarke and Wright algorithm. In this case, no time windows are defined for the customers. The main loop (Lines 3 to 13) alternates between the selection of a day to destroy and the reconstruction phase.

Day selection is done at random (line 5), ensuring, however, that all days have been selected before one is destroyed again (line 12). In line 6, a customer is selected to focus time-class reduction on this customer. The one chosen has a maximal number of time-classes among customers served on that day. From the customers who satisfy this criterion, we select one for whom the time-class involved on that day has the lowest cardinality. The purpose is to give more chances of deleting that time-class in future iterations.

Following the selection of a day and a customer, an instance of the VRPmTW-nw is defined (line 7). The way this instance is constructed is presented in section 3.3. The heuristic branch-and-price method used for the solution of the VRPmTW-nw (line 8) is detailed in section 3.4. When possible, the current solution sol is then updated (line 10) and the LNS iterations are continued until the computing time limit is reached.

When p_1 useless iterations of the LNS operators have not led to any improvement of one of the best known solutions (one per level of time-consistency), diversification or restart components are called for (line 4). These two components and the selection process between them are presented in section

3.5. Iterations are counted as useless when they do not succeed in reducing the number of customers with a maximal number of time-classes.

3.3. Time-Window Definition for the VRPmTW-nw

The definition of the instance of the VRPmTW-nw to be solved at every iteration of the algorithm is based on the current solution sol, the day d that was deconstructed and the selected customer i. As explained in section 3.2, the objective of the deconstruction and reconstruction phases is to reduce the number of time-classes for i and to limit the number of time-classes for other customers according to a set of rules that are described in this section.

To describe our scheme, we introduce the following notation. The number of time-classes for $j \in V\backslash\{0\}$ in *sol* is denoted $\mathrm{TC}_j^{sol}(D)$. The number of time-classes for j after the deconstruction of day d is denoted $\mathrm{TC}_j^{sol}(D\backslash\{d\})$. $\mathrm{TC}^{sol}(D)$ is the maximal number of time-classes of the customers during horizon D, that is the time-consistency value for *sol*. Finally, the construction of the VRPmTW-nw instance relies on a parameter denoted $\mathrm{classMax}_j$. This represents the maximal number of time-classes allowed for $j \in V\backslash\{0\}$ in the new solution to be constructed.

According to this notation, $\mathrm{classMax}_j = \mathrm{TC}_j^{sol}(D) + 1$ means that any time of service is possible for customer j on day d (no time constraint). On the contrary, $\mathrm{classMax}_j = \mathrm{TC}_j^{sol}(D)$ generally means that the time of service for customer j on day d has to be compatible with the times of service on other days for this customer. The sole exception is met when $\mathrm{TC}_j^{sol}(D\backslash\{d\}) < \mathrm{TC}_j^{sol}(D)$, that is when the time of service of customer j on day d in *sol* was defining a time-class on its own.

In the above paragraph, compatible means that the time of service for customer j on day d can be integrated into one of the existing time-classes for this customer. Given a time-class whose lowest time of service is t_{inf} and highest time of service is t_{sup}, the condition for the time of service w_j^d to be compatible with this time-class is $t_{\mathrm{sup}} - \delta \leq w_j^d \leq t_{\mathrm{inf}} + \delta$. This condition can be expressed in the form of a time window condition: $w_j^d \in [t_{\mathrm{sup}} - \delta, t_{\mathrm{inf}} + \delta]$. Being compatible with the different time-classes then comes down to satisfying multiple time windows.

The number and the size of time windows (can be none) assigned to a customer in the travel cost minimization problem on day d depends on how the classMax parameter has been defined. Let us now describe the proposed definition of classMax. We recall that i is the customer selected at the current iteration of the LNS algorithm, for whom a decrease in the number of time-classes is expected.

1. Four cases are considered for customers $j \in V\backslash\{0\}, j \neq i$:
   (a) if $\mathrm{TC}_j^{sol}(D) = \mathrm{TC}^{sol}(D)$:
      i. if this condition held in the past p_2 iterations, $\mathrm{classMax}_j = \mathrm{TC}_j^{sol}(D) + 1$,
      ii. else, if $\mathrm{TC}_j^{sol}(D) = \mathrm{TC}^{sol}(D)$ for no more than p_2 iterations, $\mathrm{classMax}_j = \mathrm{TC}_j^{sol}(D)$,
   (b) if $\mathrm{TC}_j^{sol}(D) < \mathrm{TC}^{sol}(D)$:
      i. $\mathrm{classMax}_j = \mathrm{TC}_j^{sol}(D) + 1$ with probability $\min(p_3(\mathrm{TC}^{sol}(D) - \mathrm{TC}_j^{sol}(D)), 1)$,
      ii. otherwise $\mathrm{classMax}_j = \mathrm{TC}_j^{sol}(D)$.
2. For customer i, if $\mathrm{TC}^{sol}(D) > 1$, $\mathrm{classMax}_i = \mathrm{TC}^{sol}(D) - 1$, otherwise $\mathrm{classMax}_i = 2$.

Rules 1(a)i and 1(b)i state that the number of time-classes for customer j can be increased, whereas it remains the same in rules 1(a)ii and 1(b)ii. According to $\mathrm{classMax}_j$, time windows are then defined for j. Two situations are possible:

- $\mathrm{classMax}_j = \mathrm{TC}_j^{sol}(D\backslash\{d\}) \Rightarrow$ [multiple time windows],
- $\mathrm{classMax}_j > \mathrm{TC}_j^{sol}(D\backslash\{d\}) \Rightarrow$ [no time window].

The rationale behind using probability $\min(p_3(\mathrm{TC}^{sol}(D) - \mathrm{TC}_j^{sol}(D)), 1)$ in rules 1(b)i and 1(b)ii is that customers with a large number of time-classes are given less opportunity to increase this number. Parameter p_3 is set to 0 at the beginning of the algorithm and increased by 0.1 at every restart (see section 3.5) with a maximum value of 0.5.

The only possibility that the number of time-classes in the new solution exceeds $\mathrm{TC}^{sol}(D)$ comes from rules 1(a)i and 2 (second case), where the purpose is to inject some diversification.

Note finally that the definition of time windows may not actually reduce the number of time-classes of i in the first case of rule 2. In fact, when $\mathrm{TC}_i^{sol}(D\backslash\{d\}) = \mathrm{TC}_i^{sol}(D) = \mathrm{TC}^{sol}(D)$, no solution with $\mathrm{TC}^{sol}(D) - 1$ time-classes exists (remember that i is selected from the customers with a maximal number of time-classes on day d). In this case, $\mathrm{classMax}_i$ will not exactly represent the maximal number of time-classes for customer i. Definition $\mathrm{classMax}_i = \mathrm{TC}^{sol}(D) - 1$ will rather be handled by defining one time window per time-class of customer i except for the time-class in which the time of service of customer i on day d was ranked. The purpose is to decrease the size of this time-class and eventually to empty it in further iterations.

Example The preceding principles are illustrated in the following example: we consider a solution for 15 customers on 5 days with $\delta = 10$, which results in a time-consistency $\mathrm{TC}^{sol}(D) = 3$. The selected customer i is customer 7 and day 2 is deconstructed. Columns 2 to 6 of Table 1 present the time schedule of the solution for the 15 customers. Columns $\mathrm{TC}_j^{sol}(D)$ and $\mathrm{TC}_j^{sol}(D\backslash\{d\})$ present the number of time-classes of each customer in the current and the deconstructed solutions, respectively. The following columns then present the rule applied to each customer (with an increased number of time-classes allowed for customers 6 and 8) and the time windows defined according to these values ("–" stands for no time window).

3.4. Solution of the VRPmTW-nw

The problem to solve at each iteration of the reconstruction phase of the LNS consists in finding a set of routes with

TABLE 1. Definition of multiple time windows for a solution with 15 customers, 5 days and $\delta = 10$.

Customers	day 1	**day 2**	day 3	day 4	day 5	$TC_j^{sol}(D)$	$TC_j^{sol}(D\backslash\{d\})$	Rule	*classMax*	Time windows for day 2
1	54.6	54.6	91.8	91.1	91.1	2	2	1(b)ii	2	[44.6;64.6] [81.8;101.1]
2	12.7				29.4	2	2			
3	23	69			19.1	2	1	1(b)ii	2	–
4				55.2	55.2	1	1			
5	53.9			7.7		2	2			
6	71.8	15.6	23.8	21.8	21.8	2	2	1(b)i	3	–
i=7	**37.3**	**94.3**	**118.3**	**32.2**	**117.6**	**3**	**2**		**2**	**[27.3;42.2] [108.3;127.6]**
8	40.8	40.8	78	77.3	77.3	2	2	1(b)i	3	–
9	2.4	2.4	2.4	44.4	2.4	2	2	1(b)ii	2	[0;12.4] [34.4;54.4]
10	84.6		10.6	8.6	8.6	2	2			
11		84.8				1	0	1(b)ii	1	–
12		29.8	38		51.8	2	2	1(b)ii	2	[28;48] [41.8;61.8]
13	26.6	26.6	63.8	63.1	63.1	2	2	1(b)ii	2	[16.6;36.6] [53.8;73.1]
14		108.6		17.9		2	1	1(b)ii	2	–
15			8.1			1	1			

minimal cost that serve all customers involved on that day. One additional condition, compared to the standard VRP is that deliveries are restricted by multiple time windows. Also, no waiting time is allowed. We call this problem VRPmTW-nw and describe here the solution approach developed for it. Although the VRPmTW-nw is new (to the best of our knowledge), a heuristic branch-and-price solution scheme could relatively easily be adapted from existing VRPTW solution schemes.

The principle of the branch-and-price algorithm for the VRPTW [7, 9] is: a set-covering formulation is defined where path variables represent the set of feasible routes. Due to the very large number of possible routes, the formulation is first restricted to a limited subset of variables (restricted master problem) and then extended using a pricing subproblem. At each iteration of the so-called column generation scheme, the linear relaxation of the restricted master problem is solved and the pricing problem is called for, given a vector of optimal dual variables. The aim of the pricing problem is then to determine one or several feasible routes of negative reduced cost. By subtracting the values of dual variables from arc costs, the pricing problem can be expressed as an elementary shortest path problem with resource constraints and solved with dynamic programming. This whole scheme is embedded in a branch-and-price algorithm so as to obtain optimal integer solutions. Branching rules consist in imposing or forbidding arcs in the solution.

Regarding the VRPmTW-nw, most of the previous approaches can be applied in a similar way. The only difference lies in the pricing problem and concerns the feasibility of routes. While extending paths in the dynamic programming algorithm, one has to address multiple time windows and forbidden waiting times. Feasibility is assessed by simply checking that the destination node is reached within one of its time windows. As waiting times are not allowed, the condition that two paths arrive at the destination at exactly the same time is added to the dominance rule. In fact, earlier arrival, which is usually preferred as it gives more opportunity for further extensions, could here prevent some customers from being reached. The consequence is a significantly weakened dominance.

In the process of maintaining elementary paths and reinforcing the dominance rule, label definition includes the computation of the set of reachable nodes [10]. Here, reachable nodes are those that were not visited in the partial path represented by the label and whose last time window ends after the earliest possible arrival.

Actually, in view of the way multiple time windows are defined (see section 3.3), it is possible that a large proportion of customers are not subject to time windows. So as to counterbalance the negative effect of the no-waiting-time condition on the dominance rule, we implement the following mechanism. Although computing the set of reachable nodes for a label, we determine that we do not fall into the situation where all the remaining nodes have no time windows. For these labels, the traditional dominance rule is applied.

Through these adaptations, the VRPmTW-nw can theoretically be solved exactly. As the branch-and-price algorithm has to be called many times using relatively similar data, a single pool of columns is kept and enriched throughout the successive calls of the algorithm. A filtering algorithm is performed at each new call to consider only feasible columns (regarding the current data) in the model. However, the need to solve a VRPmTW-nw instance at every iteration of Algorithm 2 also implies very limited computing times. Hence, a heuristic variant of the previous branch-and-price scheme is developed.

When solving the pricing problem with dynamic programming, the set of labels conserved with every node of the graph is limited to the $p_4 = 20$ best nondominated labels, in order of increasing time. Exploration in the dynamic programming algorithm is also limited with a limited discrepancy search policy [11]. Basically, extensions are limited to the $p_5 = 2$ nearest neighbors (sorted according to the reduced cost). However, extensions to other neighbors are permitted with the limit of at most $p_6 = 2$ extensions for a complete route.

It is well-known that column generation is subject to slow convergence. To avoid long series of column generation iterations without a clear improvement in the objective value, we stop after a given number of iterations with improvements less than $p_7 = 1\%$. This number of iterations is set to $p_8 = 5$ when a feasible solution has already been found, $p_9 = 10$ otherwise.

Finally, a simple diving strategy is implemented for the branching phase. When the current optimal solution of the relaxed restricted master problem is fractional, instead of splitting the search into two descendant nodes, we only explore the descendant node where an arc is imposed.

These three heuristic mechanisms avoid, respectively, long pricing problem solution times, a large number of iterations of column generation at each node of the search tree, and a large number of nodes in the search tree, thus ensuring that many instances of the VRPmTW-nw can be solved in a reasonable time. A drawback of these mechanisms is that they involve a relatively large set of parameters (p_4 to p_9). The value of the different parameters defined above was set through preliminary experiments, which demonstrated, however, that the method was not very sensitive to these values.

3.5. *Restart and Diversification*

The diversification operator works as follows. The first step is to select a customer with a maximal number of time-classes for a maximal number of iterations. This customer is then removed from the solution and reinserted every requested day as the first customer of a route; more precisely, the route that yields the minimal insertion cost among the existing routes on each day. Since vehicles always start from the depot at time 0, the time of service for this customer will always be the same, leading to only one time-class for this customer. This simple operator thus enforces consistency for a customer who was stuck with a large number of time-classes for a long time and starts the search in a completely new direction.

The restart operator replaces diversification when a solution with a time-consistency equal to 1 has been found since the last diversification or restart. It consists in restoring one of the best known solutions as the current solution. Our choice is to restore solutions with any potential number of time-classes. Thus, the first call to the restoration operator will generate a solution with $|D|$ time-classes, then $|D| - 1$, and so forth. When value 1 is reached, the process starts again from $|D|$. As the LNS algorithm tends to search for solutions with a small time-consistency, this restart operator aims to spend some time improving best known solutions with larger time-consistencies.

4. COMPUTATIONAL EXPERIMENTS

The LNS heuristic was coded in C++ and run on a machine with a 2.5-GHz processor and 3 GB of RAM and evaluated on two types of data instances. First, we considered the so-called small instances of [14], which comprise 10 instances with 10 or 12 customers over a 3-day horizon. The average level of presence of the customers is around 70%. The instances are named *conVRPx-y*, with $x = 10$ or 12 for the number of customers and $y = 1, \ldots, 5$, for the instance index. In these instances, a route duration limit is included, which can easily be handled in the model and the algorithm.

Then, we considered 14 real cases arising from day care centers for mentally disabled children or adults in the area of Nantes (France). The centers have to transport between 15 and 65 people everyday (inbound trips in the morning and outbound trips in the evening). The planning horizon is 5 days. For each original instance, the demand of the transported persons is represented by vectors of five binary values: 1 if the person has to be transported on that day and 0 otherwise. We modified the original demand vector so that the level of presence varied from $\rho = 0.5$ to $\rho = 0.9$ in steps of 0.1. To obtain a level of presence ρ with realistic data, we made the following assumptions: a proportion ρ^2 of the people make a demand everyday; a proportion $(1 - \rho^2)$ of the people have a level of presence of $\frac{\rho}{1+\rho}$; for the latter category, the level of presence on Tuesdays, Wednesdays, and Thursdays is twice that of Mondays and Fridays. The resulting 70 instances are named data *z-x*, where $z = 10 \times \rho$ and x is the number of customers.

The original instances [19] include additional constraints such as individual time windows or maximum trip duration, several possible addresses during the week, incompatibilities between people, wheelchair constraints, a heterogeneous fleet, and so forth. We ignored these complicating constraints to focus on the consistency aspect.

As explained in the introductory section of the article, a post processing phase is carried out to assign drivers to routes in such a way that driver-consistency is optimized. The objective is to minimize the average number of drivers per customer. This is introduced and discussed in [24], where the authors explain that it is representative of customer satisfaction in terms of driver-consistency. They also propose several integer programming formulations. In our computations, we used their (DC) model which minimizes the average number of drivers transporting each customer.

4.1. *Small Instances*

The aim of this subsection is threefold: (i) to show that the problem is difficult to solve in practice (i.e., the limits of the integer linear model); (ii) to validate the proposed heuristic in comparison with the best solutions found; and (iii) to compare the proposed model with Groër et al. [14].

4.1.1. Comparison with the Integer Linear Program

Table 2 first evaluates the ability of the integer linear programming formulation introduced in section 3.1 to solve the problem. As it seems almost incapable of solving any of the instances, we temporarily relax vehicle capacity constraints, maximum route length constraints, and assume unitary demands. These instances, derived from the original

TABLE 2. Optimal (ILP) and heuristic (LNS) solutions on modified small instances ($\delta = 5$).

