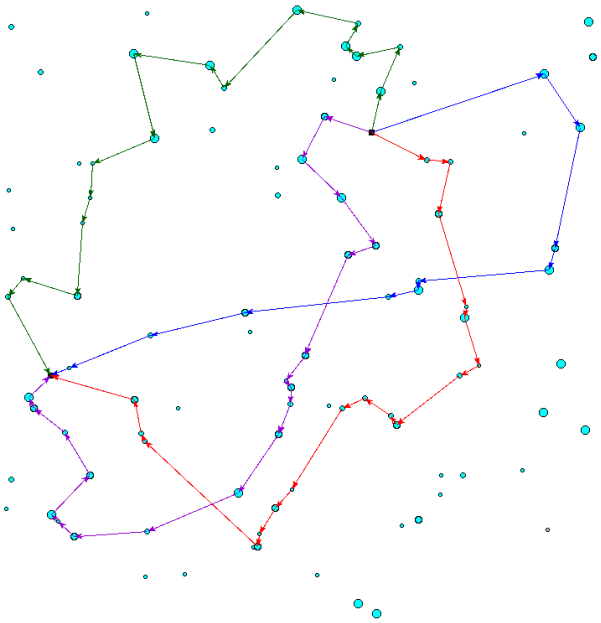
Par **Rym Nesrine GUIBADJ**

Problèmes de tournées de véhicules et application industrielle pour la réduction de l'empreinte écologique

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



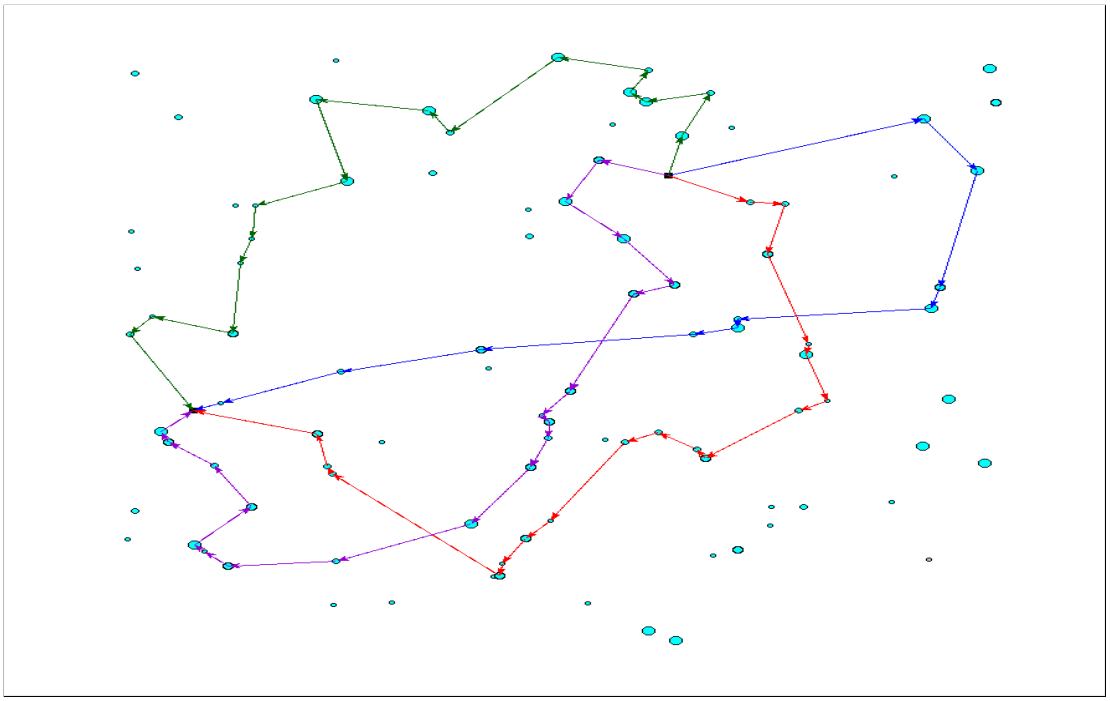
Soutenue le 16 avril 2013
Spécialité : Technologies de l'Information et des Systèmes

D2074

par Rym Nesrine GUIBADJ

Problèmes de tournées de véhicules et application industrielle pour la réduction de l'empreinte écologique

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le : 16 avril 2013
Spécialité : Technologies de l'Information et des Systèmes

Problèmes de tournées de véhicules et application industrielle pour la réduction de l'empreinte écologique

Thèse soutenue le 16 avril 2013 devant le jury composé de :

Gilles GONCALVES	Professeur des Universités (ARTOIS)	Rapporteur
Nacima LABADIE	Maître de Conférences, HDR (UTT)	Rapporteur
Jacques CARLIER	Professeur des Universités (UTC)	Examinateur
Philippe BOTTE	Chef de projet (VEOLIATRASNDEV)	Examinateur
Aziz MOUKRIM	Professeur des Universités (UTC)	Directeur

Remerciements

Je souhaite adresser mes plus sincères remerciements aux personnes qui ont cru en moi et qui m'ont permis d'arriver au bout de cette thèse.

En premier lieu, je tiens à exprimer mes plus vifs remerciements à Aziz Moukrim qui fut pour moi un directeur de thèse attentif et disponible. Sa compétence, sa rigueur scientifique et sa clairvoyance m'ont beaucoup appris. Ils ont été et resteront des moteurs de mon travail de chercheur.

Je voudrais également remercier M. Phillipe Botte de m'avoir donné l'opportunité de travailler sur un projet de recherche et de développement conséquent et ambitieux.

Je remercie cordialement M. Gilles Goncalves et Mme. Nacima Labadie d'avoir accepté d'être rapporteurs de cette thèse ainsi que M. Jacques Carlier d'avoir fait l'honneur d'examiner mon travail.

J'adresse mes plus profondes reconnaissances à tous les membres de l'équipe MERCUR et l'équipe HEUDIASYC pour tous les échanges techniques, scientifiques et pour leur sympathie et leur accueil chaleureux pendant ces trois ans de thèse.

Enfin, je tiens à remercier mes amis et mes proches en particulier mon conjoint pour son soutien sans faille et ses encouragements. Un grand merci du fond du coeur à ma mère qui m'a toujours soutenue inconditionnellement dans mon parcours universitaire, comme dans la vie.

Publications

Revue internationale

[1] Duc-Cuong Dang, **Rym Nesrine Guibadj** and Aziz Moukrim. An effective PSO-inspired algorithm for the team orienteering problem. European Journal of Operational Research, accepted, 2013.

Conférences internationales avec comité de lecture

[2] D-C. Dang, **R-N. Guibadj** and A. Moukrim. A PSO-based memetic algorithm for the team orienteering problem. EvoApplications 2011 (Turin, Italy). Lecture Notes In Computer Science, 2011, vol. 6625, p. 471-480.