	ILP			Avg LNS		
Time-classes:	3	2	1	3	2	1
RconVRP10-1	92.91	92.91	92.91	92.91	92.91	92.91
RconVRP10-2	80.42	80.42	–	80.42	80.42	80.99
RconVRP10-3	94.12	94.12	94.37	94.12	94.12	94.37
RconVRP10-4	93.71	93.71	–	93.71	93.71	94.09
RconVRP10-5	83.84	83.84	–	83.84	84.54	96.08
RconVRP12-1	103.65	103.65	–	103.65	103.65	104.40
RconVRP12-2	73.89	73.89	–	73.89	73.89	81.27
RconVRP12-3	82.77	82.77	–	82.77	82.77	83.12
RconVRP12-4	97.57	97.57	–	97.57	97.57	101.91
RconVRP12-5	83.63	83.63	–	83.63	83.63	89.25

ones, are named "RconVRPx-y." The sensitivity parameter δ is set to 5. Three runs of 10 h using ILOG-CPLEX 12.0 are performed on each instance, setting the maximal number of time-classes C – in inequality (21) – to 3, 2 and 1, respectively. The optimal solutions found are presented in the "ILP" section of the table. Optimal solutions are compared to the solutions provided by our heuristic in 5 min of computing time (each run provides a solution for every number of time-classes): the "Avg LNS" part presents the average results over 10 runs of the heuristic.

The ILP formulation was, in most cases, unable to solve the 1-time-class problems to optimality within 10 h. Moreover, ILOG-CPLEX was not able to improve the lower bounds given by the 2-time-classes problems. All the known optimal solutions were reached at each run of the LNS algorithm, except for instance RconVRP10-5; for this one, our algorithm was not able to find the optimal solution with 2 time-classes; instead solutions of costs 84.5 and 84.69 were found, 8 and 2 times, respectively, for an average value of 84.54.

To evaluate the results obtained with 1 time-class, Table 3 provides the average and best solutions found by LNS (after the 10 runs), the best feasible solution built by the ILP, and the best known solution among these solutions ("Best"). Values that match the best known solutions are written in bold. Our heuristic failed to find the best known solution for only two

TABLE 3. 1-time-class solutions on modified small instances ($\delta = 5$).

	Avg LNS	Best LNS	ILP	Best
RconVRP10-1	**92.91**	**92.91**	**92.91**	92.91
RconVRP10-2	**80.99**	**80.99**	82.83	80.99
RconVRP10-3	**94.37**	**94.37**	**94.37**	94.37
RconVRP10-4	**94.09**	**94.09**	94.53	94.09
RconVRP10-5	96.08	**96.01**	**96.01**	96.01
RconVRP12-1	**104.40**	**104.40**	106.87	104.40
RconVRP12-2	81.27	**81.25**	**81.25**	81.25
RconVRP12-3	**83.12**	**83.12**	**83.12**	83.12
RconVRP12-4	**101.91**	**101.91**	**101.91**	101.91
RconVRP12-5	**89.25**	**89.25**	**89.25**	89.25

TABLE 4. Heuristic solutions and driver-consistency on small instances ($\delta = 5$).

				Number of customers with		
Instance	Opt. cost	LNS cost	Gap	1 driver	2 drivers	3 drivers
conVRP10-1	122.03	118.70	2.8%	7	3	0
conVRP10-2	99.07	96.57	2.6%	8	2	0
conVRP10-3	123.41	123.16	0.2%	9	1	0
conVRP10-4	126.89	126.89	0.0%	10	0	0
conVRP10-5	109.31	104.04	5.1%	7	3	0
conVRP12-1	144.02	143.23	0.6%	10	2	0
conVRP12-2	89.54	83.27	7.5%	9	3	0
conVRP12-3	119.69	113.74	5.2%	9	3	0
conVRP12-4	141.37	137.26	3.0%	10	2	0
conVRP12-5	117.42	110.17	6.6%	9	3	0

cases. For the RconVRP10-5 instance, it found a suboptimal solution of cost 96.7 once out of 10 runs. For the RconVRP12-2 instance, our heuristic returned a 1-time-class solution with cost 81.34 instead of 81.25 twice out of 10 runs.

For three instances out of 10, our heuristic found better results than the ILP formulation. Moreover, since the gap between the best-LNS solutions and the lower bounds—taken from the 2-time-classes optimal solutions of Table 2—is 3.8% on average and 14.5% at maximum, we can be confident in the quality of the solutions computed by our heuristic on the modified small instances.

4.1.2. Comparison with the Approach of Groër et al. Table 4 presents a comparison between our approach and that proposed by Groër et al. [14]. For this set of experiments, the original instances *conVRPx-y* introduced in [14] are considered.

The purpose here is not to compare the solution methods but rather to discuss the modeling differences. Indeed, Groër et al. enforce in their model that a single driver is used for every customer, while we do not consider this issue implicitly but rather optimize the number of drivers *a posteriori*. Also, Groër et al. do not introduce the concept of time-classes but they consider the maximal difference between the service times. Their heuristic minimizes (implicitly) that difference while it is bounded in their ILP formulation. The chosen limit is 5. This situation corresponds to the set of 1-time-class solutions provided with our heuristic, with parameter δ set to 5.

The optimal costs obtained in [14] are reported in the column "Opt." of Table 4.[1] The costs of the 1-time-class solutions obtained for these instances with our heuristic are reported in the next column, followed by the observed gap with "Opt." The difference in cost is not significant on average (3.4%), showing improvements of up to 7.5% when relaxing driver consistency. The impact on driver-consistency can be evaluated by the last three columns of the table, where we report the number of customers with

[1] Optimal values given in [14] include service durations. They are discarded in the costs of Table 4.

TABLE 5. Solutions of the heuristic method on real instances ($\delta = 10$).

C_{max}	5	4	3	2	1
data5–15	700.8	664.7	663.2	670.5	705.7
data5–21	827.1	775.8	775.9	776.9	823.7
data5–25	724.5	617.8	617.8	618	655.9
data5–26	845.2	777.3	771.1	778.5	829.2
data5–27	1010	946.5	940.8	947.2	1066.8
data5–32		1002.1	987.8	988.4	1023.2
data5–41	1465.5	1439.5	1467.8	1467.1	1615.1
data5–44	1176.5	1158.6	1150.4	1162	1219.2
data5–46	1482	1474.3	1474.2	1497.3	1562.1
data5–48		1460.1	1457.7	1462	1590.6
data5–55		1622.7	1605.2	1606.8	1855.6
data5–59	2804.4	2787.8	2778.6	2789.9	3059.8
data5–64	2141.1	2140.1	2172.3	2163	2321.2
data5–65	1795.5	1797.4	1798.1	1802.3	2052.7
data6–15	736.4	689.7	688.3	691.4	731.3
data6–21	849.3	794.7	793.6	795.1	812.9
data6–25	788.9	695.4	683.1	682.8	736.1
data6–26	853.3	843.8	839.4	841.1	888.1
data6–27	1040.6	951.1	952.9	954.9	1030.9
data6–32	1004.2	1002	992.4	992.4	1013.5
data6–41	1883	1521.2	1540.2	1565.8	1841
data6–44	1351.5	1251.3	1246.6	1251.1	1340.9
data6–46	1618.7	1506.1	1503.1	1515.7	1608.1
data6–48	1529.7	1533.2	1536	1529.6	1663.7
data6–55	1895	1890.1	1858.8	1861.4	2011.1
data6–59	3026.4	3023	3028.4	3017.7	3337.3
data6–64	2338	2327.8	2325.4	2333.7	2558.1
data6–65	2038.4	2018.4	2003.3	2012.8	2104.7
data7–15	766	749.6	746.9	751.3	773.6
data7–21	846.5	832.3	831.6	833.6	880.7
data7–25	776.3	726.4	724.5	727.4	754.9
data7–26	900.5	885.4	882.6	890.2	918.9
data7–27	1122.8	1059.4	1057	1059.9	1108.6
data7–32	1249.5	1103	1094.6	1101.8	1146.8
data7–41	1746.5	1724.5	1669.4	1664.1	1800.1
data7–44		1306.2	1301.4	1301.8	1369.2
data7–46	1697.8	1679	1659.5	1664.1	1717.1
data7–48	1729	1723.3	1713.7	1716.1	1854
data7–55	2003.8	1945.8	1931.8	1933.8	2038.3
data7–59	3388.7	3364.2	3340.2	3348.1	3667.7
data7–64		2608.6	2630.7	2612	2817.7
data7–65		2235.4	2261.6	2280.4	2380.2

one, two, and three drivers, obtained after optimally assigning drivers to the routes in the heuristic solutions. Here, perfect driver-consistency is only reached for instance "conVRP10-4." However, the proportion of customers with two drivers never exceeds 30% and no customer has three drivers on these instances. The average losses in driver-consistency (counted in percentage of customers served by more than one driver) equal to 1.8 and 2.6% on instances with 10 and 12 customers, respectively, enable average cost savings of 2.1 and 4.6%. Hence, given the cost reduction obtained, allowing some flexibility in driver-consistency can be worthwhile when possible.

4.2. Real Instances

In this subsection, we investigate the efficiency of our solution approach on the 70 real instances. Remember that in these instances no route duration limit is considered. We first present a set of results (benchmark results) obtained with a computing time of 1 h per instance. The objective here is to give some reference results for further research. Reference results are also provided to evaluate the efficiency of our heuristic with more limited computing times and assess the impact of time-consistency on routing costs.

4.2.1. Benchmark Results Tables 5 and 6 detail the results obtained by a single run of our heuristic, limited to 1 h, on real instances with a time-consistency parameter δ set to 10.

Note that the values given in the first column of this table are the best solutions found with exactly the number of time-classes indicated in the label of the column. In practice, it is clear that a result with fewer time-classes and smaller routing costs would be preferred. For example, on instance data5–15, solutions with 5 and 4 time-classes are actually dominated by the solution with 3 time-classes. These kinds of solutions are generally generated because of diversification and are not really optimized. We decided to present the results as such considering that "non-dominated" solutions can easily be deduced. Statistics on these results and also on "non-dominated" solutions are proposed in Table 7. The first part of the table (Rough results) summarizes the results given in Table 5 and provides the average value for the instances for which a solution was found. In the second part of the table

TABLE 6. Solutions of the heuristic method on real instances ($\delta = 10$) continued.

C_{max}	5	4	3	2	1
data8–15	817.5	776.5	775.3	777.2	792.8
data8–21	910.4	899.2	899.2	902.9	916.9
data8–25		860.7	853.9	853.3	857
data8–26	1110.6	974.6	965	968.5	1000.8
data8–27	1341.1	1198.8	1186.2	1186.2	1247.3
data8–32	1172.2	1159.8	1152.4	1154.6	1175.4
data8–41		1917.5	1906.9	1910.5	1989.7
data8–44		1433.5	1410.8	1412.9	1462.7
data8–46	1806.2	1799	1785	1782.7	1856.7
data8–48	1862.1	1843.8	1848.4	1848	1942.4
data8–55	2122.5	2085.1	2054.5	2065.3	2197.2
data8–59	3657.5	3656.7	3640.7	3645.5	3848
data8–64	2823	2792.8	2774.4	2809.6	2914.9
data8–65	2498.8	2504.2	2469	2461.6	2570.3
data9–15	861.5	796.1	797.4	799.1	816.1
data9–21	1044.8	1003.6	998.8	999.9	1008.9
data9–25		917	895.4	894.6	908.1
data9–26	1066.3	1025.4	1024.6	1024.6	1025.6
data9–27	1327.8	1221.9	1211.7	1213.7	1257.7
data9–32	1380.9	1210.5	1189.6	1193.1	1202.8
data9–41		2065.3	2052.4	2038.9	2040.4
data9–44	1622.6	1617.7	1562.4	1571.3	1671.2
data9–46		1855.2	1844.5	1857.4	1899.9
data9–48	2089.7	1993.2	1982.2	1982.2	2064.2
data9–55		2183.6	2184.5	2197	2484.2
data9–59	4033.1	4021	3936.9	3932.9	4084
data9–64	3030.3	3023.1	3004.4	2999.4	3037.5
data9–65		2656.7	2625.9	2626.8	2686.3

TABLE 7. Statistics on real instances

	Number of time-classes				
	5	4	3	2	1
Rough results					
Number of solutions	56	70	70	70	70
Average value	1584.51	1573.54	1564.72	1568.14	1660.65
Non-dominated solutions					
Number of solutions	70	70	70	70	70
Average value	1561.93	1561.95	1563.50	1568.14	1660.65
Average cost of consistency	–	0.00%	0.10%	0.30%	5.90%

(Non-dominated solutions), values of "dominated" solutions are replaced by values of solutions that "dominate" them (also, the best solution with fewer time-classes is considered when no solution was found). In the last line of the table, gaps indicate the increase in travel costs observed compared to solutions where 5 time-classes are allowed ($C = 5$).

Several comments can be drawn from Tables 5–7. The first rather surprising result is that feasible solutions with a small number of time-classes could be found for all 70 instances. Only some solutions with exactly 5 time-classes were not obtained on the whole set of instances. This latter point is not really surprising since having 5 different classes can sometimes counter-optimize the solution. For example, it is possible that the solution obtained by solving the VRP each day independently (which is optimal in terms of routing) does not introduce any customer with 5 time-classes. In this case, solutions with 5 time-classes are naturally dominated and not explored during the search.

In addition, we remark that imposing time-consistency does not penalize routing costs so much. As long as 2 time-classes are accepted, the impact on routing costs is less than 1%. When a limit of 1 time-class per customer is strictly enforced, routing costs are only increased by 5.90%, which is not negligible but might be offset by the increase in consistency for the customers.

Regarding the solution method, it is interesting to add that an average of 6,000 iterations could be performed in 1 h, thus allowing a large exploration of the solution space (remember that in one iteration, one fifth (1 day) of the solution can be completely redrawn).

4.2.2. Efficiency of the Algorithm with Shorter Computing Times In Tables 8 and 9, we compare the number of solutions found and the values of the solutions, for computing times varying between 60 s, 600 s, and 1 h, respectively. Table 8 is based on rough results and Table 9 considers non-dominated solutions. Note that when computing a gap, only instances for which a solution was found for both computing time limits are considered.

TABLE 8. Number of solutions found.

	Number of time-classes				
Computing time	5	4	3	2	1
60 s	29	68	67	61	44
600 s	43	70	70	70	65
1 h	56	70	70	70	70

TABLE 9. Gap compared to (non-dominated) solutions found in one hour of computing time.

	Number of time-classes				
Computing time	5	4	3	2	1
60 s	1.38%	1.45%	2.22 %	3.30 %	2.25 %
600 s	0.56%	0.58%	0.86%	1.19 %	0.56%

One can see from Tables 8 and 9 that "good" solutions can be obtained relatively quickly using the heuristic (less than 10 min). After these few minutes, the results are only marginally improved. The only exception concerns solutions with 1 time-class, which are the most difficult to compute and sometimes require many iterations to obtain. The shortest computing times are not assessed as, due to the complexity of the problem to be solved at each iteration of the algorithm, the method is not designed to be efficient in a few seconds.

4.2.3. Impact of Parameter δ Tables 10 and 11 aim to show how solutions are impacted when a tighter gap is fixed. We compare the number of solutions found and the values of the solutions obtained when defining $\delta = 5$ with the previous results ($\delta = 10$). Table 10 is based on rough results, Table 11 considers non-dominated solutions. When computing a gap, only instances for which a solution was found for both values δ are considered.

The main observation that can be drawn from Table 10 is that solutions with one time-class can still be found most of the time (63 times out of 70). However, we cannot know whether the missing solutions could not be obtained because of our algorithm or they do not exist. Table 11 highlights that when more than 1 time-class is allowed, decreasing δ only slightly impacts the costs. Regarding solutions with 1 time-class, one can observe a clear increase in the costs.

4.2.4. Driver-Consistency Table 12 indicates the average number of drivers assigned to customers. Let us recall that

TABLE 10. Number of solutions found.

Number of time-classes	5	4	3	2	1
$\delta = 10$	56	70	70	70	70
$\delta = 5$	68	70	70	70	63

TABLE 11. Gap compared to (non-dominated) solutions found with $\delta = 10$.

Number of time-classes	5	4	3	2	1
$\delta = 5$	0.57%	0.66%	0.69%	1.29%	7.09%

TABLE 12. Average number of drivers per customer.

Number of time-classes	5	4	3	2	1
Average number of drivers	1.43	1.40	1.35	1.30	1.24

driver-consistency is not explicitly addressed in our model and approach, but is rather handled in post processing so that the assignment of drivers to the routes computed with the heuristic optimizes the driver-consistency. Table 13 gives the percentage of customers having a given number of drivers out of all the customers of all the solutions (from 5 time-classes to 1) of the 70 instances. Note that these results are compiled from the solutions presented in Table 5, that is, the set of rough results.

The average number of drivers remains quite limited though not explicitly handled in the algorithm. In addition, we remark that the majority of customers have one or two drivers.

5. CONCLUSIONS

The content of this article was initially motivated by the optimization of vehicle routes when transporting people with disabilities to day care centers. So as to capture the specific consistency expectations in this context, a new time-consistency measure, based on time-classes, was defined. From this measure, we introduced the TCVRP and proposed a solution method based on large neighborhood search.

Experiments tend to demonstrate the great difficulty of this new problem and the efficiency of the heuristic proposed. An important result is that, at least for the (realistic) instances used for the computations, very consistent solutions could always be found with a relatively small impact on travel costs.

Though driver-consistency was not addressed in the model nor in the solution approach, a fairly good consistency could also be maintained for drivers.

A clear development of this work is to provide efficient exact algorithms to solve the TCVRP. The integer programming formulation proposed in this article is not solved efficiently by the solver so it would be worth investigating more evolved modeling, valid inequalities and branch-and-cut approaches. From the application perspective, introducing the diverse constraints that were not considered here would also be important. In addition, other possible models of time-consistency could be investigated: for example, it might be interesting to consider the number of time-class changes over successive periods, instead of just counting the number of classes.

TABLE 13. Percentage of customers having a given number of drivers.

Number of drivers	1	2	3	4	5
Percentage of customers	69.06%	25.9%	4.56%	0.46%	0.00%

Finally, we introduced another new routing problem in the paper, the so-called VRPmTW-nw. Given the inherent interest and difficulty of this problem, further research regarding its solution certainly deserves to be conducted.

REFERENCES

[1] F. Baniel, Prise en compte d'objectifs de stabilité pour l'optimisation de collecte de déchets, Ph.d thesis, Université de Toulouse, France, 2009.

[2] W.C. Benton and M.D. Rossetti, The vehicle scheduling problem with intermittent customer demands, Comput Oper Res 19 (1992), 521–531.

[3] D.J. Bertsimas, A vehicle routing problem with stochastic demand, Oper Res 40 (1992), 574–585.

[4] D. Bredström and M. Rönnqvist, Combined vehicle routing and scheduling with temporal precedence and synchronization constraints, Eur J Oper Res 191 (2008), 19–31.

[5] T. Choi and J.L. Hartley, An exploration of supplier selection practices across the supply chain, J Oper Manag 14 (1996), 333–343.

[6] J.M. Day, P.D. Wright, T. Schoenherr, M. Venkataramanan, and K. Gaudette, Improving routing and scheduling decisions at a distributor of industrial gases, Omega 37 (2009), 227–237.

[7] G. Desaulniers, J. Desrosiers, and M.M. Solomon (Editors), Column generation, GERAD 25th anniversary series. Springer, US, 2005.

[8] L.-M. Roussseau, E. Prescott-Gagnon, and G. Desaulniers, A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows, Networks 54 (2009), 190–204.

[9] D. Feillet, A tutorial on column generation and branch-and-price for vehicle routing problems, 4OR–Q J Oper Res 8 (2010), 407–424.

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

[11] D. Feillet, M. Gendreau, and L.-M. Rousseau, New refinements for the solution of vehicle routing problems with branch and price, INFOR 45 (2007), 239–256.

[12] P.M. Francis, K.R. Smilowitz, and M. Tzur, "The period vehicle routing problem and its extensions," The vehicle routing problem: Latest advances and new challenges, Vol. 43 of Operations Research/Computer Science Interfaces Series, B. Golden, S. Raghavan, E. Wasil, R. Sharda, and S. Voß, (Editors), Springer, US, 2008, pp. 73–102.

[13] M. Gendreau, G. Laporte, and R. Seguin, A tabu search heuristic for the vehicle routing problem with stochastic demands and customers, Oper Res 44 (1996), 469–477.

[14] C. Groër, B. Golden, and E. Wasil, The consistent vehicle routing problem, Manuf Serv Oper Manag 11 (2009), 630–643.

[15] M.A. Haughton, Measuring and managing the learning requirements of route reoptimization on delivery vehicle drivers, J Bus Logist 23 (2001), 45–66.

[16] M.A. Haughton, Assigning delivery routes to drivers under variable customer demands, Transp Res Part E Logist Transp Rev 43 (2007), 157–172.

[17] I. Ioachim, J. Desrosiers, F. Soumis, and N. Bélanger, Fleet assignment and routing with schedule synchronization constraints, Eur J Oper Res 119 (1999), 75–90.

[18] A.A. Kovacs, S.N. Parragh, and R.F. Hartl, A template based adaptive large neighborhood search for the consistent vehicle routing problem. Networks 63 (2014), 60–81.

[19] F. Lehuédé, C. Pavageau, and O. Péton, "Un système d'aide à la décision pour planifier les transports vers les établissements médico-sociaux," Proc Handicap 08 Conf, Paris, June 2008.

[20] S. Ropke and D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, Transp Sci 40 (2006), 455–472.

[21] L.-M. Rousseau, M. Gendreau, and G. Pesant, The synchronized vehicle dispatching problem, Technical Report CRT-2003-11, Centre de Recherche sur les Transports, Universitée de Montréal, Canada, 2003.

[22] H. Schneider, A. Reinholz, and H.-W, Graf. Integrated modeling and optimization of the consistent vehicle routing problem with the super node concept, Proc 9th Metaheuristics Int Conf (MIC 2011), Udine, Italy, July 25–28, 2011.

[23] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," Principles and practice of constraint programming – CP98, Vol. 1520 of Lecture notes in Computer Science, M. Maher and J.-F. Puget (Editors), Springer, Berlin/Heidelberg, 1998, pp. 417–431.

[24] K. Smilowitz, M. Nowak, and T. Jiang, Workforce management in periodic delivery operations, Transp Sci 47 (2013), 214–230.

[25] I. Steinzen, L. Suhl, and N. Kliewer, "Branching strategies to improve regularity of crew schedules in ex-urban public transit," ATMOS 2007—7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, C. Liebchen, R.K. Ahuja, and J.A. Mesa, (Editors), Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[26] I. Sungur, Y. Ren, F. Ordóñez, M. Dessouky, and H. Zhong, A model and algorithm for the courier delivery problem with uncertainty, Transp Sci 44 (2010), 193–205.

[27] C. Tarantilis, F. Stavropoulou, and P.P. Repoussis, Template-based tabu search algorithm for the consistent vehicle routing problem, Expert Syst Appl 39 (2012), 4233–4239.

[28] H. Zhong, R.W. Hall, and M. Dessouky, Territory planning and vehicle dispatching with driver learning, Transp Sci 41 (2007), 74–89.

European Journal of Operational Research 254 (2016) 80–91

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization

Philippe Grangier[a,b], Michel Gendreau[b], Fabien Lehuédé[a,*], Louis-Martin Rousseau[b]

[a] L'UNAM, Ecole des Mines de Nantes, IRCCyN UMR CNRS 6597, 4 Rue Alfred Kastler, 44307 Nantes Cedex 3, France
[b] Department of Mathematics and Industrial Engineering and CIRRELT, Ecole Polytechnique de Montréal and CIRRELT, C.P 6079, Succursale Centre-ville, Montreal, QC H3C 3A7, Canada

ARTICLE INFO

Article history:
Received 2 July 2014
Accepted 22 March 2016
Available online 30 March 2016

Keywords:
Routing
Two-echelon VRP
Synchronization
City logistics
Adaptive large neighborhood search

ABSTRACT

The two-echelon vehicle routing problem (2E-VRP) consists in making deliveries to a set of customers using two distinct fleets of vehicles. First-level vehicles pick up requests at a distribution center and bring them to intermediate sites. At these locations, the requests are transferred to second-level vehicles, which deliver them. This paper addresses a variant of the 2E-VRP that integrates constraints arising in city logistics such as time window constraints, synchronization constraints, and multiple trips at the second level. The corresponding problem is called the two-echelon multiple-trip vehicle routing problem with satellite synchronization (2E-MTVRP-SS). We propose an adaptive large neighborhood search to solve this problem. Custom destruction and repair heuristics and an efficient feasibility check for moves have been designed and evaluated on modified benchmarks for the VRP with time windows.

1. Introduction

The two-echelon vehicle routing problem (2E-VRP) consists in routing freight from a central depot to customers through a set of intermediate sites. The depot is an intermodal logistics site called the distribution center (DC). It has some storage capacity, and it is where consolidation takes place. Intermediate sites, usually called satellites, have little or no storage capacity but are located closer to customers. Two fleets of vehicles are involved: first-level vehicles carry requests from the DC to the satellites, and second-level vehicles carry requests from the satellites to the customers. First-level vehicles are usually significantly larger than second-level vehicles.

Over the last few years freight transportation in urban areas has received much attention (Cattaruzza, Absi, Feillet, & González-Feliu, 2015). Indeed, because of increasing traffic congestion, environmental issues, and low average truckloads, new policies (e.g., London Congestion Charges, Monaco UDC) and initiatives (Amsterdam City Cargo) have emerged to ban large trucks from city centers. This movement is known as *city logistics* and represents a move from independent direct shipping strategies toward integrated logistics systems. In this context, multi-echelon distribution systems and particularly two-tiered systems are often proposed as an alternative to current distribution systems (Crainic, Ricciardi, & Storchi, 2009).

Several specific constraints arise in the urban context: time windows, multiple use of vehicles, and synchronization. Delivery hours are often restricted because of customer requirements or city regulations. Moreover, second-level vehicles are usually small in order to access every street, so even a full load does not represent an entire work-day. Finally, operating a satellite in a city is expensive, because of labor costs and high rents. More and more cities allow transporters to use dedicated or existing infrastructures (reserved parking spaces, bus depots) to unload (Crainic, Ricciardi, & Storchi, 2004). No storage capacity is normally available at these locations, thus requiring a synchronization of the two levels.

The contribution of this paper is a solution methodology for a 2E-VRP that integrates constraints that have not yet been addressed in the literature: time windows, synchronization, and multiple trips. Similar problems have been discussed in Perboli, Tadei, and Vigo (2011) under the name *two-echelon vehicle routing problem with satellite synchronization* (2E-VRP-SS) and modeled in Crainic et al. (2009) under the name *two-echelon, synchronized, scheduled, multidepot, multiple-tour, heterogeneous VRPTW* (2SS-MDMT-VRPTW). However, to the best of our knowledge, no implementation has been reported.

Related work includes models for city logistics, multi-echelon vehicle routing problems with multiple routes and transfer or synchronization constraints. A general model for city logistics

* Corresponding author. Tel.: +33 2 51 85 83 21.
E-mail addresses: philippe.grangier@mines-nantes.fr (P. Grangier), michel.gendreau@cirrelt.ca (M. Gendreau), fabien.lehuede@mines-nantes.fr, fabien.lehuede@emn.fr (F. Lehuédé), louis-martin.rousseau@polymtl.ca (L.-M. Rousseau).

http://dx.doi.org/10.1016/j.ejor.2016.03.040
0377-2217/

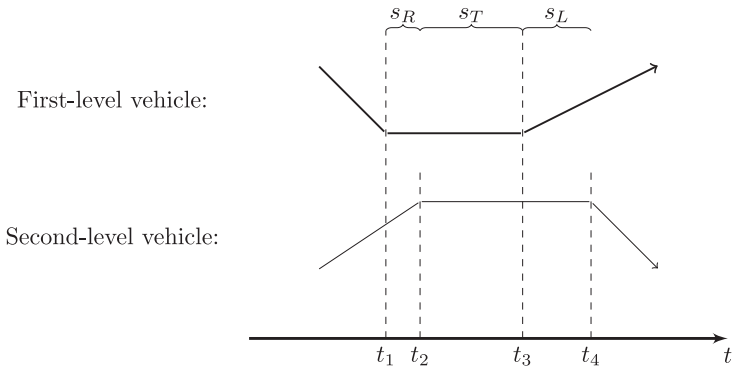


Fig. 1. Time chart for a transfer.

systems is presented by Crainic et al. (2009), while Mancini focuses on multi-echelon systems (Mancini, 2013). The 2E-VRP was introduced by Perboli and Tadei (2010), who proposed a mathematical model. Since then several algorithms have been developed: math-based heuristics (Perboli & Tadei, 2010; Perboli et al., 2011), clustering-based heuristics (Crainic, Mancini, Perboli, & Tadei, 2008), GRASP (Crainic, Mancini, Perboli, & Tadei, 2011; 2013; Zeng, Xu, Xu, & Shao, 2014), adaptive large neighborhood search (ALNS) (Hemmelmayr, Cordeau, & Crainic, 2012), and a large neighborhood search combined with a local search (Breunig, Schmid, Hartl, & Vidal, 2015). Exact methods include (Jepsen, Spoorendonk, & Ropke, 2013; Roberti, 2012; Santos, Cunha, & Mateus, 2012; Santos, Mateus, & da Cunha, 2014). Crainic, Perboli, Mancini, and Tadei (2010) study the impact of satellite location on the cost of a 2E-VRP solution compared to that of a VRP. Cuda, Guastaroba, and Speranza (2014) recently published a survey on two-echelon routing problems. A similar problem is the two-echelon location routing problem (2E-LRP) (Nguyen, Prins, & Prodhon, 2012). Our problem also integrates some multiple-trip aspects (Taillard, Laporte, & Gendreau, 1996) that have been solved with tabu search (Nguyen, Crainic, & Toulouse, 2013), ALNS (Azi, Gendreau, & Potvin, 2014) and iterated local search (Cattaruzza, Absi, Feillet, & Vigo, 2014). Synchronization of multiple trips supplied by a single bus line as been studied in Masson et al. (2015) in a city logistics context. We refer to (Drexl, 2012) for a detailed survey of synchronization in vehicle routing problems and to (Cattaruzza et al., 2015) for a recent survey of vehicle routing problems in city logistics.

The 2E-MTVRP-SS can be considered as a particular Pickup and Delivery Problem with Transfers (PDPT) which has been recently studied in (Masson, Lehuédé, & Péton, 2013a; 2013b; Qu & Bard, 2012). The major differences are that in the 2E-MTVRP-SS transfers are mandatory, routes should be designed for two types of vehicles which do not share the same network, and that two vehicles must be simultaneously present at a satellite during a transfer. In this paper, we extend the previous approaches to integrate those differences and we exploit the specificities of the 2E-MTVRP-SS to propose a better exploration of the search space, as well as a more compact graph representation of temporal constraints.

The remainder of this paper is organized as follows. Section 2 presents a formulation of the problem, and Sections 3 and 4 are devoted to the solution method with a special focus on efficiently solving the timing subproblem. Computational results are presented in Section 5, followed by the conclusion in Section 6.

2. Problem formulation

In this section we define the problem and discuss the synchronization model at satellites.

2.1. Problem statement

We introduce the two-echelon multiple-trip vehicle routing problem with satellite synchronization (2E-MTVRP-SS). We consider a city distribution center (CDC), a set of satellites V_s, a set of requests R, and two homogeneous fleets of vehicles K_1 and K_2 of capacity Q_1 and Q_2, based at o_1 and o_2. Each request r is located at the CDC at the beginning of the time horizon and must be delivered within the time window $[e_r, l_r]$ to a customer denoted by d_r (the set of customers is denoted by V_c). The quantity associated with r is q_r. No direct shipping from the CDC is allowed. Second-level vehicles can perform multiple trips, which may start at different satellites. As second-level vehicles are small we assume that they are empty every time they arrive at a satellite, a similar assumption is made in Crainic et al. (2009). Satellites have no storage capacity, thus requiring an exact synchronization between the vehicles of the two levels.

The 2E-MTVRP-SS is defined on a directed graph $G = (V, A)$, which reflects the two-level system. The first level is defined by $G_1 = (V_1, A_1)$ with $V_1 = \{o_1\} \cup \{CDC\} \cup V_s$ and $A_1 = \{(o_1, CDC)\} \cup \{(CDC, i) | i \in V_s\} \cup \{(i, j) | i, j \in V_s\} \cup \{(i, o_1) | i \in V_s\}$. The second level is defined by $G_2 = (V_2, A_2)$ with $V_2 = \{o_2\} \cup V_c \cup V_s$ and $A_2 = \{(o_2, i) | i \in V_s\} \cup \{(i, j) | i \in V_s, j \in V_c\} \cup \{(i, j) | i, j \in V_c\} \cup \{(i, j) | i \in V_c, j \in V_s\} \cup \{(i, o_2) | i \in V_c\}$. With each arc $(i, j) \in A = A_1 \cup A_2$ is associated a travel time t_{ij} and a travel cost c_{ij}. Each node i has a known service duration s_i. Solving the 2E-MTVRP-SS involves finding $|K_1|$ first-level routes and $|K_2|$ second-level routes, and a schedule for them, such that the capacity and time-related constraints are satisfied.

2.2. Transfer and synchronization at satellites

We define a transfer as the operation during which a first-level vehicle transfers one or more requests to a second-level vehicle at a satellite. Given the lack of storage capacity at the satellites, the two vehicles must be at the satellite at the same time. Thus, the first and second levels must be synchronized. In details, for a transfer to happen, the two vehicles:

- May need time to get ready for the transfer (for example if the first-level vehicle has a lift gate to open).
- Should spend some time transferring the items.
- Should get ready to leave the satellite (for example the second-level vehicle may have to sort items).

$s_R[t_1, t_2]$: Preparation time for the first-level vehicle.
s_T $[t_2, t_3]$: Time in common for the vehicles of both levels. For a transfer to occur, the two vehicles should spend at least this time together at the satellite.
s_L $[t_3, t_4]$: Sorting time for the second-level vehicle.

Fig. 1 illustrates the temporal aspects of a transfer. In this example, if $t_2 > t_1 + s_R$, the first-level vehicle must wait. Conversely,

if $t_2 < t_1 + s_R$ the second-level vehicle must wait for the first-level vehicle. If the first-level vehicle transfers requests to several second-level vehicles, it cannot leave before $\max_{i \in K_2} t_i + s_T$. In this paper, we assume that s_R, s_T, and s_L can reasonably be considered independent of the transferred quantity, without inducing a significant imprecision. This simplifying hypothesis allow to integrate those times into the travel times from and to the satellites. Thus, we later consider that all the transfer-related periods (s_R, s_T, s_L) are equal to zero. If a second-level vehicle returns several times to pick up requests from the same first-level vehicle at the same satellite, each visit corresponds to a different transfer.

2.3. Mathematical formulation

We present a mixed integer linear programming formulation for the 2E-MTVRP-SS. In the model, to represent the transfer of one request at one satellite, for each request r and satellite s, we create a node $v_{s,r}$ whose associated demand is $-q_r$, and we denote by $\tilde{V}_s = \{v_{i,r} | v_i \in V_s, r \in R\}$ all the satellites. For each vehicle k, we create a start node o_k and an end node o'_k ($\tilde{O} = \cup_k o_k$, $\tilde{O}' = \cup_k o'_k$).