Conférences nationales

[3] **R-N. Guibadj**, D-C. Dang et A. Moukrim. Optimisation par essaim particulaire pour le problème de m-tournées sélectives. ROADEF 2011 (Saint-Etienne, France).

[4] **R-N. Guibadj** et A. Moukrim. Un algorithme mémétique pour résoudre le problème de m-tournées sélectives avec fenêtres de temps. ROADEF 2013 (Troyes, France).

En préparation

[5] **R-N. Guibadj** and A. Moukrim. Memetic algorithm for the Team Orienteering Problem with Time Windows.

[6] **R-N. Guibadj**, S. Afifi and A. Moukrim. New lower bounds and exact algorithm for the Vehicle Routing Problem with Time Windows.

Table des matières

Liste des algorithmes

Liste des tableaux

Liste des figures

Introduction

Les engagements pris par de nombreux pays pour réduire la consommation d'énergies et les émissions de gaz à effet de serre sont ambitieux. La France s'est engagée à réduire d'un facteur 4 ses émissions pour 2050, après une réduction de 20% en 2020. Les villes, telles que Paris, ont décidé d'aller plus vite, soit 30% en 2020. Or les pouvoirs publics, et plus particulièrement les collectivités territoriales, n'ont pas les outils, notamment réglementaires pour obliger les entreprises et les individus à changer totalement d'habitudes pour atteindre ces objectifs. Dans ce contexte de crise, un processus d'adaptation permanent des comportements doit nécessairement être initié. Désormais, économiser et recycler les matières premières, améliorer les réseaux et modes de transport, améliorer les performances énergétiques des bâtiments doivent devenir des priorités. Cette période de transition énergétique, succédant à des énergies fossiles abondantes et bon marché, doit permettre l'émergence de nouveaux modèles et amener les populations à adopter de nouveaux comportements porteurs d'avenir.

Grâce au concours de ses nombreux partenaires, les travaux d'ingénierie de MERCUR, direction industrielle et service de VeoliaTransdev, sont orientés vers la transition et la substitution énergétique. Différents outils sont en cours de développement, en partenariat avec des responsables territoriaux. Une réflexion a été récemment engagée sur le thème de la "Mobilité Multimodale Intelligente". Un appel à manifestation d'intérêt a été exprimé par l'Agence de l'Environnement et de la Maîtrise de l'Énergie (ADEME). Cet appel a pour objectif de promouvoir la réalisation de démonstrateurs et d'expérimentations de nouvelles solutions de mobilité pour les personnes et les marchandises adaptées à différents types de territoires. Pour répondre aux problématiques posées, Mercur a réalisé plusieurs études sur les systèmes de mobilité actuels. Les nouvelles technologies de transport, plus économiques, moins polluantes, arrivent avec de nouveaux modes opératoires. Ces dernières rendent la gestion des ressources de transport de plus en plus complexe. L'intérêt pour les problèmes de logistique du transport est grandissant dans la vie actuelle puisqu'ils peuvent s'appliquer à de nombreuses activités comme la distribution de courrier, la livraison des produits, le traitement du réseau routier ou le ramassage des ordures.

Tous ces éléments, nous ont incités à étudier les problèmes de tournées. Sous sa forme la plus simple, le problème de tournées consiste à minimiser la distance totale parcourue par une flotte de véhicules homogènes afin d'assurer la livraison d'un nombre fixe de clients tout en respectant les contraintes de capacités des véhicules. Cependant, en tenant compte de toutes les intrications des cas de distribution

réels, le problème se complexifie rapidement. L'ajout de composantes telles que les véhicules hétérogènes, les fenêtres de temps ou une longueur maximale des routes rendent la résolution d'autant plus difficile. L'étude d'un problème de distribution réel n'est donc pas triviale. Elle demande une attention particulière afin de modéliser les particularités opérationnelles rencontrées.

Nous utilisons des approches de résolution basées sur une *méthode d'extraction*. Dans cette approche, la résolution du problème de base est décomposée en deux phases. On commence par l'extraction de sous problèmes que l'on résout à l'optimum puis on déduit une solution pour le problème le plus général du départ. Ceci permet d'utiliser différentes techniques d'optimisation dans les deux phases. La méthode a été utilisée dans la littérature pour différents problèmes sous différentes dénominations comme le *découpage optimal* (optimal split) pour les problèmes de tournées (VRP/ARP) [87, 111], l'*orientation d'arêtes* (edge orienting) pour la coloration de graphe (GCP) [61] ou l'*extraction de sous graphe* (subgraph extraction) [5] pour le problème de la clique maximum (MCP). Dans cette thèse nous proposons de nouvelles approches pour le problème de tournées sélectives. La thèse s'organise de la façon suivante.

Le premier chapitre décrit l'environnement industriel dans lequel s'inscrit notre travail. Nous présentons les concepts développés par MERCUR pour la production des scénarios d'évolution afin de se projeter dans le temps et montrer les changements possibles de l'empreinte écologique. Différents problèmes relevant de l'optimisation combinatoire sont identifiés. Le choix du sujet à traiter a porté essentiellement sur le périmètre de la mobilité. Il s'agit de planifier sur un territoire donné et dans un horizon restreint, le déplacement d'un groupe d'individus et l'acheminement final des marchandises. Cette problématique intègre des exigences associant performance, qualité de service, respect de l'environnement et qualité des conditions de travail qui s'expriment par un objectif complexe et de nombreuses contraintes.

Nous présentons le domaine de l'étude dans le second chapitre. L'objectif de cette partie est d'introduire les notions fondamentales de la recherche opérationnelle utiles pour la compréhension des travaux présentés dans la suite de la thèse. Une brève introduction est donnée pour la recherche opérationnelle et l'optimisation combinatoire. Ensuite, nous rappelons la théorie de la complexité des algorithmes et celle de la complexité des problèmes. Différentes approches de résolution sont présentées : méthodes heuristiques, métaheuristiques et méthodes exactes. A la fin de ce chapitre, les problèmes de tournées de véhicules abordés dans la thèse sont brièvement décrits.

Les troisème et quatrième chapitres s'intéressent à des problèmes particuliers de tournées avec profits, le *Team Orienteering Problem* (TOP) et le *Team Orienteering Problem With Time Windows* (TOPTW). Le TOP consiste à choisir et organiser les visites d'un ensemble de clients potentiels en utilisant une flotte de véhicules tous identiques, en respectant une contrainte limitant la longueur et/ou la durée des tournées. L'objectif du TOP est de maximiser la somme des profits collectés par le service des clients visités. Dans le TOPTW, de nouvelles contraintes temporelles sont ajoutées : chaque client a une durée de service et une fenêtre de temps définie par une

date de service au plus tôt et une date de service au plus tard. Le service d'un client doit impérativement être inclus dans cette fenêtre de temps. Plus formellement, ces problèmes sont représentés par un graphe complet non-orienté où les sommets sont les clients et les arêtes représentent les distances entre ces clients. Nous considérons que les distances sont symétriques et euclidiennes. Nous représentons une solution du TOP/TOPTW par une permutation de tous les clients, indiquant l'ordre dans lequel les visiter. Sur ce principe, les solutions associées à une telle représentation doivent être telles que les clients d'une même tournée soient adjacents dans la permutation et apparaissent dans l'ordre induit par cette permutation. Dans les travaux de Bouly et al. [25], il a été montré qu'il suffit de considérer les *tournées saturées* lors du découpage d'une permutation en plusieurs tournées. Une tournée est dite *saturée* si elle contient le plus grand nombre de clients possible sans pour autant dépasser la contrainte sur la longueur maximale.