The mathematical formulation is defined on a graph $G^{\text{math}} = (V^{\text{math}}, A^{\text{math}})$. The first level is $G_1^{\text{math}} = (V_1^{\text{math}}, A_1^{\text{math}})$ with $V_1^{\text{math}} = \tilde{O_1} \cup \tilde{O'_1} \cup \tilde{V_s}$ and $A_1^{\text{math}} = \{(o,i) | o \in \tilde{O_1}, i \in \tilde{V_s}\} \cup \{(i,j) | i,j \in \tilde{V_s}\} \cup \{(i,o') | i \in \tilde{V_s}, o' \in \tilde{O'_1}\} \cup O_1 \times \tilde{O_1}$. The second level is $G_2^{\text{math}} = (V_2^{\text{math}}, A_2^{\text{math}})$ with $V_2^{\text{math}} = \tilde{O_2} \cup \tilde{O'_2} \cup V_c \cup \tilde{V_s}$ and $A_2^{\text{math}} = \{(o,i) | o \in \tilde{O_2}, i \in \tilde{V_s}\} \cup \{(i,j) | i \in \tilde{V_s}, j \in V_c\} \cup \{(i,j) | i,j \in V_c\} \cup \{(i,j) | i \in V_c, j \in \tilde{V_s}\} \cup \{(i,o') | i \in V_c, o' \in O'_2\} \cup O_2 \times \tilde{O_2}$. We introduce the following variables:

$$x_{i,j}^k = \begin{cases} 1 & \text{if vehicle } k \text{ travels from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}$$

$h_i =$ service time at node i
$\quad \times$ (point in time when service at node i starts)

$u_i =$ load of the second-level vehicle after serving i.

We introduce two constants: M_h corresponds to the end of the planning horizon, and M_u is the sum of all the ordered quantities.

We use a classical approach in vehicle routing problems with time windows, which consists in lexicographically minimizing the fleet-size and the routing cost. The first objective is the number of first-level vehicles, the second objective is the number of second-level vehicles and the third objective is the sum of arc costs.

$$lex - min\left(\sum_{k \in K_1} \sum_{j \in \tilde{V_s}} x_{o_k,j}, \sum_{k \in K_2} \sum_{j \in \tilde{V_s}} x_{o_k,j}, \sum_{k \in K_1} \sum_{(i,j) \in A_1^{\text{math}}} c_{i,j} \times x_{i,j}^k + \sum_{k \in K_2} \sum_{(i,j) \in A_2^{\text{math}}} c_{i,j} \times x_{i,j}^k \right) \quad (1)$$

s.t.

$$\sum_{(o_k,j) \in A_e^{\text{math}}} x_{o_k,j}^k = \sum_{(j,o'_k) \in A_e^{\text{math}}} x_{j,o'_k}^k = 1 \quad \forall e \in \{1,2\}, \forall k \in K_e \quad (2)$$

$$\sum_{(i,j) \in A_e^{\text{math}}} x_{i,j}^k = \sum_{(j,i) \in A_e^{\text{math}}} x_{j,i}^k \quad \forall e \in \{1,2\}, \forall k \in K_e, \forall i \in V_e^{\text{math}} \setminus (\tilde{O_e} \cup \tilde{O'_e}) \quad (3)$$

$$\sum_{k \in K_2} \sum_{(j,d_r) \in A_2^{\text{math}}} x_{j,d_r}^k = 1 \quad \forall r \in R \quad (4)$$

$$\sum_{k \in K_e} \sum_{j \in V_{s,r}} \sum_{(i,j) \in A_e^{\text{math}}} x_{i,j}^k = 1 \quad \forall e \in \{1,2\}, \forall r \in R \quad (5)$$

$$\sum_{k \in K_1} \sum_{i \in V_1^{\text{math}}} x_{i,v_{s,r}}^k = \sum_{k \in K_2} \sum_{i \in V_2^{\text{math}}} x_{i,v_{s,r}}^k \quad \forall v \in V_s, \forall r \in R \quad (6)$$

$$\sum_{v \in V_{s,r}} \sum_{(i,v) \in A_2^{\text{math}}} x_{i,v}^k = \sum_{(j,d_r) \in A_2^{\text{math}}} x_{j,d_r}^k \quad \forall r \in R, \forall k \in K_2 \quad (7)$$

$$h_j \geq h_i + s_i + t_{i,j} - M_h \times \left(1 - \sum_{k \in K_e} x_{i,j}^k\right) \quad \forall e \in \{1,2\}, \quad \forall (i,j) \in A_e^{\text{math}} \quad (8)$$

$$h_{o_k} \geq 0 \quad \forall k \in K \quad (9)$$

$$e_i \leq h_i \leq l_i \quad \forall i \in V_c \quad (10)$$

$$h_{o'_k} \leq M_h \quad \forall k \in K \quad (11)$$

$$\sum_{r \in R} q_r \times \sum_{j \in V_{s,r}} \sum_{(i,j) \in A_1^{\text{math}}} x_{i,j}^k \leq Q_1 \quad \forall k \in K_1 \quad (12)$$

$$u_j \geq u_i - q_i - M_u \times \left(1 - \sum_{k \in K_2} x_{i,j}^k\right) \quad \forall (i,j) \in A_2^{math} \quad (13)$$

$$u_j \leq M_u \times \left(1 - \sum_{k \in K_2} \sum_{i \in V_2^{\text{math}} \setminus \tilde{V_s}} x_{i,j}^k\right) \quad \forall j \in \tilde{V_s} \quad (14)$$

$$0 \leq u_i \leq Q_2 \quad \forall i \in V_2^{\text{math}} \quad (15)$$

$$x_{i,j}^k \in \{0,1\} \quad \forall k \in K, (i,j) \in A^{\text{math}} \quad (16)$$

The objective function (1) lexicographically minimizes the fleet-size and the travel costs. Constraints (2) state that each vehicle must start and end its route at its base. Constraints (3) are flow conservation constraints. Constraints (4) ensure that each request is delivered. Constraints (5) ensure for each request that only one transfer node is used, and (6) ensure that it is visited by both a first and a second-level vehicle. Constraints (7) ensure for each request that the second-level vehicle that visits the transfer node is the one that delivers the request. Constraints (8) compute the travel time between two nodes if they are visited consecutively by the same vehicle, and constraints (9) handle the special case of bases for which there is no predecessor. Constraints (10) ensure that each request is delivered within its time window. Constraints (11) ensure that each vehicle has completed its route within the time horizon. Constraints (12) (resp. (15)) ensure for each first-level (second-level) vehicle that the load does not exceed the vehicle capacity. Constraints (13) ensure for each second-level vehicle that the load after visiting a node is equal to the load before plus (or minus) the quantity that has been loaded (unloaded). Constraints (14) ensure that each second-level vehicle is empty when arriving at a satellite.

3. An ALNS for the 2E-MTVRP-SS

In this section we describe the destruction and repair methods used in our ALNS for the 2E-MTVRP-SS. ALNS was proposed by Ropke and Pisinger (2006) as an extension of the large neighborhood search introduced by Shaw (1998). The general principle of ALNS is described in Algorithm 1: it iteratively destroys and repairs the current solution using heuristics, which are selected based on their past successes. Destroying here means removing a number p of requests from the current solution (p is bounded by some parameters), while repairing means inserting unplanned requests in the solution. Note that a solution s may be *incomplete*, if a set of requests $\mathcal{L}_s$ remained unplanned. The size of this set is then penalized in the objective function. The solution obtained after the destroy and repair operations is accepted if it satisfies

Algorithm 1: Adaptive large neighborhood search.

Data: Candidate solution s, destroy operators $\mathcal{N}^-$, repair operators $\mathcal{N}^+$
Result: The best found solution $s^\star$

$s^\star \leftarrow s$
while *stop-criterion not met* **do**
  $s' \leftarrow s$
  Destroy quantity: select a number p
  of requests to remove from the candidate solution
  Operator selection: select a destruction method $or \in \mathcal{N}^-$ and a repair method $oi \in \mathcal{N}^+$
  Removal: $\mathcal{L} \leftarrow \mathcal{L}_{s'} \cup or(s', p)$
  Insertion: $s' \leftarrow oi(s', \mathcal{L})$
  if $f(s') \leq f(s^\star)$ **then**
    $s^\star \leftarrow s'$
  Acceptance criterion: set the candidate solution s to s'
return $s^\star$

an acceptance criterion. As in Ropke and Pisinger (2006), we use a roulette-wheel mechanism as adaptive layer for the selection of destruction and repair methods, and a simulated annealing as acceptance criterion.

Since its introduction, ALNS has been used to solve the 2E-VRP (Hemmelmayr et al., 2012) as well as many complex vehicle routing problems (Azi et al., 2014; Bortfeldt, Hahn, Männel, & Mönch, 2015; Kovacs, Parragh, Doerner, & Hartl, 2012; Masson et al., 2013a; 2014; Pisinger & Ropke, 2007).

We refer to Ropke and Pisinger (2006) for a more detailed explanation of ALNS. In the following, we focus on the specific components of our method, namely the construction of the initial solution, and the destroy and repair methods.

3.1. Initial solution

We design a two-phase constructive algorithm to obtain the initial solution. First, we design the second level routes using a best insertion algorithm for a multiple-trip multiple-depot problem. In this heuristic the possible insertion of a customer are all the insertions in existing trips and all insertions by creation of a new trip. Then we create the first-level routes to supply the satellites according to the second-level routing plan previously created.

3.2. Destroy methods

When partially destroying a solution we select a method and a number p of requests to remove. Unless stated otherwise, this method is reused until p is reached. Following (Azi et al., 2014), we use three levels of destruction methods: workday, route, and customer.

3.2.1. Workday level

The following operators are used for first and second-level vehicles.

Random Vehicle Removal: we randomly remove a vehicle.

Least Used Vehicle Removal: we remove the vehicle with the smallest load. For the second level, the total load of a vehicle is defined as the sum of the load of each trip.

3.2.2. Route level

Random trip removal: we randomly remove a trip from the solution.

First-level stop removal: we randomly remove a first-level stop from the solution. The trips that get their requests from this stop are removed.

Trip related removal: this method is similar to that of (Azi et al., 2014). Trips are removed based on a proximity measure: we start by randomly selecting a trip and removing it. We then find the trip that contains the nearest customer to any customer in the trip just removed, and we remove that trip.

Synchronization-based trip removal: intuitively, a *good* synchronization occurs when the vehicles involved arrive at approximately the same time. If a second-level vehicle arrives a long time before (or after) the first-level vehicle there will be a long waiting time; this should be avoided. This method removes the trip for which the time between the arrival of the second-level vehicle and the arrival of the first-level vehicle is maximum.

3.2.3. Customer level

Random customer removal: we randomly remove a customer.

Worst removal: this operator is the same as in Ropke and Pisinger (2006). For each request we compute the difference in cost of the solution with and without this request. We then sort the requests from the largest to the smallest difference. And we remove the *worst* (largest difference) request. Some randomization is introduced to avoid repeatedly removing the same requests.

Related removal heuristics: these methods aim to remove related requests. Let the relatedness of requests i and j be $R(i, j)$. We use two distinct relatedness measures: distance and time. The distance measure is the distance between the delivery points of i and j. The time measure is the sum of the absolute gap between their start of service and the absolute gap between their latest delivery times. In both cases a lower $R(i, j)$ value indicates a great degree of relatedness.

We ran preliminary tests to compare these two measures with that of (Shaw, 1998), which groups time and distance into a single measure. The methods gives similar results, but we chose time and distance because they do not require parameter tuning.

History-based removal: This is inspired by Masson et al. (2013a) and removes requests that seem poorly placed in the current solution with regard to the best-known solutions. For requests r and r', let $\xi_{r,r'}$ be the number of solutions among the 50 best-known in which r' is a direct successor of r. For each request, let $\delta^-(r)$ (resp. $\delta^+(r)$ be its direct predecessor (resp. successor). For request r and satellite s, let $\chi_{s,r}$ be the number of solutions in which r is delivered via a transfer at s. For each request r, delivered in the current solution via a transfer at s, we define a score ϕ as follows:

$$\phi_r = \xi_{\delta^-(r),r} + \xi_{r,\delta^+(r)} + \chi_{s,r}.$$

Then we remove the p requests with the lowest scores.

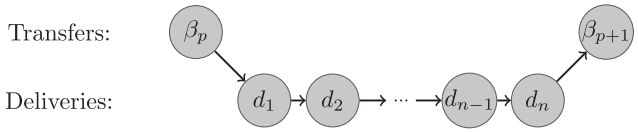
3.3. Repair methods

In this section we describe the methods used to repair a solution. We first describe the three different ways to insert a given customer into a given second-level route that we use, and then we describe the repair methods.

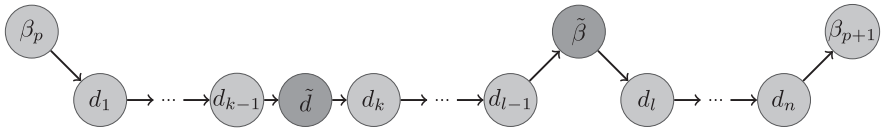
3.3.1. Three insertion operators

In the VRP, the insertion of a customer c into a partially built solution is fully described by giving the route and the position for the insertion. Thus, all possible insertions can be described by the unique set $\{(k, i) : \ k \in \text{Vehicles}, \ 0 \leq i \leq |route(k)| + 1\}$. Given the multiple-trip and two-echelon characteristics of the 2E-MTVRP-SS, we consider three distinct greedy ways of inserting a customer into a solution. We call them *insertion operators* and describe them below.

Insertion into an existing trip: The customer is inserted into an existing trip.



(a) Example of a second-level trip; β are transfer points and d_i are delivery points.



(b) Insertion into the first trip

Fig. 2. Example of the use of a trip split operator with insertion of the request into the first resulting trip.

Insertion by creation of a new trip: A new trip is created for the customer. This new trip can be connected either to an existing stop or to a newly created stop of a first-level vehicle.

Insertion by trip split: Before inserting c into trip t, we split t into two trips, t_1 and t_2. Trip t_1 is still connected to the same first-level stop as t, but t_2 is connected either to an existing stop or to a newly created stop of a first-level vehicle. This vehicle can be different from that involved in t_1. Then c is inserted into one of the two resulting trips. Fig. 2 illustrates the use of a trip split operator.

3.3.2. Reducing the size of the neighborhoods

The insertion of a request corresponds to two decisions: one at the first level and one at the second level. Thus, the neighborhoods generated by the insertion operators are huge. For example, to test every possible insertion of a request r with the *trip split* operator, we have to test for each pair (*insertion position*, *split position*) in each trip of second-level vehicles, each existing stop of first-level vehicles, and each creation of a stop (i.e., each satellite at each position). To ensure a reasonable runtime, we have created restricted neighborhoods.

For the *trip split* we have introduced two variants: *existing stops* and *customer first*. In the *existing stops* variant, we only try to connect t_2 with an existing stop of a first-level vehicle. In the *customer first* variant, we first select the insertion position that leads to the smallest increase in the cost of the global solution. Then we select the best possible way to split t into feasible trips t_1 and t_2, trying both existing and newly created first-level stops.

When we create a new first-level stop to be connected to a second-level trip, it is likely that the best choice for the satellite will be close to the second vehicle. At the beginning of our algorithm we sort the satellites in order of distance for every pair of customers. When creating a new trip connected to a new first-level stop, we consider only the s satellites closest to the pair (predecessor of the trip, first customer in the trip). This is used for the *trip creation* and the variants of the *trip split* operator.

As shown in Section 5.3.3, using these restricted neighborhoods makes our algorithm about 2.3 times faster while maintaining the quality of the solution.

3.3.3. Repair methods

All the unplanned requests are stored in a request bank.

Best insertion: From the requests in the request bank, we insert the one with the cheapest insertion cost considering all possible insertion operators.

K-Regret: For each request in the request bank, let δ_r^i represent the gap between the insertion of r at its best position in its best trip and the insertion at its best position in its i^{th} best trip. We select the request where $\sum_{i=2}^{k} \delta_r^i$ is maximum. In other words, we maximize the sum of the differences of the cost of inserting request r into its best trip and its i^{th} best trip. To control the computational time, we use small values of k.

Table 1
Notation in the temporal graph.

Notation	Definition
$\psi_{u,v}$	Ordered set of vertices on path $(u, \ldots, v)$
$\delta^+(i)$	Direct successors of vertex i
$\Gamma^+(i)$	Set of all successors of i
$\Omega_{u,v}$	Set of paths from u to v

3.3.4. First-level routes

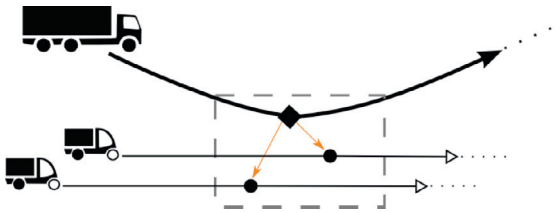
Inserting a request r by moving a first-level vehicle to a new satellite generates a large increase in the routing cost compared to transferring the request at an existing transfer point. Thus, it is rare for repair methods to choose such insertions. However, subsequent insertions may benefit from a new transfer point, because the second-level vehicle may have a smaller distance to travel. Therefore, when an insertion operator creates a new stop, we consider the following *biased* cost:

$$\begin{aligned}\text{Biased cost} &= \text{second-level insertion cost} \\ &+ \text{first-level insertion cost} \\ &\times \max\left(\alpha, \frac{\text{load in second-level trip}}{\text{second-level vehicle capacity}}\right).\end{aligned} \tag{17}$$

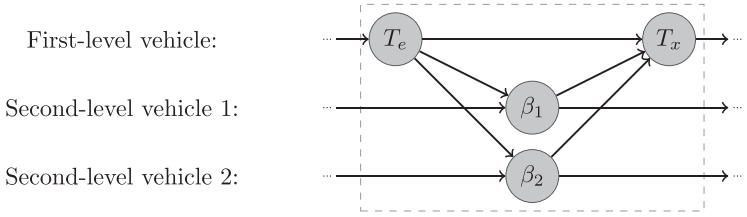
In this biased cost, we acknowledge that if there is some room in the second-level trip, then it is likely that we will later use it for a customer. For our instances, after some tuning, we have used $\alpha = 0.7$ for the *trip creation* operator and $\alpha = 0$ for the *trip split* operator and its variants.

4. Route scheduling and feasibility algorithm

For each actual insertion, the repair methods evaluate thousands of potential insertions in terms of profitability and feasibility. In this section we describe an efficient way to test if an insertion is feasible with respect to the temporal constraints. The method is an adaptation of the feasibility algorithm designed by Masson, Lehuédé, and Péton (2013b) for the PDPT, which was based on forward-time slacks (FTSs) (Savelsbergh, 1985). We introduce the idea behind FTSs in Section 4.1, give a way to model time constraints in Section 4.2, and describe the feasibility tests in Section 4.3. Throughout this section, we use the notation of Table 1.



(a) First-level stop with two transfers



(b) Precedence graph representation.

Fig. 3. First-level stop with two transfers and its precedence graph representation.

4.1. Efficient feasibility test for the vehicle routing problem with time windows

Inserting a request into a feasible route with an *as early as possible schedule* may postpone deliveries later in this route, potentially violating time windows. For a given insertion, we can check for this kind of violation by rescheduling the operations that follow the insertion. This test has a linear complexity in the size of the route. For the same test, (Savelsbergh, 1985) introduced an incremental constant-time algorithm. It is based on the computation of FTS variables that have to be maintained after each route modification. The FTSs can be calculated as follows: let u and v be two nodes in the same route, with v being delivered after u. For all vertices i in the route, let h_i be the current service time at i, w_i the waiting time before the service, and l_i the latest time for service at i. The total waiting time on the path $(u, \ldots, v)$ is

$$TWT_{\psi_{u,v}} = \sum_{i \in \psi_{\delta^+(u),v}} w_i. \tag{18}$$

The FTS at node u is

$$F_u = \min_{i \in \{u\} \cup \Gamma^+(u)} \{TWT_{\psi_{u,i}} + l_i - h_i\} \tag{19}$$

An intuitive explanation of Eq. (19) is the following: for every successor i of u, the path $(u, \ldots, i)$ can be decomposed into a working time and an idle/waiting time. If u is postponed, first the idle time will be reduced and then the service at node v will be postponed. The maximum postponement of v such that the time window at i is not violated is the total waiting time between u and i plus the difference between the current start time at i and the end of its time window. This is true for every indirect successor of v, and thus for the minimum over all successors.

A direct corollary of this result is that if the service at u is postponed by a value δ, then the new service time $\tilde{h_v}$ at v is

$$\tilde{h_v} = h_v + \max(0, \delta - TWT_{u,v}). \tag{20}$$

4.2. Modeling time constraints in the 2E-MTVRP-SS

In contrast to the VRP, in problems with synchronization, a change in the schedule of one route may affect other routes, potentially making them infeasible with respect to time. For example, in the 2E-MTVRP-SS, if we insert a delivery into the trip of a second-level vehicle A, it may arrive later at its next transfer, thus delaying the first level-vehicle with which it is synchronized. The first-level vehicle will propagate this delay to second-level vehicles at its next transfer, and so on, eventually causing a time-window violation for a second-level vehicle B whose link with A was not obvious at first sight. This is known as the *interdependence problem* (Drexl, 2012), and we model it through a precedence graph.

4.2.1. Precedence graph definition

Given the routes in a solution of the 2E-MTVRP-SS, we can represent the time constraints as a directed acyclic graph, G_t. We refer to this as a *precedence graph* and build it as follows: for every operation except transfers, we create a node and add an arc to each of its direct successors in the given route. Each arc (u, v) has a weight that corresponds to $t_{u,v}$. For a transfer we create three nodes: a transfer entrance node T_e, a transfer exit node T_x for the first-level vehicle, and a pick-up node β for the second-level vehicle. We create three arcs (T_e, T_x), (T_e, β), (β, T_x) with weight 0. If the first-level vehicle transfers its load to several second-level vehicles, we create only one pair (T_e, T_x). We assume that transfers to second-level vehicles can occur simultaneously. Fig. 3 illustrates this transformation.

4.2.2. Route scheduling

G_t corresponds to a PERT chart. As mentioned in Cormen, Leiserson, Ronald, and Stein (2010) (p. 657), we can schedule tasks in such a diagram using a shortest-path algorithm, with linear complexity in the number of vertices. Furthermore, in an as-early-as-possible schedule, only the downstream operations have to be rescheduled after a change, thus reducing the number of modifications.

4.3. Using the precedence graph for efficient feasibility testing

With G_t we can check if an insertion is feasible with respect to time by rescheduling the downstream operations. However, this is more time-consuming than in the VRP, because it is linear in the number of operations in the solution (versus the number of operations in a route in the VRP). Masson et al. (2013b) were faced with a similar problem for the PDPT. They introduced a directed acyclic graph to model their precedence constraints and extended the FTSs to obtain a constant-time feasibility check for the time constraints. We now present their main results and show how they can be applied to the 2E-MTVRP-SS.

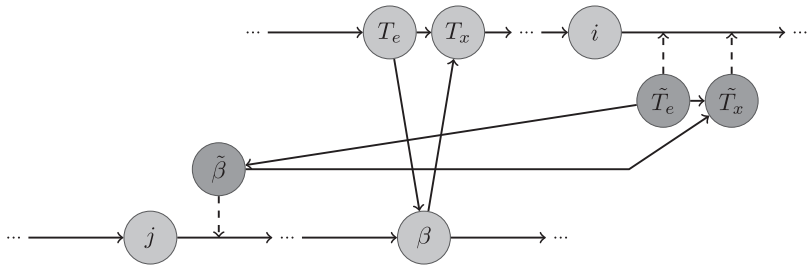


Fig. 4. Infeasible insertion: it creates a cycle in the precedence graph.

4.3.1. Extension of FTSs to directed acyclic graph modeling precedence constraints (Masson et al., 2013b)

First, Masson et al. introduced the notion of *slack time*. This is a generalization of the total waiting time between two nodes:

$$ST_{u,v} = \min_{w \in \Omega_{u,v}} TWT_w.$$

The FTS at node u becomes

$$F_u = \min_{i \in \{u\} \cup \Gamma^+(u)} \{ST_{u,i} + l_i - h_i\}. \tag{21}$$

The intuitive explanation given for Eq. (19) also holds for Eq. (21), with slack times accounting for the fact that there may be several time paths between two vertices. Masson et al. proved the above result in their Proposition 2. Their proof does not rely on the particular structure of their precedence graph for the PDPT, so it is valid for any directed acyclic graph. In particular, this result (and the start-of-service result below) is valid for our precedence graph.

Similarly to the VRP, if the start of service of node u is postponed by δ, the new service time $\tilde{h}$ of any successor v is

$$\tilde{h_v} = h_v + max(0, \delta - ST_{u,v}). \tag{22}$$

Note that in contrast to Masson et al., we recompute the FTSs using the Floyd–Warshall algorithm ((Cormen et al., 2010), p. 693), which is faster in our case than the suggested shortest path method.

4.3.2. Efficient feasibility test for the 2E-MTVRP-SS

When evaluating an insertion of an unplanned request into a feasible solution, we need to ensure that the solution will remain feasible with respect to time after the insertion. With the insertion operators of Section 3.3.1, infeasibility can occur in two ways: the creation of a cycle of precedence constraints or the violation of a time window.

4.3.2.1. Cycle detection. Some insertion operators create new transfers, which may lead to infeasible precedence relations (see Fig. 4 for an illustration). We extend the method of (Masson et al., 2013b) for detecting cycles in constant time.

Proposition 1. *Synchronizing a first-level stop T and a second-level trip $(\beta \to d_1 \dots d_n)$ creates a cycle in the precedence graph if and only if*

$$T_e \in \Gamma^+(\beta) \text{ or } \beta \in \Gamma^+(T_x).$$

Proof. $\Rightarrow$ With the synchronization, just two arcs are created ($T_e \to \beta$) and ($\beta \to T_x$). One of them is responsible for the cycle. If it is ($T_e \to \beta$), then previously $T_e \in \Gamma^+(\beta)$. If it is ($\beta \to T_x$), then previously $\beta \in \Gamma^+(T_x)$.

$\Leftarrow$ If $T_e \in \Gamma^+(\beta)$, then since $\beta \in \delta^+(T_e)$ by definition, a cycle is created. If $\beta \in \Gamma^+(T_x)$, then since $T_x \in \Gamma^+(\beta)$ by definition, a cycle is created. □

Corollary 1. *Synchronizing a new first-level stop inserted after node i and a new second-level trip inserted after node j creates a cycle in G_t if and only if*

$$i \in \Gamma^+(j) \text{ or } j \in \Gamma^+(i).$$

Provided we maintain a successor matrix, which can easily be computed together with the slack times, we can check in constant time if an insertion creates a cycle in G_t.

4.3.2.2. Use of FTSs. It is easy to use FTSs to check the feasibility with respect to time windows of an *insertion into an existing trip* or an *insertion by creation of a new trip*. For an *insertion into an existing trip*, we check that the shift of the operations of the second-level vehicle right after the insertion is smaller than its FTSs. For an *insertion by creation of a new trip*, we check that the shift of the operations of the second-level vehicle right after the new delivery and the shift of the operations of the first-level vehicle right after the new transfer exit are smaller than their respective FTSs. The case of an *insertion by trip split* is more complex; it is illustrated in Fig. 5 and the details are given in Algorithm 2 . For all the insertion operators, with FTSs we get a constant-time feasibility check.

Algorithm 2: Evaluation procedure for the feasibility of an insertion with trip split.

Result: return **true** if the insertion illustrated in Figure 5 is feasible, and **false** otherwise

$\bar{h}_{\tilde{T}_e} \leftarrow \max(h_i + s_i + t_{i,\tilde{T}_e}, e_{\tilde{T}_e})$
$\bar{h}_{\tilde{d}} \leftarrow \max(h_j + s_j + t_{j,\tilde{d}}, e_{\tilde{d}})$
if $\bar{h}_{\tilde{T}_e} > l_{\tilde{T}_e}$ **or** $\bar{h}_{\tilde{d}} > l_{\tilde{d}}$ **then**
  return false
$\bar{h}_{\sigma(j)} \leftarrow \max(\bar{h}_{\tilde{d}} + s_{\tilde{d}} + t_{\tilde{d},\sigma(j)}, e_{\sigma(j)})$
$\delta_{\sigma(j)} \leftarrow \bar{h}_{\sigma(j)} - h_{\sigma(j)}$
if $\delta_{\sigma(j)} > F_{\sigma(j)}$ **then**
  return false
$\bar{h}_k \leftarrow h_k + \max(\delta_{\sigma(j)} - ST_{\sigma(j),k}, 0)$
$\bar{h}_{\tilde{\beta}} \leftarrow \max(\bar{h}_{\tilde{T}_e}, \bar{h}_k + s_k + t_{k,\tilde{\beta}})$
$\bar{h}_{\sigma(k)} \leftarrow \max(\bar{h}_{\tilde{\beta}} + t_{\tilde{\beta},\sigma(k)}, e_{\sigma(k)})$
$\delta_{\sigma(k)} \leftarrow \bar{h}_{\sigma(k)} - h_{\sigma(k)}$
if $\delta_{\sigma(k)} > F_{\sigma(k)}$ **then**
  return false
$\bar{h}_{\tilde{T}_x} \leftarrow \bar{h}_{\tilde{\beta}}$
$\bar{h}_{\sigma(i)} \leftarrow \max(\bar{h}_{\tilde{T}_x} + t_{\tilde{h}_{\tilde{T}_x},\sigma(i)}, e_{\sigma(i)})$
$\delta_{\sigma(i)} \leftarrow \bar{h}_{\sigma(i)} - h_{\sigma(i)}$
if $\delta_{\sigma(i)} > F_{\sigma(i)}$ **then**
  return false
return true

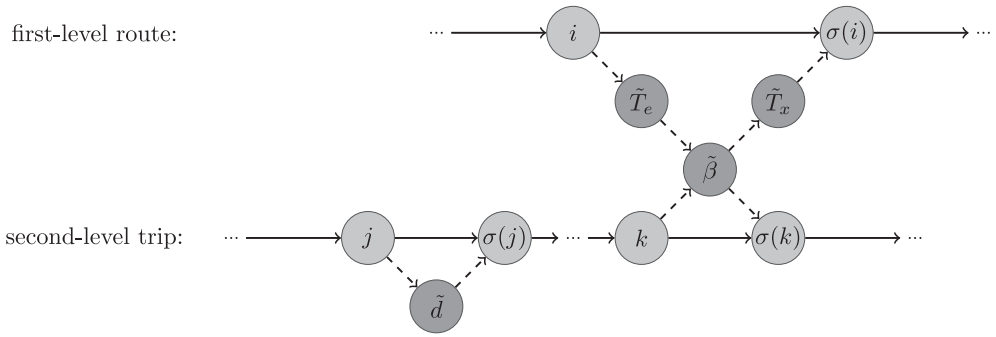


Fig. 5. Example of insertion with split trip in precedence graph. Customer $\tilde{d}$ is inserted between vertices j and $\sigma(j)$ in a second-level trip. This trip is split and connected at a new satellite $\tilde{\beta}$ between vertices k and $\sigma(k)$. The new trip is supplied at a new stop in a first-level route, which is inserted between vertices i and $\sigma(i)$. .

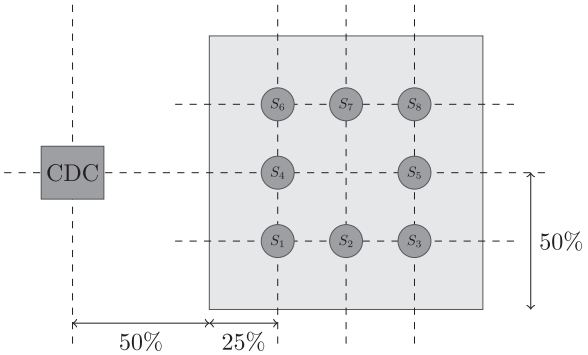


Fig. 6. Geometrical layout −50 / 50 / 3 / 3.

4.3.3. Efficiency of the method

We compare the runtime of the FTS extension and that of a check based on an incremental PERT: on average our algorithm is approximately 12 times faster with the former (see Section 5.3.2).

5. Computational experiments

In this section, we first describe the adaptation of some well-known VRPTW instances to the 2E-MTVRP-SS. Parameters configuration is discussed in Section 5.2. In Section 5.3 we show that our custom heuristics are efficient, and we present our results.

5.1. Instances

Since the 2E-MTVRP-SS is a new problem, there are no instances for it. We adapt the well-known Solomon's instances for the VRPTW (Solomon, 1987). We use a subset of these instances to tune our algorithm: the first two of every type, for a total of 12 tuning instances.

5.1.1. Geographical configuration

The customer requests are unchanged. The depot (node 0) is the base of second-level vehicles; the first-level vehicles are based at the CDC.

We adopt the following $X/Y/M/N$ notation to describe the position of the CDC and the satellites. X and Y give the position of the CDC expressed as a percentage of the size of the map. M and N describe the number of rows (resp. columns) of a grid. We locate a satellite at each exterior intersection of the grid. Fig. 6 illustrates a −50 / 50 / 3 / 3 configuration.

According to (Crainic et al., 2010), the maximum benefit of the 2E-VRP compared to the VRP occurs when the CDC is *external* (outside the customer's zone), thus saving travel from the depot to the customers and back, and when the satellites are between the CDC and the customers. The appropriate satellite number is between 7 and 10 for instances with between 100 and 200 customers. In all the benchmarks we use a 50 / 150 / 3 / 3 configuration.

Because the CDC is situated farther from the customers than the depot, some orders may be impossible to deliver. We therefore add an offset δ to each time window, with $\delta = \lceil t_{CDC,Depot} \rceil$. Hence, a time window $[e_i, l_i]$ in a Solomon instance becomes $[e_i + \delta, l_i + \delta]$.

5.1.2. Vehicle configuration

According to Savelsbergh, the instances labeled with a 1 (R1, C1, RC1) have short scheduling horizons, whereas instances labeled with a 2 have long scheduling horizons. Thus, the former are more time-constrained, and the latter are more capacity-constrained. To preserve this idea we use the following ratios for first-level vehicles capacity/second-level vehicles capacity: 4/0.5 ratio for instances 1; 2/0.25 ratio for instances 2.

5.1.3. Objectives

As the number of vehicles is not fixed, our algorithm is used twice. In a first phase we try to minimize the number of vehicles used by minimizing first the number of first level vehicles and then the number of second level vehicles. In a second phase, given the fleet obtained in the first phase, we minimize the routing cost.

5.2. Parameters configuration

In this section we discuss the parameters used in our algorithm.

5.2.1. Parameters for fleet optimization phase

We use a sequential optimization scheme to reduce the number of vehicles. We start by reducing the number of first-level vehicles and then we reduce the number of second-level vehicles. When we find a feasible solution with $n+1$ vehicles, we start looking for a feasible solution with n vehicles by calling the least-used-vehicle removal heuristic. $LB_1 = \lceil \sum_{r \in R} d_r / q_1 \rceil$ is a lower bound on the number of first-level vehicles. If we reach it, we switch to the reduction of the number of second-level vehicles. Overall, we perform a maximum of 25,000 iterations with no more than 12,500 dedicated to the reduction of the first-level fleet. As (Ropke & Pisinger, 2006), we stop the search if 5 or more requests are unplanned and no improvement in the number of unplanned requests has been found in the last 2000 iterations.

5.2.2. Parameters for cost optimization phase

In the cost optimization phase we use the following parameters $(w, c, \sigma_1, \sigma_2, \sigma_3, r, \rho_{min}, \rho_{max}) = (0.05$ percent, 0.99975 percent, 33 percent, 9 percent, 13 percent, 0.1 percent, 10 percent, 40 percent). The notation is that of (Ropke & Pisinger, 2006). We perform 25,000 iterations, since this gives a compromise between runtime and solution quality.