Nous avons exploité la notion de tournées saturées pour proposer un nouveau découpage optimal plus efficace. Une mise en correspondance avec les modèles de graphe d'intervalle a été établie et le problème a été ramené à un problème de sac à dos particulier avec conflits. Suite à ces différentes opérations, la résolution de ce nouveau problème, plus restreint, peut s'effectuer efficacement par la programmation dynamique. Le détail de cette approche est décrit dans le troisième chapitre. Ce chapitre présente également un schéma général basé sur l'optimisation par essaims particulaires (PSO) pour résoudre le TOP. Une adaptation fine de différents paramètres permet d'obtenir toutes les meilleures solutions de la littérature à l'exception d'une instance avec un écart d'une seule unité de profit. Afin de promouvoir les développements algorithmiques sur le TOP, nous avons proposé un nouveau benchmark avec des instances plus larges contenant jusqu'à 400 clients. En se basant sur la même méthode d'extraction développée, dans le quatrième chapitre, nous procédons à l'élaboration d'un algorithme mémétique (MA) pour le TOPTW. L'efficacité de la méthode proposée est démontrée sur les instances d'un benchmark standard avec plusieurs améliorations strictes par rapport aux méthodes de résolution de la littérature.

Le cinquième chapitre s'intéresse au Problème de Tournées de Véhicules avec Fenêtres de Temps (VRPTW). Nous procédons à une étude de bornes inférieures sur le nombre de véhicules requis pour servir l'ensemble des clients. Notre méthode est une adaptation du Raisonnement Énergétique. Cet algorithme, initialement développé pour les problèmes d'ordonnancement, permet la détection des infaisabilités et l'ajustement des intervalles de temps pendant lesquels l'exécution des tâches est autorisée. Nous proposons différentes procédures visant à définir les temps de trajets minimaux succédant au service de chaque client de sorte à intégrer ces temps à l'expression d'un problème d'ordonnancement sous contrainte de ressources (le CuSP en l'occurrence). Nous étendons le raisonnement énergétique, en utilisant le problème de bin-packing avec conflits, pour obtenir des bornes inférieures plus serrées. En se comparant aux résultats de la littérature, nous avons pu prouver l'optimalité du nombre de véhicules trouvé grâce aux prétraitements proposés. Ces résultats avantageux doivent être poursuivis afin d'intégrer ce nouveau concept dans un schéma de résolution exacte.

Nous terminons par une conclusion qui fait la synthèse de l'ensemble de nos contributions et décrit quelques perspectives de recherche.

Contexte et expression industrielle du problème

1.1 Résumé

Les travaux présentés dans ce chapitre sont relatifs à la formulation et la résolution de problèmes d'optimisation exprimés par Mercur, direction industrielle et service de VeoliaTransdev. Dans le cadre de ses activités de développement de solutions aux problématiques liées aux enjeux climatiques et environnementaux, Mercur accompagne les entreprises, les collectivités et les particuliers dans leurs projets d'économie d'énergie et de réduction d'émissions de gaz à effet de serre. Dans un premier temps, une mise en contexte est effectuée. Nous présentons la nature du problème et montrons en quoi il constitue un sujet de recherche. L'objectif est de cadrer notre travail pour visualiser le périmètre d'étude retenu et comprendre les principales caractéristiques. Ensuite, nous nous attardons sur les problématiques liées au transport des personnes et marchandises. Les explications concernant les différents objectifs et les différentes contraintes sont données d'un point de vue industriel.

1.2 Contexte

La thèse s'inscrit dans le cadre d'une collaboration entre la société MERCUR (direction industrielle et service de VeoliaTransdev) et le laboratoire Heudiasyc de l'Université de Technologie de Compiègne. L'objectif est d'apporter des éléments de solution pour l'amélioration de l'empreinte écologique (économie d'énergie et réduction des émissions de gaz à effet de serre). Les travaux de la société s'intéressent en particulier à la gestion du parcours de vie d'un foyer pour assurer son passage à un mode de vie meilleur, adapté à ses besoins, et moins énergivore.

Le changement des habitudes d'un foyer nécessite des prises de décision à plusieurs niveaux. Les techniques et les outils de la recherche opérationnelle peuvent fournir une aide dans ces prises de décision. Nos travaux de recherche visent à appliquer des techniques de recherche opérationnelle dans le contexte d'amélioration de l'empreinte écologique. Dans un contexte de réduction des dépenses énergétiques, nos travaux montrent la faisabilité et la pertinence d'un outil d'aide à la décision

pouvant être mis en œuvre par des entreprises, des collectivités ou directement par les particuliers. Cet outil permet d'évaluer puis de visualiser différents scénarios d'investissement et de ventiler au mieux un budget limité.

Suite à plusieurs réunions exploratoires, trois périmètres de travail ont été identifiés : mobilité, habitat et consommation. Pour chaque périmètre, une collecte de données est effectuée afin de constituer un inventaire aussi exhaustif que possible des produits consommés, des services sollicités et des pratiques mises en œuvre. Ceci entraîne une diversification au niveau des produits et services collectés autant qu'alternatifs. Il est nécessaire donc d'avoir recours à des travaux de recherche afin de modéliser ces différents éléments au sein d'un même système.

Sur la base de ces données, nous cherchons à mettre en œuvre des algorithmes pertinents pour la recherche et la génération automatisée de scénarios, ou plans d'amélioration, intégrant les différentes contraintes de l'utilisateur (financières, temporelles) et l'aidant dans ses prises de décision. De plus, cette modélisation doit inclure les liens existants entre les composants afin d'aboutir à la conception d'un système ayant la capacité d'exploiter cette base de connaissances et fournissant des solutions, équivalentes à celles que fourniraient les experts des domaines considérés. Par exemple, des décisions relatives à la mobilité peuvent libérer des ressources qui seront exploiter dans l'habitat.

Le problème de base considéré s'apparente à un problème d'ordonnancement avec contraintes de ressources. En effet, Le problème peut être vu comme un projet avec un ensemble fini d'activités dont le but est de répondre à un besoin précis, dans des délais fixés et dans la limite de l'enveloppe budgétaire allouée. Une activité modélise un besoin temporaire ou permanent dans un horizon de planification fixe. Elle nécessite une quantité donnée de chacune des ressources tout au long de son exécution (énergie, argent, etc.). Cette quantité dépend du moyen utilisé pour effectuer l'activité.