Table 2
Comparison of the results of the fleet optimization phase, using best insertion and two-phase best insertion as initialization methods. The FL (resp. SL) columns indicate the number of first-level (resp. second-level) vehicles. Bold figures indicate best solutions; italics indicate best average values.

Instance	Average initial solution				Average solution after fleet opt.				Best solution after fleet opt.			
	Best insertion		Two phase		Best insertion		Two phase		Best insertion		Two phase	
	FL	SL	FL	SL	FL	SL	FL	SL	FL	SL	FL	SL
c101	3.0	13.0	3.16	13.0	3.0	11.32	*3.0*	*11.28*	**3**	**11**	**3**	**11**
c102	3.0	13.0	3.04	12.0	**3.0**	**10.0**	**3.0**	**10.0**	**3**	**10**	**3**	**10**
c201	2.0	4.0	2.0	4.0	2.0	4.0	*2.0*	*3.72*	2	4	**2**	**3**
c202	2.0	4.0	2.0	4.0	2.0	3.24	*2.0*	*3.04*	**2**	**3**	**2**	**3**
r101	2.0	22.0	2.84	22.0	2.0	19.72	*2.0*	*19.64*	**2**	**19**	**2**	**19**
r102	2.0	21.0	2.88	20.0	2.0	18.08	**2.0**	**18.00**	**2**	**18**	**2**	**18**
r201	1.0	5.0	2.0	5.0	**1.0**	**4.0**	**1.0**	**4.0**	**1**	**4**	**1**	**4**
r202	1.0	5.0	2.43	5.0	*1.0*	*3.88*	1.0	3.90	**1**	**3**	**1**	**3**
rc101	3.0	21.0	4.28	20.0	*3.0*	*16.48*	3.0	16.56	**3**	**16**	**3**	**16**
rc102	3.0	18.0	5.24	17.0	3.0	14.4	*3.0*	*14.2*	**3**	**14**	**3**	**14**
rc201	1.0	6.0	3.0	5.0	**1.0**	**4.0**	**1.0**	**4.0**	**1**	**4**	**1**	**4**
rc202	1.0	5.0	2.0	5.0	**1.0**	**4.0**	**1.0**	**4.0**	**1**	**4**	**1**	**4**

Table 3
Computational time for incremental PERT versus FTS for 25,000 iterations in the cost optimization phase.

Instance	c101	c102	c201	c202	r101	r102	r201	r202	rc101	rc102	rc201	rc202
PERT (in minutes)	529.2	527.1	768.1	912.0	330.9	270.4	499.0	937.5	403.0	522.0	687.8	1599.0
FTS (in minutes)	**42.4**	**56.2**	**55.4**	**77.9**	**26.0**	**32.8**	**42.6**	**72.9**	**26.2**	**42.8**	**50.6**	**63.5**
Difference (in percent)	−92.0	−89.3	−92.8	−91.5	−92.1	−87.7	−91.5	−92.2	−93.5	−91.8	−92.6	−96.0

Table 4
Comparison of the average cost and runtime over 10 runs for 5 different neighborhood configurations. Configuration 1 uses the entire *trip split* neighborhood. In configuration 2 the *trip split* operator is replaced by its variants *customer first* and *existing stops*. In configurations 3–5 the number of satellites explored when creating a new first-level stop is limited to s, with $s = 2$ in neighborhood 3, $s = 3$ in neighborhood 4, and $s = 4$ in neighborhood 5. Bold figures indicate best values.

	Neighborhood 1		Neighborhood 2		Neighborhood 3		Neighborhood 4		Neighborhood 5	
Instance	Avg. cost	Avg. time	Avg. cost	Avg. time	Avg. cost	Avg. time	Avg. cost	Avg. time	Avg. cost	Avg. time
C101	2061.0	59.4	**2048.1**	48.3	2060.0	28.7	2053.7	33.5	2065.1	35.3
C102	1981.3	119.2	1984.0	61.7	1983.3	36.3	**1974.4**	42.7	1977.6	45.1
C201	1290.6	79.6	**1290.4**	38.4	1291.6	26.1	1293.8	27.7	1290.6	31.0
C202	1316.1	120.0	1311.3	64.8	**1309.4**	45.1	1311.4	48.9	1319.4	50.0
R101	**2343.5**	28.7	2352.0	27.8	2355.5	17.6	2353.2	20.8	2355.6	21.7
R102	2156.0	46.7	**2148.9**	31.7	2150.4	19.1	2158.1	23.0	2158.4	24.7
R201	1602.8	88.7	**1598.3**	33.1	1605.9	23.7	1604.5	26.8	1605.7	26.5
R202	1550.1	120.0	**1544.5**	49.9	1547.2	37.2	1546.1	40.0	1545.9	40.6
RC101	2624.0	34.2	2637.1	33.5	2616.3	20.7	**2609.0**	23.3	2613.0	23.3
RC102	2447.3	79.7	2451.2	46.9	2451.2	29.3	2446.9	33.9	**2435.1**	36.0
RC201	1821.3	104.8	1820.7	40.2	1812.5	29.2	**1809.3**	30.6	1811.0	32.0
RC202	1531.6	120.0	1534.8	49.5	**1525.0**	31.1	1527.1	37.8	1529.8	39.0
Dev. from Neigh. 1	–	–	−0.05 percent	−40.4 percent	−0.10 percent	−61.6 percent	−0.16 percent	−56.3 percent	−0.07 percent	−54.5 percent

5.2.3. Heuristics

For both the fleet optimization phase and the cost reduction phase, we use the following heuristics.

Destruction heuristics: random vehicle removal, least-used-vehicle removal, random trip removal, first-level stop removal, trip-related removal, synchronization-based trip removal, random customer removal, worst removal, distance-related removal, time-related removal, history-based removal.

Repair heuristics: best insertion, 3-regret, 4-regret, 5-regret. Each was used in three variants: without split, existing stops, and customer first (see Section 3.3.2).

5.3. Results

The algorithm was coded in C++, and the experiments were conducted using a single core of an Intel Xeon X5675 @ 3.07 GHz under Linux. We report figures for a subset of 12 instances, and the best solutions values and average values found for all instances out of ten runs.

5.3.1. Impact of two-phase initialization

For the construction of the initial solution, we compare the best insertion method of Section 3.3.3 and the dedicated two-phase best insertion method of Section 3.1. The latter alone produces worse results, but it is a better starting point for the fleet reduction; see Table 2.

5.3.2. Impact of FTS on reduction of runtime

On average the repair heuristics are far more time-consuming than the destruction heuristics (96.1 percent versus 1.1 percent of the total runtime). This partly comes from the fact that the repair methods have to reschedule the solution every time an insertion is performed, while the destruction methods reschedule the entire solution only when p customers have been removed from the current solution.

Table 3 compares the runtime of our algorithm using the FTS extension and that of an incremental PERT to check the feasibility of an insertion. It clearly shows that the FTS approach is key to reducing the computational effort: our algorithm is 91.9 percent

Table 5
Average and best solutions. FL (resp. SL) columns indicate the number of first-level (resp. second-level) vehicles. *T*1 (resp. *T*2) is the time spent in the fleet optimization (resp. cost optimization) phase. Bold figures indicate optimal values. Italics indicate that the average value is equal to its equivalent in the best known solution.

	Average					Best solution found				
Instance	FL	SL	T1 (minutes)	Cost	T2 (minutes)	FL	SL	T1 (minutes)	Cost	T2 (minutes)
c101	**3.0**	11.3	21.9	2057.4	30.5	**3**	11	8.3	2022.4	27.5
c102	**3.0**	*10.0*	48.4	1974.9	40.1	**3**	*10*	54.0	1947.6	45.6
c103	**3.0**	*9.0*	9.8	1946.8	46.9	**3**	*9*	13.4	1880.7	51.6
c104	**3.0**	*9.0*	8.4	1857.3	50.5	**3**	*9*	6.9	1811.1	51.5
c105	**3.0**	10.9	26.6	1959.2	35.1	**3**	10	26.4	1934.0	33.5
c106	**3.0**	10.8	32.2	1974.5	39.7	**3**	10	17.0	1945.0	34.0
c107	**3.0**	*10.0*	8.2	1902.8	35.5	**3**	*10*	5.8	1888.9	35.5
c108	**3.0**	*10.0*	8.8	1910.5	38.6	**3**	*10*	8.1	1875.3	34.1
c109	**3.0**	9.2	23.7	1921.3	46.1	**3**	9	5.2	1863.1	47.4
c201	**2.0**	3.5	27.8	1414.2	37.2	**2**	3	19.4	1389.3	33.9
c202	**2.0**	*3.0*	6.3	1317.9	45.4	**2**	*3*	3.6	1305.0	46.6
c203	**2.0**	3.1	9.2	1282.6	52.5	**2**	3	14.3	1272.7	51.8
c204	**2.0**	*3.0*	6.5	1261.3	57.5	**2**	*3*	9.1	1237.9	55.0
c205	**2.0**	3.1	6.8	1322.8	33.0	**2**	3	2.9	1312.1	33.0
c206	**2.0**	*3.0*	4.4	1328.6	36.1	**2**	*3*	4.8	1312.6	31.6
c207	**2.0**	*3.0*	3.9	1306.6	38.2	**2**	*3*	3.4	1280.4	39.5
c208	**2.0**	*3.0*	4.2	1300.0	37.3	**2**	*3*	3.2	1278.3	37.2
r101	**2.0**	19.6	28.9	2352.2	19.3	**2**	19	31.3	2333.5	17.0
r102	**2.0**	18.1	34.4	2152.8	21.7	**2**	18	30.5	2136.8	22.1
r103	**2.0**	13.8	45.2	1955.3	25.5	**2**	13	40.2	1942.7	22.2
r104	**2.0**	10.8	51.7	1804.8	47.2	**2**	10	44.4	1777.2	42.0
r105	**2.0**	14.9	27.7	2138.5	17.2	**2**	14	29.3	2096.8	11.5
r106	**2.0**	12.9	35.9	2028.0	26.8	**2**	12	23.8	1992.4	21.3
r107	**2.0**	11.2	37.7	1817.0	29.6	**2**	11	7.1	1779.2	29.9
r108	**2.0**	10.2	23.8	1702.2	36.6	**2**	10	17.8	1654.3	31.6
r109	**2.0**	12.9	30.3	1982.8	30.1	**2**	12	9.9	1925.9	26.6
r110	**2.0**	12.1	36.9	1888.9	33.6	**2**	12	39.6	1833.6	29.7
r111	**2.0**	*12.0*	44.3	1812.5	31.9	**2**	*12*	38.2	1770.8	29.7
r112	**2.0**	11.1	31.4	1800.9	46.9	**2**	11	54.5	1746.0	39.3
r201	**1.0**	*4.0*	8.2	1614.4	25.5	**1**	*4*	6.8	1587.8	26.6
r202	**1.0**	3.3	32.8	1548.3	39.9	**1**	3	36.6	1530.8	42.8
r203	**1.0**	*3.0*	12.6	1278.3	43.5	**1**	*3*	18.3	1255.1	44.4
r204	**1.0**	2.8	73.4	1200.1	49.3	**1**	2	6.2	1191.7	51.2
r205	**1.0**	*3.0*	6.7	1358.0	29.8	**1**	*3*	8.6	1319.1	31.0
r206	**1.0**	*3.0*	13.0	1251.5	39.1	**1**	*3*	17.5	1228.3	41.4
r207	**1.0**	*3.0*	20.7	1157.7	44.3	**1**	*3*	10.7	1140.2	43.7
r208	**1.0**	*2.0*	10.2	1082.5	49.6	**1**	*2*	5.3	1050.2	50.2
r209	**1.0**	*3.0*	9.2	1275.6	37.1	**1**	*3*	7.8	1258.7	46.8
r210	**1.0**	*3.0*	11.1	1300.4	42.7	**1**	*3*	11.5	1279.8	43.3
r211	**1.0**	*3.0*	15.9	1149.0	49.8	**1**	*3*	7.2	1118.2	44.6
rc101	**3.0**	16.4	26.5	2613.4	20.3	**3**	16	25.7	2577.0	19.7
rc102	**3.0**	14.3	34.8	2460.6	31.4	**3**	14	33.8	2407.1	29.2
rc103	**3.0**	12.0	41.7	2541.9	80.1	**3**	11	48.6	2476.9	72.7
rc104	**3.0**	11.1	41.1	2163.5	41.5	**3**	11	41.4	2125.9	39.7
rc105	**3.0**	15.4	33.1	2602.4	28.3	**3**	15	28.9	2542.6	30.0
rc106	**3.0**	13.9	27.4	2584.7	47.0	**3**	13	36.0	2494.9	47.8
rc107	**3.0**	13.1	41.1	2311.1	39.4	**3**	13	38.0	2271.1	40.4
rc108	**3.0**	12.2	38.9	2229.5	45.8	**3**	12	52.6	2202.9	41.5
rc201	**1.0**	*4.0*	6.8	1834.4	29.5	**1**	*4*	8.2	1787.6	25.4
rc202	**1.0**	*4.0*	65.2	1544.2	37.3	**1**	*4*	68.1	1513.8	36.9
rc203	**1.0**	*3.0*	8.3	1450.7	45.4	**1**	*3*	9.1	1416.2	44.3
rc204	**1.0**	*3.0*	10.0	1211.8	50.1	**1**	*3*	8.6	1188.2	42.8
rc205	**1.0**	*4.0*	27.4	1714.7	34.3	**1**	*4*	5.9	1693.7	30.1
rc206	**1.0**	3.4	34.8	1627.1	41.4	**1**	3	4.3	1583.1	35.3
rc207	**1.0**	*3.0*	10.0	1509.2	50.7	**1**	*3*	11.5	1449.8	49.8
rc208	**1.0**	*3.0*	9.8	1301.4	56.5	**1**	*3*	6.0	1257.3	55.9

faster with the FTS extension. With FTS, time-related functions account for 37.7 percent of the total runtime.

5.3.3. Impact of reduced neighborhoods on solution quality and runtime

Table 4 shows the impact of the reduced neighborhoods on the solution quality and the runtime. We observe that using these neighborhoods has a limited impact on the solution quality, but the runtimes are significantly reduced (by 56.3 percent). The runtime reduction is larger for type-2 instances, which are capacity constrained, than for type-1 instances, which are time constrained. This is expected, because the tighter the time constraint the smaller the number of reachable satellites. Based on these results we have chosen the following configuration (neighborhood 4 in Table 4): we use *customer first* and *existing stops* instead of the full *trip split* operator, and we limit the number of satellites explored to three when creating a new stop for a first-level vehicle.

5.3.4. Impact of custom destruction and repair methods on cost optimization phase

We now focus on the contribution of our new heuristics to the solution of the 2E-MTVRP-SS. The introduction of split-recreate heuristics leads to solutions that are 0.5 percent cheaper on

Table 6
Comparison of the number of stops in first-level routes, the number of trips in second-level routes, and the number of satellites used for each type of instance.

Type	Avg. no. stops in FL routes	Avg. no. trips in SL routes	No. of satellites used
C1	2.6	2.5	6.1
C2	2.7	4.8	5.2
R1	2.2	1.5	3.5
R2	2.7	2.7	2.3
RC1	2.6	1.8	5.6
RC2	3.8	2.7	3.3

Table 7
Comparison of the averaged best results for each instance type for the original 2E-MTVRP-SS, without time windows or synchronization.

Type	2E-MTVRP-SS		No TW		Precedence	
	SL	Cost	SL	Cost	SL	Cost
C1	9.8	1911.0	9	1701.7	9.7	1911.8
C2	3.1	1284.2	3	1185.5	3	1268.4
R1	12.8	1913.3	9	1595.6	13	1911.5
R2	2.9	1268.3	2	975.1	2.9	1265.4
RC1	13.3	2348.6	10	2018.5	13.1	2343.2
RC2	3.4	1476.3	2	1061.2	3.25	1525.6

average, showing that this neighborhood, although time-consuming, is worth considering.

The use of biased costs and synchronization-based removal do not have a significant impact on the quality of the solutions. However, they make our method more stable; this can be seen by comparing the standard deviations. When we normalize the average results for each instance to 100, the average standard deviations for all the test instances are 0.93 with all the heuristics, 0.97 without the synchronization-based removal, and 1.10 without the biased costs.

5.3.5. Solutions

In Table 5, we report the best and average results found based on 10 runs for each instance. In the fleet optimization phase, we observe that the number of first-level vehicles is always equal to *LB*1 and thus optimal. In the cost optimization phase, there is an average gap of 4.68 percent between the best solution and the average results. In Table 6, we evaluate the impact of the instance characteristics on the routing network used by the vehicles. We notice that there are on average more trips in second-level routes for type-2 instances (capacity-constrained), and that more satellites are used in instances with clustering.

5.4. Impact of time windows and synchronization on the cost of solutions

We now assess the impact of the time windows and the exact synchronization on the cost of the solutions. To evaluate the impact of the time windows, we remove them and re-solve the instances. The members of each group of Solomon instances differ only in their time windows, so when we remove the windows there are only six instances.

To evaluate the impact of exact synchronization, we run our algorithm on the instances of the 2E-MTVRP-SS with only a precedence constraint for transfers at satellites: a second-level vehicle can pick up goods only if a first-level vehicle has already dropped them. This corresponds to the case in which the satellites have storage capacity.

We ran our algorithm ten times for each instance. We selected the best result for each instance and then computed the average second-level fleet size and cost for each type of instance. Table 7 gives the results. We do not report the results for the first-level fleet size since our earlier tests for the 2E-MTVRP-SS achieved the lower bound.