Pour des raisons de temps, et étant donné l'ampleur des travaux, nous avons décidé de se concentrer sur le premier thème, lié au transport. L'objectif est de développer et tester des technologies, systèmes ou services susceptibles de rendre le système de mobilité (voyageurs et marchandises) plus performant : plus économique, plus efficace du point de vue énergétique et environnemental et exploitant mieux les différents modes. Afin de mieux comprendre les enjeux de la problématique posée, nous présentons dans ce qui suit les deux champs d'applications étudiés.

1.3 Problématique de la mobilité

Le secteur des transports se développe rapidement. Nous en apprécions les avantages : rapidité et accessibilité géographique mais cette médaille a son revers : bruit, congestion, émissions polluantes et dioxyde de carbone (CO_2), principal gaz à effet de serre responsable du réchauffement climatique. Des progrès technologiques sont attendus sur les motorisations automobiles des véhicules thermiques, électriques et hybrides, et sur les énergies utilisées. A eux seuls, ces progrès ne seront cependant

pas suffisants pour répondre aux objectifs de lutte contre le changement climatique, ainsi qu'aux enjeux sociaux et territoriaux d'une mobilité durable pour tous. Il faut donc considérer dès à présent de nouvelles mobilités, de nouveaux usages pour l'automobile, tout en optimisant le système complet et en exploitant le potentiel offert par les technologies de l'information.

Mercur a répondu à un appel à manifestations d'intérêt lancé par L'ADEME (Agence de l'Environnement et de la Maîtrise de l'Énergie). Cet appel est le premier d'une série. Il se focalise sur les déplacements quotidiens des personnes et sur l'acheminement final des marchandises. Il s'adresse à de nombreux acteurs (constructeurs, fournisseurs d'énergie, opérateurs de transports, collectivités, entreprises et laboratoires de recherche) pour développer, dans des territoires d'expérimentation, des démonstrateurs destinés à la création d'un système de mobilité plus simple et plus rapide pour l'usager et l'entreprise. Autrement dit, un système permettant d'optimiser et de mieux exploiter les modes de transport actuels, publics et privés, collectifs et individuels.

1.3.1 Transport des personnes

Le secteur des transports représente un enjeu stratégique de premier plan sur le long terme en matière de maîtrise de ses émissions de gaz à effet de serre. Du fait des problématiques liées à l'usage de la voiture comme moyen de transport principal (circulation croissante dans les agglomérations, pics de pollution, etc.), l'utilisation de modes de transport alternatifs devient un enjeu important. Afin de résorber les problèmes existants engendrés de par l'utilisation massive de la voiture particulière, plusieurs alternatives existent. On peut citer :

Le transport en commun : ou *transport collectif* met en œuvre des véhicules adaptés à l'accueil simultané de plusieurs personnes. Cependant, certains lieux sont parfois mal desservis et difficiles à rejoindre.

L'autopratge : il s'agit plus d'un partage de parcs de véhicules par les membres du service. L'utilisateur d'un service d'autopartage dispose d'une voiture qu'il ne finance que pour la durée de son besoin. Le reste du temps, la voiture est utilisée par d'autres membres.

le covoiturage : cette deuxième forme de partage de voiture particulière, se réfère à l'utilisation d'une voiture par un conducteur non professionnel et un ou plusieurs passagers dans le but d'effectuer un trajet commun.

Compte tenu de la montée en puissance du concept de la voiture partagée, il est déterminant de pouvoir évaluer et proposer des informations de plus en plus élaborées pour aider les individus à choisir le ou les modes de transport les mieux appropriés (voiture, transport en commun, covoiturage, vélo, combinaison de plusieurs modes, etc.) par rapport à leur besoin en déplacement. La demande de déplacement est dérivée de la réalisation des activités qui peut être partagée entre plusieurs individus. C'est pourquoi de nouveaux outils sont nécessaires pour analyser les interactions entre les systèmes de transport et les activités des usagers. En effet,

lorsqu'un individu prend une décision en termes de déplacement, il ne considère pas qu'un seul trajet, mais l'ensemble des trajets combinés. Nous cherchons à modéliser le choix des usagers en considérant le lien entre la mobilité et les activités à effectuer. L'objectif est de proposer un modèle permettant de simuler l'enchaînement des activités et des déplacements, afin d'aider les personnes à choisir le ou les modes de transport les plus appropriés en fonction de leurs besoins.

Expression industrielle du problème

Nous cherchons à bâtir un système de mobilité de préfiguration d'une société post-carbone sur un territoire donné. Pour cela, un ensemble de personnes "adhérents" envisagent de mutualiser leurs véhicules et d'optimiser leurs déplacements grâce à des outils d'optimisation adaptés. Par exemple, on peut imaginer deux familles ouvertes à utiliser plusieurs modes les uns à la suite des autres. Pour aller au travail, Monsieur et Madame Zimmermann covoiturent avec M. et Mme Becker en prenant une voiture autopartage jusqu'à la gare, où sont déposés Mme Zimmermann et M. Becker qui prennent le train jusqu'à Strasbourg puis les transports en commun. Mme Becker dépose ensuite M. Zimmermann à son lieu de travail puis se rend à son entreprise située dans une zone d'activité où se trouve un parking d'autopartage. D'un autre côté, lorsque M. Becker fait ses courses le weekend, il peut aller récupérer en même temps les enfants de sa voisine Mme Zimmermann de leur séance de sport.

D'une manière générale, un groupe de personnes disposant d'un ensemble de véhicules, décrivent les activités qui doivent être effectuées sur un horizon de temps donné (il s'agit par exemple de la période d'une semaine). Chaque activité possède une durée et une contrainte de ponctualité (plage horaire d'arrivée) qui lui sont propres. Le contexte pratique impose par ailleurs que certaines activités sont déjà affectées à un individu particulier à une date donnée. Ces activités ne doivent en aucun cas être remises en cause par les algorithmes de construction des tournées.

Les personnes se déplacent pour exercer leurs activités dans un réseau de transport multimodal où plusieurs alternatives de mobilités sont disponibles. La prise en compte de la multimodalité des réseaux de transport introduit un certain nombre de contraintes supplémentaires. Un chemin multimodal doit respecter un ensemble de contraintes sur les séquences d'utilisation des différents modes de transports. Par exemple prendre la voiture après avoir commencé en transport en commun est irréalisable pour une personne donnée. Cependant, l'existence de points de stationnement permet l'apparition ou la disparition d'un mode lors d'une tournée (laisser sa voiture dans un parc relais, continuer le trajet par train puis récupérer la voiture en fin de journée lors du trajet de retour). Ceci permet d'utiliser les modes privatifs d'une manière plus efficace en les combinant avec les autres modes de transport (transport commun, transport à la demande, etc.). Ce critère de cohérence de la disponibilité des modes est très important pour ne pas obtenir des solutions non réalistes.