Table 7 shows that considering a precedence constraint instead of the exact synchronization constraint has only a small impact on the second-level fleet size and the average solution cost. This is not surprising since in the 2E-MTVRP-SS solutions first-level vehicles tend to stay for long periods of time at the same satellite, acting as depots. A more realistic scenario would impose a maximum stay at the satellites and add waiting stations. Removing the time windows has a large impact on the second-level fleet size and the solution cost.

6. Conclusion

We have extended the well-known 2E-VRP to take into account constraints arising in city logistics (time windows, synchronization constraints, and multiple trips for some vehicles). We have developed an ALNS to solve the 2E-MTVRP-SS. Our approach has custom destruction and repair heuristics and an efficient way to check if an insertion is feasible. It can find good solutions in a reasonable time. Our tests show that time windows have a greater influence than exact synchronization on the solution cost. In future work, our algorithm could be applied to other VRPs with synchronization constraints.

Acknowledgments

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada through its Discovery Grants program and by the Fonds de recherche du Québec - Nature et technologies through its Team Research program.

References

Azi, N., Gendreau, M., & Potvin, J.-Y. (2014). An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers & Operations Research, 41*, 167–173.

Bortfeldt, A., Hahn, T., Männel, D., & Mönch, L. (2015). Hybrid algorithms for the vehicle routing problem with clustered backhauls and 3D loading constraints. *European Journal of Operational Research, 243*, 82–96.

Breunig, U., Schmid, V., Hartl, R. F., & Vidal, T. (2015). A fast large neighbourhood based heuristic for the two-echelon vehicle routing problem. Working Paper.

Cattaruzza, D., Absi, N., Feillet, D., & González-Feliu, J. (2015). Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, 1–29.

Cattaruzza, D., Absi, N., Feillet, D., & Vigo, D. (2014). An iterated local search for the multi-commodity multi-trip vehicle routing problem with time windows. *Computers & Operations Research, 51*, 257–267.

Cormen, T. H., Leiserson, C. E., Ronald, R. L., & Stein, C. (2010). *Introduction to algorithms* (3rd). The MIT Press.

Crainic, T. G., Mancini, S., Perboli, G., & Tadei, R. (2008). Clustering-based heuristics for the two-echelon vehicle routing problem. *Technical Report, CIRRELT-2008-46*. CIRRELT.

Crainic, T. G., Mancini, S., Perboli, G., & Tadei, R. (2011). Multi-start heuristics for the two-echelon vehicle routing problem. In P. Merz, & J.-K. Hao (Eds.), *Evolutionary computation in combinatorial optimization* (pp. 179–190). Springer.

Crainic, T. G., Mancini, S., Perboli, G., & Tadei, R. (2013). GRASP with path relinking for the two-echelon vehicle routing problem. *Advances in Metaheuristics, 53*, 113–125.

Crainic, T. G., Perboli, G., Mancini, S., & Tadei, R. (2010). Two-echelon vehicle routing problem: a satellite location analysis. *Procedia - Social and Behavioral Sciences, 2*(3), 5944–5955.

Crainic, T. G., Ricciardi, N., & Storchi, G. (2004). Advanced freight transportation systems for congested urban areas. *Transportation Research Part C: Emerging Technologies, 12*(2), 119–137.

Crainic, T. G., Ricciardi, N., & Storchi, G. (2009). Models for evaluating and planning city logistics systems. *Transportation Science, 43*(4), 432–454.

Cuda, R., Guastaroba, G., & Speranza, M. (2014). A survey on two-echelon routing problems. *Computers & Operations Research*, 1–15.

Drexl, M. (2012). Synchronization in vehicle routing–A survey of VRPs with multiple synchronization constraints. *Transportation Science, 46*(3), 297–316.

Hemmelmayr, V. C., Cordeau, J.-F., & Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research, 39*(12), 3215–3228.

Jepsen, M., Spoorendonk, S., & Ropke, S. (2013). A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem. *Transportation Science, 47*(1), 23–37.

Kovacs, A., Parragh, S., Doerner, K., & Hartl, R. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling, 15*(5), 579–600.

Mancini, S. (2013). Multi-echelon distribution systems in city logistics. *European Transport/Trasporti Europei, 54*. http://etabeta.univ.trieste.it/dspace/handle/10077/8868.

Masson, R., Lehuédé, F., & Péton, O. (2013a). An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science, 47*(3), 344–355.

Masson, R., Lehuédé, F., & Péton, O. (2013b). Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters, 41*(3), 211–215.

Masson, R., Lehuédé, F., & Péton, O. (2014). The dial-a-ride problem with transfers. *Computers & Operations Research, 41*, 12–23.

Masson, R., Trentini, A., Lehuédé, F., Malhéné, N., Péton, O., & Tlahig, H. (2015). Optimization of a city logistics transportation system with mixed passengers and goods. *EURO Journal on Transportation and Logistics*, 1–29.

Nguyen, P. K., Crainic, T. G., & Toulouse, M. (2013). A tabu search for time-dependent multi-zone multi-trip vehicle routing problem with time windows. *European Journal of Operational Research, 231*(1), 43–56.

Nguyen, V.-P., Prins, C., & Prodhon, C. (2012). Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking. *European Journal of Operational Research, 216*(1), 113–126.

Perboli, G., & Tadei, R. (2010). New families of valid inequalities for the two-echelon vehicle routing problem. *Electronic Notes in Discrete Mathematics, 36*, 639–646.

Perboli, G., Tadei, R., & Vigo, D. (2011). The two-echelon capacitated vehicle routing problem: models and math-based heuristics. *Transportation Science, 45*(3), 364–380.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research, 34*(8), 2403–2435.

Qu, Y., & Bard, J. F. (2012). A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research, 39*(10), 2439–2456.

Roberti, R. (2012). *Exact algorithms for different classes of vehicle routing problems*. Bologna. (Ph.D. thesis).

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science, 40*(4), 455–472.

Santos, F. A., Cunha, A. S., & Mateus, G. R. (2012). Branch-and-price algorithms for the two-echelon capacitated vehicle routing problem. *Optimization Letters, 7*(7), 1537–1547.

Santos, F. A., Mateus, G. R., & da Cunha, A. S. (2014). A branch-and-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Transportation Science, Articles i*(April), 1–14.

Savelsbergh, M. W. P. (1985). Local search in routing problems with time windows. *Annals of Operations Research, 4*(1), 285–305.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Fourth international conference on principles and practice of constraint programming* (pp. 417–431).

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research, 35*(2), 254–265.

Taillard, E. D., Laporte, G., & Gendreau, M. (1996). Vehicle routing with multiple use of vehicles. *Journal of the Operational Research Society, 47*, 1065–1070.

Zeng, Z.-Y., Xu, W., Xu, Z.-Y., & Shao, W.-H. (2014). A Hybrid GRASP+ VND heuristic for the two-echelon vehicle routing problem arising in City Logistics. *Technical Report*. School of Electronics and Information Engineering, Tongji University.

Bibliographie

ALK Technologies. Copilot® previews next generation intelligent gps navigation technology, 2014. URL http://blog.copilotgps.com/uk/2014/02/23/copilot-previews-next-generation-intelligent-gps-navigation-technology/. (Cité page 39.)

C. Bana e Costa and J. Vansnick. A theoretical framework for Measuring Attractiveness by a Categorical Based Evaluation TecHnique (MACBETH). In *Proc. XIth Int. Conf. on MCDM*, pages 15–24, Coimbra, Portugal, 1994. (Cité page 14.)

C. A. Bana e Costa, J.-M. De Corte, and J.-C. Vansnick. On the mathematical foundation of macbeth. In *Multiple Criteria Decision Analysis : State of the Art Surveys*, volume 78 of *International Series in Operations Research & Management Science*, pages 409–437. Springer New York, 2005. (Cité page 14.)

M. Battarra, J.-F. Cordeau, and M. Iori. *Pickup-and-Delivery Problems for Goods Transportation*, volume 18, pages 161–191. SIAM. (Cité page 50.)

N. Benabbou, P. Perny, and P. Viappiani. Incremental elicitation of choquet capacities for multicriteria decision making. In *ECAI*, pages 87–92, 2014. (Cité page 39.)

P. M. Berry, M. Gervasio, B. Peintner, and N. Yorke-Smith. Ptime : Personalized assistance for calendaring. *ACM Trans. Intell. Syst. Technol.*, 2(4) :40 :1–40 :22, July 2011. (Cité page 38.)

J. Branke, K. Deb, K. Miettinen, and R. Slowinski. *Multiobjective optimization : Interactive and evolutionary approaches*, volume 5252. Springer, 2008. (Cité pages 9 et 39.)

D. Bredström and M. Rönnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1) :19–31, 2008. (Cité page 44.)

R. Carraway, T. Morin, and H. Moskowitz. Generalized dynamic programming for multicriteria optimization. *European Journal of Operational Research*, 44 :95–104, 1990. (Cité page 19.)

D. Cattaruzza, N. Absi, D. Feillet, and J. GonzÃ¡lez-Feliu. Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, pages 1–29, 2015. (Cité page 43.)

G. Choquet. Theory of capacities. *Annales de l'Institut Fourier*, 5 :131–295, 1953. (Cité pages 11 et 15.)

N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *OR*, pages 309–318, 1969. (Cité page 70.)

J. Climaco and M. Pascoal. Multicriteria path and tree problems – discussion on exact algorithms. In *Proceedings of ALIO–INFORMS International Meeting*, Buenos Aires, may 2010. (Cité page 17.)

R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Welsey, 1967. (Cité page 71.)

J. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B : Methodological*, 37(6) :579 – 594, 2003. (Cité page 65.)

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, et al. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001. (Cité page 59.)

C. E. Cortés, M. Matamala, and C. Contardo. The pickup and delivery problem with transfers : Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3) :711–724, 2010. (Cité pages 46 et 66.)

T. G. Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2) :272–288, 2000. (Cité page 75.)

T. G. Crainic and G. Laporte. Planning models for freight transportation. *European journal of operational research*, 97 (3) :409–438, 1997. (Cité page 75.)

T. G. Crainic, N. Ricciardi, and G. Storchi. Models for evaluating and planning city logistics systems. *Transportation science*, 43(4) :432–454, 2009. (Cité pages 43 et 46.)

T. G. Crainic, M. Hewitt, M. Toulouse, and D. M. Vu. Service network design with resource constraints. *Transportation Science*, 2014. doi : 10.1287/trsc.2014.0525. (Cité page 75.)

K. L. Croxton, B. Gendron, and T. L. Magnanti. Models and methods for merge-in-transit operations. *Transportation Science*, 37(1) :1–22, 2003. (Cité page 77.)

M. Daskin. *Network and Discrete Location : Models, Algorithms and Applications*. Wiley Interscience, 1995. (Cité pages 5 et 79.)

R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3) :61–95, 1991. (Cité page 64.)

E. Demir, T. Bektaş, and G. Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2) :346–359, 2012. (Cité page 51.)

M. Drexl. Branch-and-Price and Heuristic Column Generation for the Generalized Truckand- Trailer Routing Problem. *Revista de Métodos Cuantitativos para la Economía y la Empresa*, 12 :5–38, 2011. (Cité page 44.)

M. Drexl. Synchronization in Vehicle Routing–A Survey of VRPs with Multiple Synchronization Constraints. *Transportation Science*, 46(3) :297–316, Mar. 2012. (Cité pages 43, 45, 46, 47 et 48.)

M. Drexl. A Generic Heuristic for Vehicle Routing Problems with Multiple Synchronization Constraints. *Gutenberg School of Management and Economics – Discussion Paper Series*, 1412, 2014. (Cité page 43.)

M. Ehrgott. *Multicriteria optimization*, volume 2. Springer, 2005. (Cité page 9.)

M. Ehrgott and X. Gandibleux. A survey and annoted bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22 :425–460, 2000. (Cité page 17.)

N. El Hachemi, M. Gendreau, and L.-M. Rousseau. A heuristic to solve the synchronized log-truck scheduling problem. *Computers & Operations Research*, 40(3) :666–673, 2013. (Cité page 44.)

A. L. Erera, M. Hewitt, M. W. Savelsbergh, and Y. Zhang. Creating schedules and computing operating costs for LTL load plans. *Computers & Operations Research*, 40(3) :691–702, 2013a. (Cité page 75.)

A. L. Erera, M. Hewitt, M. W. P. Savelsbergh, and Y. Zhang. Improved load plan design through integer programming based local search. *Transportation Science*, 47(3) :412–427, 2013b. (Cité page 75.)

D. Feillet, T. Garaix, F. Lehuédé, O. Péton, and D. Quadri. A new consistent vehicle routing problem for the transportation of people with disabilities. *Networks*, 63(3) :211–224, 2014. (Cité pages 4 et 69.)

J. Figueira, S. Greco, and M. Ehrgott. *Multiple criteria decision analysis : state of the art surveys*, volume 78 of *International Series in Operations Research & Management Science*. Springer, 2005. (Cité page 12.)

H. Fouchal. Optimisation combinatoire et préférences du décideur, 2009. (Cité page 39.)

H. Fouchal. *Optimisation de l'intégrale de Choquet pour le calcul de plus courts chemins multi-objectifs préférés*. Thèse de doctorat, Université de Nantes, 2011. (Cité pages 2, 16, 17 et 26.)

H. Fouchal, X. Gandibleux, and F. Lehuédé. Preferred solutions computed with a label setting algorithm based on choquet integral for multi-objective shortest paths. In *SSCI 2011 MCDM - 2011 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making*, Paris, France, 2011a. (Cité page 2.)

H. Fouchal, X. Gandibleux, and F. Lehuédé. A lower bound of the choquet integral integrated within martins' algorithm. In Y. Shi, S. Wang, G. Kou, and J. Wallenius, editors, *New State of MCDM in the 21st Century*, volume 648 of *Lecture Notes in Economics and Mathematical Systems*, pages 79–89. Springer Berlin Heidelberg, 2011b. (Cité page 2.)

V. Gabrel and D. Vanderpooten. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite. *European Journal of Operational Research*, 139(3) :533–542, 2002. (Cité page 19.)

L. Galand and P. Perny. Search for Choquet-optimal paths under uncertainty. In *Proceedings of 23rd conference on Uncertainty in Artificial Intelligence*, pages 125–132, Vancouver, July 2007. AAAI Press. (Cité pages 20 et 21.)

L. Galand, J. Lesca, and P. Perny. Dominance Rules for the Choquet Integral in Multiobjective Dynamic Programming. In *proceedings of IJCAI*, pages 538–544, 2013. (Cité page 30.)

X. Gandibleux, F. Beugnies, and S. Randriamasy. Martins' algorithm revisited for multi-objective shortest path problems with a maxmin cost function. *4OR : A Quarterly Journal of Operations Research*, 4(1) :47–59, March 2006. (Cité page 18.)

B. Gendron and F. Semet. Formulations and relaxations for a multi-echelon capacitated location–distribution problem. *Computers & Operations Research*, 36(5) :1335–1355, 2009. (Cité page 77.)

J. Gonzalez-Feliu. Vehicle Routing in Multi-Echelon Distribution Systems with Cross-Docking : A Systematic Lexical-Metanarrative Analysis. *Computer and Information Science*, 6(3) :28–47, 2013. (Cité page 43.)

J. Gonzalez-Feliu, F. Semet, and J. Routhier. *Sustainable urban logistics : concepts, methods and information systems*. Springer, 2014. (Cité page 43.)

M. Grabisch. Fuzzy integral in multicriteria decision making. *Fuzzy Sets & Systems*, 69 :279–298, 1995. (Cité pages 11 et 16.)

M. Grabisch. The application of fuzzy integrals in Multicriteria Decision Making. *European J. of Operational Research*, 89 :445–456, 1996. (Cité page 16.)

M. Grabisch. Alternative representations of discrete fuzzy measures for decision making. *Int. J. of Uncertainty,*

Fuzziness, and Knowledge Based Systems, 5 :587–607, 1997. (Cité pages 11, 15 et 16.)

M. Grabisch and C. Labreuche. Fuzzy Measures and Integrals in MCDA. In J. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis*, State of the Art. Kluwer Academic Press, 2005. (Cité page 11.)

M. Grabisch and C. Labreuche. A decade of application of the choquet and sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 175(1) :247–286, 2010. (Cité pages 11, 12 et 15.)

M. Grabisch, I. Kojadinovic, and P. Meyer. A review of capacity identification methods for Choquet integral based multi-attribute utility theory – applications of the KAPPALAB R package. *European Journal of Operational Research*, 186 :766–785, 2008. (Cité page 16.)

P. Grangier, M. Gendreau, F. Lehuédé, and L.-M. Rousseau. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *Accepted for publication European Journal of Operational Research*, Available under reference CIRRELT-2014-33 :17p, 2014. (Cité pages 4 et 47.)

A. Grimault, F. Lehuédé, and N. Bostel. A two-phase heuristic for full truckload routing and scheduling with split delivery and resource synchronization in public works. In *2014 International Conference on Logistics Operations Management*, pages 57–61. IEEE, June 2014. (Cité pages 5 et 48.)

C. Groër, B. Golden, and E. Wasil. The consistent vehicle routing problem. *Manufacturing & service operations management*, 11(4) :630–643, 2009. (Cité pages 69 et 70.)

P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making : Theory and Application*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 109–127. Springer, 1980. (Cité page 18.)

M. Henig. The shortest path problem with two objective functions. *European Journal of Operational Research*, 25(2) : 281–291, 1986. (Cité page 19.)

E. Jacquet-Lagreze and J. Siskos. Assessing a set of additive utility functions for multicriteria decision-making, the uta method. *European journal of operational research*, 10(2) :151–164, 1982. (Cité page 13.)

N. Jozefowiez, F. Semet, and E.-G. Talbi. An evolutionary algorithm for the vehicle routing problem with route balancing. *European Journal of Operational Research*, 195(3) :761 – 769, 2009. (Cité page 9.)