1.3.2 Transport des marchandises

Dans l'économie actuelle, rare est le produit qui arrive à être consommé par son utilisateur final sans l'intervention du transport. Naturellement, ces déplacements entraînent des coûts financiers et environnementaux. Nous cherchons donc à améliorer la distribution en introduisant un outil d'aide à la décision pour les livraisons aux clients finaux de la chaîne logistique.

La spécialisation des livraisons s'accentue encore, conduisant à de nouveaux véhicules, de nouvelles énergies, et des objectifs de performances environnementales à atteindre. Le passage à des véhicules électriques impose d'adapter la chaîne logistique aux caractéristiques et contraintes de ces véhicules : respect de la durée de vie des batteries, maximisation de l'autonomie du véhicule.

Dans ce contexte, nous pouvons imaginer une ville où plusieurs entreprises et commerces locaux mutualisent leurs véhicules et optimisent les livraisons de leurs clients grâce à des outils d'optimisation des ressources (rétro planning des véhicules, mutualisation d'une tournée) avec des véhicules économes en énergie et en carbone. De son côté, lorsqu'il fait ses courses ou passe une commande, un client peut demander une livraison à domicile ou bien de pouvoir passer chercher en même temps sa commande et la commande de son voisin.

L'intégration d'un outil d'aide à la décision apporte de nombreux avantages au système de transport. Cela permet d'accroître la performance de l'entreprise sans pour autant augmenter le nombre de véhicules requis. De plus, la régularisation des heures de travail des chauffeurs est souvent perçue comme étant un facteur motivant. L'adoption d'un logiciel de routage diminue aussi la charge du travail requis pour accomplir les tâches reliées à la distribution et donc encourage une surveillance accrue et un contrôle plus serré sur l'ensemble des opérations.

Expression industrielle du problème

Nous cherchons à livrer un ensemble de clients répartis sur un territoire donné. Une interface de prise de rendez vous permet d'enregistrer les différentes commandes et de définir le créneau horaire pendant lequel un client peut être servi. Une commande peut être formulée à tout moment. Ces commandes étant très nombreuses et certaines étant surtout formulées tardivement, la planification ne peut être réalisée que sur un horizon relativement court et toutes les livraisons ne peuvent pas être planifiées dans cet horizon restreint.

Le choix des livraisons met en jeu un caractère sélectif important. Par exemple, il est nécessaire de s'assurer que la livraison des produits alimentaires périssables sont traités en priorité afin de fournir des aliments frais. La satisfaction des rendez vous clientèle est aussi hautement considérée.

L'objectif est de servir tous les clients, en minimisant le coût total de transport. Ce coût est relatif au nombre de véhicules utilisés et à la distance parcourue par chaque véhicule ainsi que l'empreinte écologique des déplacements. Les tournées produites doivent, naturellement, respecter les disponibilités des livreurs et l'hétérogénéité de la flotte (électrique ou thermique, capacité, possédant

ou pas une aire réfrigérée).

1.3.3 Positionnement du cas industriel

A partir des différentes données recueillies sur le terrain, nous avons remarqué que la problématique comporte plusieurs éléments des problèmes de tournées avec fenêtre de temps, et des problèmes de tournées sélectives. Nous identifions donc la maximisation des profits et la minimisation des coûts fixes de sortie des véhicules.

Les coûts du trajet sont donnés par quatre grandeurs physiques considérées comme représentatives de l'empreinte écologique : émission CO_2, énergie fossile, énergie renouvelable, budget. Conformément à la volonté exprimée par les intervenants industriels de quantifier les différents éléments d'évaluation sous la forme d'une expression homogène, nous optons pour une approche mono-critère agrégeant l'ensemble des coûts. Ainsi, ces valeurs sont agrégées pour fournir un seul coût d'usage d'un trajet.

1.4 Problématique de l'habitat

Le deuxième volet de nos travaux a porté sur le domaine de l'habitat. L'objectif est de définir des axes de réhabilitation permettant d'aller dans le sens d'une réduction des dépenses énergétiques pour les maisons existantes. Il s'agit de dégager des solutions pouvant être proposées aux propriétaires de maisons individuelles. L'outil informatique développé doit permettre d'aider à la prescription d'actions de réhabilitation. Cet outil a pour objet d'être un support de conviction pour le propriétaire de maison individuelle. Ces objectifs sont traduits aux travers d'un ensemble de besoins et de recommandations tels que :

- définir le logement existant (caractéristiques générales, date de construction, situation, etc.),
- réaliser un diagnostic du logement existant,
- proposer automatiquement une sélection des packs de réhabilitation adaptés au logement existant,
- établir un rapport de la solution de réhabilitation finale choisie mettant en avant les avantages d'un point de vue énergétique environnemental, économique et amélioration du confort.

Des voies d'amélioration, qu'on appelle aussi des composants alternatifs, sont extraites à partir d'une base de données. Cette base a été fournie par notre partenaire GEST'énergie (expert des études en énergies vertes et efficacité énergétique). Elle est basée sur la méthode des calculs des consommations conventionnelles dans les logements (3CL), qui est le procédé officiel fourni par l'ADEME pour les calculs de consommation des logements. Ceci permet d'intégrer des solutions techniques innovantes au sein d'un nouveau cœur de calcul. On intègre dans cette réflexion à la fois les travaux sur les équipements énergétiques (chauffage, production d'ECS ...) et les travaux améliorant les caractéristiques du bâti (ventilation, isolation, menuiseries,

solaire passif ...), soit toutes les opérations d'évolution et d'amélioration ayant un impact direct ou indirect sur la consommation d'énergie du logement. Ces solutions techniques sont réparties sur plusieurs postes :

- isolation des parois (murs, plafonds, planchers),
- système de chauffage et d'eau chaude sanitaire,
- solaire et climatisation,
- ventilation et climatisation,
- éclairage,
- appareils électriques.

A chaque brique correspond une fiche décrivant ses caractéristiques principales : performance énergétique, compatibilité croisée, avantages et contraintes des solutions (aspects techniques, économiques). La notion de scénario améliorant ou pack de solutions repose sur l'idée de considérer la rénovation d'une maison de manière globale. Il s'agit justement d'éviter une logique de rénovation poste à poste, relevant de la mise en œuvre de briques technologiques sans cohérence d'ensemble, pour au contraire proposer des packs pertinents et cohérents constitués d'un assemblage bien conçu des composants alternatifs générés. La combinaison des solutions est une étape purement logique. On commence par répartir les briques technologiques par poste d'application (les murs, les combles, le système de chauffage, etc.) de manière à ce qu'au sein d'un même poste, une seule amélioration puisse être réalisée à la fois. A ce moment là, réaliser un pack de rénovation revient à piocher (ou non) une brique dans chaque poste. Plusieurs possibilités étaient envisageables dans le choix des solutions de travaux d'économies d'énergie. Cela dépend de ce que l'on privilégie : l'investissement de départ, les économies en énergie réalisées, les économies financières réalisées ou le retour sur investissement. Tous ces éléments étant importants, il convient de faire un choix qui les prend tous en compte, par ordre de priorité.

Le problème qui se pose lorsqu'on cherche à optimiser la solution, est un problème de planification sous contraintes. La modélisation du problème consiste principalement en la mise en forme d'un critère. Ce critère doit être une représentation analytique du problème qui tient compte de toutes les contraintes. En effet, on constate que certaines solutions ne sont pas compatibles avec certains logements. Par exemple, les robinets thermostatiques ne peuvent s'appliquer que pour un certain type de chauffage. Il faut aussi tenir compte du fait que certaines solutions ne sont pas compatibles : on ne va pas proposer d'installer à la fois une pompe à chaleur géothermique et une autre aérothermique. Le choix de certaines solutions interdit donc le choix d'autres solutions incompatibles. Le logiciel doit proposer une solution à la fois adaptée aux besoins de l'utilisateur, qui respecte la liste des incompatibilités et qui est intéressante financièrement.

1.5 Conclusion

Nos travaux s'inscrivent dans le cadre de la conception de systèmes d'aide à la décision permettant de valoriser les économies d'énergie réalisées par l'individu. Cet

outil est conçu pour mobiliser et mettre en valeur des initiatives et compétences nouvelles. Il produit des scénarios d'évolution de la consommation d'un particulier afin de le projeter dans le temps en lui montrant les changements possibles et son empreinte future.

Une fois le cadre général a été défini, nos efforts se sont focalisés sur le transport des personnes et des marchandises. Les problèmes étudiés se présentent sous la forme de problèmes de tournées de véhicules dont la résolution suppose une prise en compte aussi exhaustive que possibles des contraintes liées au métier et de la complexité associée à sa résolution. Les problèmes de tournées de véhicules ont fait l'objet de nombreuses recherches en vue d'appréhender les différents aspects des applications réelles.

Afin de préserver le caractère confidentiel du projet, nous ne détaillons pas dans cette thèse les contributions industrielles en termes de modélisation, adaptation des algorithmes et expérimentations. Nous nous attachons par contre à discuter les différentes contributions académiques apportées.

Le chapitre suivant donne un aperçu de la littérature sur les problèmes de tournées de véhicules, au travers duquel apparaîtra la manière dont ont déjà été appréhendés certains éléments de la problématique étudiée, ainsi que ceux sur lesquels ont dû se concentrer nos travaux de recherche.

Domaine de l'étude

Ce chapitre décrit le contexte scientifique des travaux de cette thèse. Le paysage d'un problème d'optimisation est au cœur de nos travaux. Nous commençons par donner les principales définitions correspondantes. Puis nous présentons les outils mathématiques et les classes d'algorithmes permettant la résolution de ces problèmes. Enfin, nous présentons les problèmes de tournées en nous attardant davantage sur les variantes traitées dans la thèse. Nous concluons sur les méthodes actuelles qu'il est intéressant d'approfondir.

2.1 Problèmes d'optimisation combinatoire

Un problème d'optimisation consiste à trouver le meilleur élément (appelé optimum) parmi un ensemble d'éléments autorisés, en fonction d'un critère de comparaison [107]. Il est défini par une fonction mathématique (appelée fonction objectif) et un espace de recherche représentant l'ensemble des solutions. On parle d'un problème de maximisation quand la qualité est donnée par une fonction objectif à maximiser et d'un problème de minimisation quand la qualité est donnée par une fonction objectif à minimiser.

Un problème d'optimisation combinatoire est un problème d'optimisation dans lequel l'espace de recherche (noté Ω) est dénombrable [106]. Soit $f : \Omega \longrightarrow R$ la fonction objectif qui assigne à chaque solution discrète $s \in \Omega$ le nombre réel $f(s)$. Le but est de trouver s^* tel que : $s^* = argmax_{s \in \Omega} f(s)$. Une telle solution s^* s'appelle une solution optimale.

Dans ce contexte, nous définissons les termes d'*optimum global* et d'*optimum local* associés à un problème d'optimisation. Pour un problème de minimisation (resp. maximisation), un optimum global est une solution $s^* \in \Omega$ telle que : $\forall s \in \Omega, f(s^*) \leq f(s)$ (resp. $f(s^*) \geq f(s)$). Si l'espace de recherche est muni d'une relation de voisinage V. Un optimum local est alors défini comme une solution $s^* \in \Omega$ telle que : $\forall s \in V, f(s^*) \leq f(s)$ (resp. $f(s^*) \geq f(s)$).

2.2 Théorie de la complexité

La théorie de la complexité s'intéresse à l'étude formelle de la difficulté des problèmes en informatique. Elle est basée sur les travaux d'Edmonds [49] et de Cook

[37]. L'objectif de calculer la complexité d'un algorithme, est d'obtenir un ordre de grandeur du nombre d'opérations élémentaires nécessaires pour que l'algorithme fournisse la solution du problème à l'utilisateur. Ceci permet de comparer la performance des algorithmes indépendamment des caractéristiques de la machine ou du langage utilisé. Les travaux théoriques dans ce domaine ont permis d'identifier différentes classes de problèmes en fonction de la complexité de leur résolution [57]. Les classes de complexité ont été introduites pour les problèmes décisionnels, c'est-à-dire les problèmes posant une question dont la réponse est *oui* ou *non*. Un problème décisionnel peut appartenir à deux classes :

- La classe P (Polynomial time) : contient l'ensemble des problèmes pouvant être résolus, de manière exacte, par un algorithme de complexité polynomiale.
- La classe NP (Nondeterministic Polynomial time) : contient l'ensemble des problèmes dont on peut vérifier qu'une proposition donnée est bien une solution du problème avec un algorithme de complexité polynomiale.

La résolution d'un problème NP peut nécessiter l'examen d'un grand nombre (éventuellement exponentiel) de cas mais l'examen de chaque cas doit se faire en temps polynomial. On déduit que $P \subset NP$. La question est de savoir si $NP \subset P$. La différence entre P et NP n'a pas été prouvée, mais la conjecture est considérée comme hautement probable. Les problèmes les plus difficiles de NP définissent la classe des problèmes NP-Complet. La NP-complétude s'appuie sur la notion de réduction polynomiale [78]. Un problème est NP-Complet si n'importe quel autre problème de NP peut être transformé en ce problème par une procédure polynomiale. Ces problèmes constituent le *noyau dur* des problèmes NP, si bien que si l'on trouvait un algorithme polynomial permettant de résoudre un seul problème NP-Complet, on pourrait en déduire un autre pour tout problème NP.

Nous distinguons également dans la classe NP, la classe des problèmes NP-Difficile. Ce sont les problèmes d'optimisation combinatoire dont le problème de décision associé est NP-Complet.