R. Keeney and H. Raiffa. *Decision with Multiple Objectives*. Wiley, New York, 1976. (Cité page 11.)

A. Klose and A. Drexl. Facility location models for distribution system design. *European Journal of Operational Research*, 162(1) :4–29, 2005. (Cité page 75.)

A. A. Kovacs, S. N. Parragh, K. F. Doerner, and R. F. Hartl. Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of scheduling*, 15(5) :579–600, 2012. (Cité page 44.)

A. A. Kovacs, B. L. Golden, R. F. Hartl, and S. N. Parragh. Vehicle routing problems in which consistency considerations are important : A survey. *Networks*, 64(3) :192–213, 2014. (Cité page 71.)

La compagnie des mobilités. Géovélo : Service de calcul d'itinéraire dédié aux cyclistes, 2014. URL `http://www.geovelo.fr/`. (Cité page 38.)

C. Labreuche. A general framework for explaining the results of a multi-attribute preference model. *Artif. Intell.*, 175 (7-8) :1410–1448, 2011. (Cité page 17.)

C. Labreuche and F. Lehuédé. Myriad : a tool suite for mcda. *Proceedings of EUSFLAT'05*, pages 204–209, 2005. (Cité pages 11 et 16.)

S. D. Lapierre, A. B. Ruiz, and P. Soriano. Designing distribution networks : Formulations and solution heuristic. 38 (2) :174–187, 2004. (Cité page 77.)

A. Lee and M. Savelsbergh. An extended demand responsive connector. *EURO Journal on Transportation and Logistics*, pages 1–26, 2014. (Cité page 44.)

F. Lehuédé and C. Labreuche. Inconsistencies in the determination of a capacity. In *ECAI 2006 Multidisciplinary Workshop on Advances in Preference Handling*, Riva del Garda, Italy, 2006. (Cité page 11.)

F. Lehuédé, P. Gérard, M. Grabisch, C. Labreuche, and P. Savéant. Integration of a Multicriteria Decision Model in Constraint Programming. In B. Brabble, J. Koehler, and I. Refanidis, editors, *Proceedings of the AIPS'02 Workshop on Planning and Scheduling with Multiple Criteria*, pages 15–20, Toulouse, France, 2002. (Cité page 2.)

F. Lehuédé, M. Grabisch, C. Labreuche, and P. Savéant. Multicriteria Search in Constraint Programming. In *Proceedings of CP-AI-OR'03*, pages 291–304, Montréal, Canada, 2003a. (Cité page 2.)

F. Lehuédé, M. Grabisch, C. Labreuche, and P. Savéant. Procédé permettant de produire des solutions à un problème concret d'optimisation multicritère. Technical Report Brevet Nationnal No 03 00898, THALES, 2003b. (Cité page 2.)

F. Lehuédé, M. Grabisch, C. Labreuche, and P. Savéant. Integration and propagation of a multi-criteria decision making model in constraint programming. *Journal of Heuristics*, 12(4-5) :329–346, September 2006a. (Cité pages 2 et 11.)

F. Lehuédé, M. Grabisch, C. Labreuche, and P. Savéant. MCS - a new algorithm for Multicriteria Optimisation in Constraint Programming. *Annals of Operations Research*, 147(1) :143–174, October 2006b. (Cité pages 2 et 11.)

F. Lehuédé, M. Grabisch, C. Labreuche, and P. Savéant. Integration and propagation of a multi-criteria decision making model in constraint programming. *Journal of Heuristics*, 12(4-5) :329–346, September 2006. (Cité page 38.)

F. Lehuédé, C. Pavageau, and O. Péton. Un système d'aide à la décision pour planifier les transports vers les établissements médico-sociaux. In *Handicap 2008*, page 6p, Paris, 2008. (Cité page 3.)

F. Lehuédé, C. Pavageau, and O. Péton. Outil d'optimisation pour le transport des personnes handicapées mentales. *Techiques de l'ingénieur*, Avril :0–7, 2009. (Cité pages 3 et 30.)

F. Lehuédé, R. Masson, S. N. Parragh, O. Péton, and F. Tricoire. A multi-criteria large neighbourhood search for the transportation of disabled people. *Journal of the Operational Research Society*, 65(7) :983–1000, 2013. (Cité pages 2, 31, 36 et 51.)

F. Lehuédé, O. Péton, and X. Tang. Optimization of customer orders routing in a collaborative distribution network. *EMN WORKING PAPER*, 14/6/AUTO, submitted, 2014. (Cité pages 5 et 81.)

J. Lesca, M. Minoux, and P. Perny. Compact versus Noncompact LP Formulations for minimizing Convex Choquet Integrals. *Discrete Applied Mathematics*, (161) :184–199, 2013. (Cité page 30.)

B. Li, D. Krushinsky, H. A. Reijers, and T. Van Woensel. The share-a-ride problem : People and parcels sharing taxis. *European Journal of Operational Research*, 238(1) :31–40, 2014. (Cité page 43.)

K. A. Lindsey, A. L. Erera, and M. W. Savelsbergh. A pickup and delivery problem using crossdocks and truckload lane rates. *EURO Journal on Transportation and Logistics*, 2(1-2) :5–27, 2013. (Cité page 77.)

S. Mancini, J. Gonzalez-Feliu, and T. Crainic. Planning and optimization methods for advanced urban logistics systems at tactical level. In J. Gonzalez-Feliu, F. Semet, and J.-L. Routhier, editors, *Sustainable Urban Logistics : Concepts, Methods and Information Systems*, EcoProduction, pages 145–164. Springer Berlin Heidelberg, 2014. (Cité page 43.)

E. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2) :236–245, 1984. (Cité page 19.)

R. Masson, F. Lehuédé, and O. Péton. A tabu search algorithm for the dial-a-ride problem with transfers. In *International Conference on Industrial Engineering and System Management*, Metz, France, 2011. (Cité page 3.)

R. Masson, F. Lehuédé, and O. Péton. Simple temporal problems in route scheduling for the dial–a–ride problem with transfers. In N. Beldiceanu, N. Jussien, and E. Pinson, editors, *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems*, volume 7298 of *Lecture Notes in Computer Science*, pages 275–291. Springer Berlin, Heidelberg, 2012. (Cité pages 3 et 64.)

R. Masson, F. Lehuédé, and O. Péton. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3) :344–355, 2013a. (Cité pages 3, 46 et 52.)

R. Masson, F. Lehuédé, and O. Péton. Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, 41(3) :211–215, 2013b. (Cité pages 3, 46, 58 et 59.)

R. Masson, F. Lehuédé, and O. Péton. The dial–a–ride problem with transfers. *Computers & Operations Research*, 41 : 12–23, 2014a. (Cité pages 3, 46, 52 et 64.)

R. Masson, S. Ropke, F. Lehuédé, and O. Péton. A branch-and-cut-and-price approach for the pickup and delivery problem with shuttle routes. *European Journal of Operational Research*, 236(3) :849–862, 2014b. (Cité page 3.)

R. Masson, A. Trentini, F. Lehuédé, N. Malhéné, O. Péton, and H. Tlahig. Optimization of a city logistics transportation system with mixed passengers and goods. *EURO Journal on Transportation and Logistics*, pages 1–29, 2015. doi : 10.1007/s13676-015-0085-5. (Cité pages 4 et 68.)

B. Mayag. *Interactions entre critères en Aide à la Décision : Élaboration d''une démarche constructive prenant en compte les interactions entre critères en Aide Multicritère à la Décision*. Editions universitaires européennes, 2010. (Cité pages 12 et 17.)

F. Meisel and H. Kopfer. Synchronized routing of active and passive means of transport. *OR Spectrum*, 36(2) :297–322, 2014. (Cité page 44.)

P. Meyer. *Contributions au processus d'Aide Multicritère à la Décision : Méthodes, Outils et Applications*. Habilitation à diriger les recherches, Université Paris Dauphine, 2013. (Cité page 12.)

P. Meyer and S. Bigaret. Diviz : A software for modeling, processing and sharing algorithmic workflows in MCDA. *Intelligent decision technologies*, 6 :283 – 296, 2012. (Cité page 16.)

J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, and G. Vitry. Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. *Computers & Operations Research*, 41 (0) :196 – 207, 2014. (Cité page 71.)

S. Mitrović-Minić and G. Laporte. The pickup and delivery problem with time windows and transshipment. *INFOR*, 44(3) :217–228, 2006. (Cité pages 46 et 55.)

L. Moccia, J.-F. Cordeau, G. Laporte, S. Ropke, and M. P. Valentini. Modeling and solving a multimodal transportation problem with flexible-time and scheduled services. *Networks*, 57(1) :53–68, 2011. (Cité pages 75, 77 et 83.)

J. Morana, J. Gonzalez-Feliu, and F. Semet. Urban consolidation and logistics pooling. In J. Gonzalez-Feliu, F. Semet, and J.-L. Routhier, editors, *Sustainable Urban Logistics : Concepts, Methods and Information Systems*, EcoProduction, pages 187–210. Springer Berlin Heidelberg, 2014. (Cité page 43.)

I. Murthy and S. Her. Solving min-max shortest-path problems on a network. *Naval Research Logistics*, 39 :669–683, 1992. (Cité page 20.)

J. Paquette, J.-F. Cordeau, and G. Laporte. Quality of service in dial-a-ride operations. *Computers & Industrial Engineering*, 56(4) :1721–1734, 2009. (Cité page 30.)

J. Paquette, J.-F. Cordeau, G. Laporte, and M. Pascoal. Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B : Methodological*, 52 :1–16, 2013. (Cité page 9.)

S. Parragh. *Ambulance routing problems with rich constraints and multiple objectives*. PhD thesis, University of Vienna, Austria, 2009. (Cité page 31.)

S. N. Parragh, K. F. Doerner, R. F. Hartl, and X. Gandibleux. A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks*, 54(4) :227–242, 2009. (Cité page 9.)

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34 : 2403–2435, 2007. (Cité pages 33, 50 et 51.)

PREDIT 3 - Groupe 5. Projet logistique mutualisée et durable. rapport final, 2009. (Cité page 79.)

E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54(4) :190–204, 2009a. (Cité pages 33 et 51.)

E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54(4) :190–204, 2009b. (Cité page 51.)

Y. Qu and J. F. Bard. A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research*, 39(10) :2439–2456, 2012. (Cité page 46.)

A. Rais, F. Alvelos, and M. Carvalho. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235(3) :530–539, 2014. (Cité page 66.)

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4) :455–472, 2006a. (Cité pages 33, 48, 51, 52 et 72.)

S. Ropke and D. Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171 :750–775, 2006b. (Cité pages 33, 50 et 51.)

B. Roy. Classement et choix en présence de points de vue multiples (la méthode ELECTRE). *R.I.R.O.*, 2 :57–75, 1968. (Cité page 10.)

B. Roy. *Méthodologie multicritère d'aide à la décision*. Economica, 1985. (Cité page 10.)

M. A. Salazar-Aguilar, A. Langevin, and G. Laporte. Synchronized arc routing for snow plowing operations. *Computers & Operations Research*, 39(7) :1432–1440, 2012. (Cité page 44.)

M. A. Salazar-Aguilar, A. Langevin, and G. Laporte. The synchronized arc and node routing problem : application to road marking. *Computers & Operations Research*, 40(7) :1708–1715, 2013. (Cité page 44.)

F. A. Santos, G. R. Mateus, and A. S. da Cunha. The Pickup and Delivery Problem with Cross-Docking. *Computers & Operations Research*, 40(4) :1085–1093, 2013. (Cité page 43.)

G. Sauvanet and E. Néron. Search for the best compromise solution on multiobjective shortest path problem. *Electronic Notes in Discrete Mathematics*, 36(0) :615 – 622, 2010. {ISCO} 2010 - International Symposium on Combinatorial Optimization. (Cité page 38.)

M. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4(1) :285–305, 1985. (Cité page 59.)

M. W. Savelsbergh. The vehicle routing problem with time windows : Minimizing route duration. *ORSA journal on computing*, 4(2) :146–154, 1992. (Cité page 71.)

J. S. Shang and C. K. Cuff. Multicriteria pickup and delivery problem with transfer opportunity. *Computers & Industrial Engineering*, 30(4) :631–645, 1996. (Cité page 46.)

L. Shapley. A value for n-person games. In H. Kuhn and A. Tucker, editors, *Contributions to the Theory of Games, Vol. II*, volume 28 of *Annals of Mathematics Studies*, pages 307–317. Princeton University Press, 1953. (Cité page 22.)

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming - CP98*, pages 417–431. Springer-Verlag, 1998. (Cité pages 33 et 51.)

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011 : An adaptive large neighborhood search algorithm. In *9th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492, 2012. (Cité page 50.)

M. Spada, M. Bierlaire, and T. Liebling. Decision-aiding methodology for the school bus routing and scheduling problem. *Transportation Science*, 39(4) :477–490, 2005. (Cité page 32.)

R. Spliet and G. Desaulnier. The discrete time window assignment vehicle routing problem. *Les cahiers du GERAD*, (G-2012-81), 2012. (Cité page 71.)

R. Spliet and A. F. Gabor. The time window assignment vehicle routing problem. *Transportation Science*, to appear, 2015. doi : 10.1287/trsc.2013.0510. (Cité page 71.)

M. SteadieSeifi, N. Dellaert, W. Nuijten, T. V. Woensel, and R. Raoufi. Multimodal freight transportation planning : A literature review. *European Journal of Operational Research*, 233(1) :1 – 15, 2014. (Cité pages 43 et 75.)

Z. Tarapata. Selected multicriteria shortest path problems : An analysis of complexity, models and adaptation of standard algorithms. *International Journal of Applied Mathematics and Computer Science*, 17(2) :269–287, 2007. (Cité page 17.)

R. E. Tarjan. Shortest paths. Technical report, AT&T Bell Laboratories, Murrray Hill, NJ, 1981. (Cité page 65.)

M. Timonin. Maximization of the choquet integral over a convex set and its application to resource allocation problems. *Annals of Operations Research*, 196(1) :543–579, 2012. ISSN 0254-5330. (Cité page 30.)

A. Trentini. *Proposition d'un système de transport urbain mixte. Application dans le cadre de la ville moyenne de La Rochelle.* PhD thesis, Ecole nationale supérieure des mines de Paris, 2012. (Cité pages 43 et 67.)

A. Trentini, R. Masson, F. Lehuédé, N. Malhéné, O. Péton, and H. Tlahig. A shared "passengers & goods" city logistics system. In *Proceedings of the 4th International Conference on Information Systems, Logistics and Supply Chain (ILS 2012)*, 2012. (Cité pages 4 et 68.)

P. Vallin and D. Vanderpooten. *Aide à la décision : une approche par les cas*. Ellipses, 2000. (Cité page 39.)

J. Van Belle, P. Valckenaers, and D. Cattrysse. Cross-docking : State of the art. *Omega*, 40(6) :827–846, 2012. (Cité page 43.)

D. Vanderpooten. The interactive approach in mcda : A technical framework and some basic conceptions. *Mathematical and Computer Modelling*, 12(10 – 11) :1213 – 1220, 1989. (Cité page 39.)

D. Vanderpooten and P. Vincke. Description and analysis of some representative interactive multicriteria procedures. *Mathematical and Computer Modelling*, 12(10 – 11) :1221 – 1238, 1989. (Cité page 9.)

Waze Mobile. Waze, 2014. URL `https://www.waze.com/`. (Cité page 39.)

M. Wen, J. Larsen, J. Clausen, J.-F. Cordeau, and G. Laporte. Vehicle routing with cross-docking. *Journal of the Operational Research Society*, 60(12) :1708–1718, 2009. (Cité page 43.)

Titre Problèmes de Tournées de Véhicules avec Synchronisation et Optimisation Multicritère avec l'Intégrale de Choquet

Résumé Ce manuscrit présente un ensemble de contributions réalisées depuis ma thèse, principalement à l'école des Mines de Nantes et au laboratoire IRCCyN. Ces travaux peuvent être regroupés en trois problématiques.
Une première partie concerne l'intégration d'un modèle multicritère basé sur l'intégrale de Choquet dans des algorithmes d'optimisation combinatoire. Je présenterai en premier lieu les motivations inhérentes à cette intégration. Les contributions portent d'une part sur la recherche de chemins Choquet–optimaux dans les graphes, qui a fait l'objet de la thèse de Hugo Fouchal, et d'autre part sur l'intégration du modèle dans une métaheuristique pour la résolution d'un problème de transport de personnes.
La seconde et principale partie de mon HDR propose une vue unifiée sur plusieurs contributions en tournées de véhicules avec synchronisation. Sous la dénomination synchronisation sont regroupées les caractéristiques qui créent une interdépendance entre tournées. Dans ce cadre, l'exploration de l'espace de recherche et la vérification des contraintes temporelles offrent des sujets de recherche tout à fait intéressants, à la croisée entre optimisation de tournées et ordonnancement. Ce chapitre présente la synthèse de plusieurs travaux, fruits de trois thèses (Renaud Masson, Philippe Grangier, Axel Grimault), dont deux sont toujours en cours, et de multiples collaborations.
La dernière partie porte sur la conception de réseaux de transport. Je présenterai une synthèse de travaux récents ou en cours qui abordent plusieurs problèmes d'industriels avec, à chaque fois, deux caractéristiques relativement originales. En premier lieu le transport est sous-traité, ce qui induit des modèles de coûts et des contraintes différents de ceux rencontrés dans la littérature. Deuxièmement, nos travaux interviennent dans un contexte collaboratif : un ensemble de petites et moyennes entreprises se regroupent pour consolider leurs flux et réduire ainsi les coûts de transport. Ces travaux font notamment l'objet de la thèse en cours de Juliette Medina.