2.3 Méthodes de résolution

Les méthodes de résolution de problèmes d'optimisation NP-Difficile sont classées en deux catégories : les méthodes exactes et les méthodes approchées.

2.3.1 Méthodes exactes

Les méthodes exactes sont conçues pour trouver un optimum du problème. Toutefois, leurs durées de calcul tendent à augmenter exponentiellement avec la taille des instances des problèmes NP-Difficile qu'elles essaient de résoudre. Nous présentons dans cette section deux méthodes exactes très connues et très utilisées : la programmation linéaire et l'exploration arborescente.

2.3.1.1 Programmation linéaire

Un problème de programmation linéaire (PL) est un problème d'optimisation où la fonction objectif et les contraintes sont linéaires. Un programme linéaire s'écrit de la façon suivante :

$$\begin{aligned} \min f(x) &= c^T x \\ \text{s.c. } Ax &\geq b \\ x &\in R^n \end{aligned}$$

- c vecteur de coût (dimension n),
- A matrice de contraintes (dimension $n * m$),
- b vecteur nommé partie droite (dimension n),

On parle de programme linéaire en nombres entiers (PLNE) lorsque les valeurs des variables doivent prendre des valeurs entières. On dispose pour ce type de problèmes des méthodes issues de la programmation mathématique permettant de les résoudre efficacement. L'algorithme du Simplexe est le plus souvent implémenté pour ces problèmes dans les environnements logiciels de développement comme ILOG Cplex.

La plupart des problèmes d'optimisation combinatoires peuvent être formulés sous forme d'un PLNE. La seule contrainte d'intégrité sur les variables rend la résolution généralement difficile. La relaxation continue consiste à supprimer les contraintes d'intégralité des variables du PLNE, de manière à obtenir un PL. Si la solution optimale trouvée par le PL est en nombres entiers, c'est également une solution optimale pour le problème discret. Toutefois, une relaxation peut produire des solutions dont le coût est très éloigné de l'optimum du problème original. Résoudre un PLNE nécessite donc l'application de méthodes dédiées, telles que les algorithmes de séparation et évaluation et les méthodes de coupes.

2.3.1.2 L'exploration arborescente

Les algorithmes d'exploration arborescente sont basés sur l'idée d'énumérer (complètement ou partiellement) les solutions possibles d'un problème combinatoire pour y retrouver la meilleure. Une telle énumération se traduit par l'exploration d'un arbre virtuel.

Les algorithmes basés sur une exploration arborescente utilisent des techniques de filtrage afin de réduire la taille de l'arbre effectivement exploré. Parmi les techniques d'exploration arborescente qui incluent des mécanismes de filtrage, on retrouve l'algorithme de séparation et évaluation (*branch and bound* [36]). La procédure de séparation divise le problème en un certain nombre de sous-problèmes. Le même principe de séparation est appliqué récursivement à chacun des sous-ensembles de solutions obtenus. Ainsi, en résolvant tous les sous-problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Les sous-ensembles de solutions ainsi construits peuvent être assimilés à des nœuds d'un arbre appelé arbre de recherche ou arbre de décision.

Une procédure d'évaluation est appliquée aux nœuds de l'arbre afin de réduire sa taille. A chaque itération de la procédure de séparation, une borne est calculée au

niveau de chaque nœud en résolvant une relaxation. Dans le cadre de problèmes de minimisation nous parlons de borne inférieure. Trois situations peuvent se présenter :

- La solution du problème relaxé est réalisable pour le problème non relaxé en ce nœud. Si elle est réalisable pour le problème initial alors une mise à jour de la meilleure solution connue est effectuée.
- La solution du problème n'est pas réalisable mais elle donne une borne inférieure qui excède le coût de la meilleure solution connue. Il est donc inutile de continuer l'exploration du nœud.
- La solution du problème n'est pas réalisable et la borne inférieure ne dépasse pas le coût de la meilleure solution connue. Alors la procédure de séparation doit être appliquée en ce nœud.

L'exploration de l'arbre de recherche peut être notamment en largeur, en profondeur ou en donnant la priorité au meilleur nœud.

L'utilisation de méthodes de coupes pour calculer de bonnes bornes au cours de la procédure d'évaluation donne lieu a l'algorithme de branchement et coupes (*branch-and-cut*) ([107], chapitre 3.3). L'idée de cette méthode est de simplifier la résolution d'un problème en relaxant certain de ses contraintes. Si la solution optimale du problème relaxé est une solution valide pour le problème original, alors elle est l'optimum recherché. Sinon, Il faut trouver un sous-ensemble de ces contraintes alors appelées coupes, puis les ajouter à la relaxation avant de la résoudre à nouveau. La procédure se poursuit jusqu'à ce que la solution obtenue soit réalisable pour le problème original, ou alors que les procédures d'identification de coupes n'arrivent plus à trouver de contraintes violées. Dans ce dernier cas, le coût de la dernière solution trouvée est une borne inférieure du coût optimal s'il s'agit d'un problème de minimisation, ou alors une borne supérieure dans le cas contraire.

2.3.2 Méthodes approchées

Lorsqu'il s'agit de problèmes de dimension importante, le recours aux méthodes approchées s'impose. Ces méthodes ont pour but de produire une solution réalisable, de bonne qualité mais pas forcément optimale, sans nécessiter des temps de calcul importants. Dans ce qui suit, nous présentons d'abord les heuristiques et les recherches locales puis nous détaillons des méthodes de résolution plus sophistiquées basées sur ces dernières, telles les métaheuristiques.

2.3.2.1 Heuristiques et recherches locales

Les heuristiques permettent d'obtenir des solutions réalisables en effectuant des décisions élémentaires consécutives. Typiquement, elles trouvent une solution approchée à un problème NP en temps polynomial. Parmi les heuristiques très simples, on retrouve les algorithmes gloutons (ou *greedy algorithms* [40], chapitre 16). Le principe des méthodes gloutonnes est de faire une succession de choix optimaux localement, jusqu'à ce que l'on ne puisse plus améliorer la solution, et ce sans retour en arrière possible. Ce principe, assez générique, doit être adapté en fonction de la

structure du problème.

Les solutions heuristiques peuvent être améliorées par des recherches locales [72]. Il s'agit de visiter itérativement de proche en proche les solutions voisines d'une solution courante S du problème. On entend par voisinage de la solution S, l'ensemble des solutions que l'on peut obtenir en appliquant à S une transformation simple appelée mouvement. L'idée est de choisir un mouvement améliorant et à partir de ce nouveau voisin généré, explorer à nouveau le voisinage. La solution obtenue après de telles améliorations successives sera alors un minimum local.

2.3.2.2 Métaheuristiques

Les métaheuristiques [21, 62] sont des algorithmes capables de s'extraire de minima locaux. Leur nature stochastiquée leur permet d'explorer plus facilement un espace des solutions de très grande taille. Le principe de base d'une métaheuristique est de parcourir l'espace des solutions à la recherche de son optimum global en utilisant deux mécanismes : l'intensification qui se ramène souvent à une recherche locale et la diversification qui permet de sortir de la région de cet optimum local. Il existe deux grandes familles de métaheuristiques : les mono-solutions et les multi-solutions. Les premières manipulent une solution à la fois alors que les secondes manipulent une population de solutions et sont pour cela appelées métaheuristiques à population.

Dans ce qui suit, Nous donnons un aperçu des métaheuristiques les plus connues en insistant sur celles abordées dans le cadre de la thèse.

Algorithme constructif adapté (*greedy randomized adaptative search procedure* - GRASP [53]) A chaque itération de l'algorithme une nouvelle solution est créée indépendamment des précédentes grâce à un algorithme glouton randomisé. La solution est par la suite améliorée avec une recherche locale. La meilleure solution trouvée lors des différentes itérations est restituée comme résultat.

Recherche locale itérée (*iterated local search* - ILS [62], chapitre 11) La recherche locale itérée est une métaheuristique de trajectoire. Son principe consiste à alterner entre une recherche locale et une procédure de perturbation. Lorsque la méthode découvre un nouvel optimum, elle choisit soit de le prendre comme démarrage référence soit de garder l'ancien. La solution choisie sert de point de départ pour la procédure de perturbation. Il existe plusieurs variantes d'ILS, par exemple certaines acceptent des solutions de qualité détériorée sous certaines conditions, d'autres considèrent plusieurs types de perturbation et elles sont appliquées à certaines étapes appropriées de l'algorithme.

Recherche locale à voisinage variable (*variable neighborhoods search* - VNS [66]) Cette métaheuristique nécessite la définition d'un ensemble de voisinages $(V_1, V_2, ..., V_k)$. Une relation d'ordre entre ces voisinages est à définir en fonction du problème. L'idée principale est de changer itérativement la meilleure solution en explorant différents voisinages dans l'espoir d'améliorer la solution courante ou bien de la diversifier. A partir d'une solution courante s, le voisinage V_i est sélectionné et

une recherche locale est appliquée. Si la solution obtenue est meilleure que la solution s alors la recherche reprend à partir de cette solution avec le voisinage de départ V_1. Sinon, la recherche passe à un voisinage de portée supérieur V_{i+1}. De nombreuses variantes de VNS existent, nous pouvons citer la version réduite (reduced VNS), la version biaisée (skewed VNS) et la méthode de descente à voisinage variable (variable neighborhood descent - VND).

Recuits simulés (*simulated annealing* - SA [83, 134]) Cet algorithme est inspiré du comportement des atomes en métallurgie. Lorsque le métal est refroidi lentement, les atomes ont le temps de s'arranger dans une forme cristalline parfaite équivalente au minimum global. L'analogie avec ce processus est établie en assimilant la fonction objetcif f à de l'énergie. La température T devient un paramètre de contrôle de la convergence de l'algorithme. Plus concrètement, si la solution courante s engendre une amélioration dans la fonction objectif elle est alors acceptée et réalisée. Sinon, la solution est acceptée avec une probabilité $e^{-\frac{f(s')-f(s)}{T}}$. La température T est diminuée au fur et à mesure des itérations et la méthode s'arrête quand T a atteint une certaine valeur proche de 0.

Recherche tabou (*tabu search* - TS [63]) Il s'agit d'une méthode déterministe. Le principe consiste à balayer entièrement un voisinage de la solution courante et à effectuer la meilleure transformation possible, même si cette dernière détériore la fonction-objectif. Pour éviter un comportement cyclique de la méthode, l'historique de la recherche est maintenu dans une liste tabou de longueur limitée. Cette liste sert à interdire tout retour en arrière vers une solution venant d'être visitée. La méthode se termine quand un certain nombre d'itérations est atteint et restitue la meilleure solution trouvée. En pratique, seulement quelques attributs comme par exemple les mouvements ayant permis de passer d'une solution à l'autre sont conservés dans la liste tabou. Ceci dans un souci d'économie de mémoire et de temps de calcul.

Algorithmes évolutionnaires (*evolutionary algorithm* - EA [56]) Les algorithmes évolutionnaires constituent une discipline impliquant la simulation du processus de l'évolution naturelle sur un ordinateur. Ceci peut être vu comme un processus d'optimisation où des individus évoluent dans le temps, afin de devenir de plus en plus adéquats à un environnement donné. Parmi les algorithmes évolutionnaires largement utilisés dans les problèmes d'optimisation, on retrouve l'algorithme génétique (ou *genetic algorithm* - GA) introduit par Holland [69]. C'est une métaheuristique inspirée des processus naturels observés en génétique. Elle consiste à faire évoluer une population d'individus (ensemble de solutions) par des phénomènes de reproduction et de mutation. La population initiale regroupe un ensemble d'individus représentés sous forme de *chromosomes*. A chaque itération, un sous ensemble d'individus appelés *parents* sont sélectionnés de la population pour la reproduction en favorisant la sélection des plus prometteurs (possédant un bon *fitness*). Le croisement entre des individus *parents* génère des nouveaux individus appelés *enfants* qui combinent les caractéristiques de ces derniers. Avec une faible probabilité, une mutation est appliquée sur le chromosome enfant, ceci permet une diversification évitant une convergence prématurée. Deux modes de gestion de la population peuvent être utilisés. Soit la population initiale est remplacée

par la population-enfant lors de l'itération suivante (gestion générationnelle). Soit les enfants générés sont directement intégrés en remplaçant des individus dans la population courante (gestion incrémentale). Les algorithmes génétiques peuvent être une bonne solution pour résoudre des problèmes d'optimisation combinatoire. Cependant, l'inconvénient de cet algorithme est que les opérateurs standards de croisement et de mutation ne permettent pas d'intensifier suffisamment la recherche. C'est pourquoi les algorithmes génétiques sont souvent combinés avec des méthodes de recherche locale. Cette forme hybride entre méthode évolutionnaire et recherche locale désigne les algorithmes mémétiques [98]. Les deux méthodes sont complémentaires car l'une permet de détecter de bonnes régions dans l'espace de recherche alors que l'autre se concentre de manière intensive à explorer ces zones de l'espace de recherche.

Optimisation par essaim particulaire (*particle swarm optimization* - PSO [82]) Cette méthode est inspirée du comportement social des animaux évoluant en essaim. Elle repose sur un ensemble d'individus appelés particules. Une particule représente une solution potentielle qui se déplace dans l'espace de recherche, en quête de l'optimum global. L'intelligence globale de l'essaim est donc la conséquence directe des interactions locales entre les différentes particules qui le constituent. Chaque particule dispose d'une mémoire concernant sa meilleure solution visitée ainsi que la capacité de communiquer avec les particules de son entourage. À partir de ces informations, la particule calcule une vitesse de mouvement qui indique le degré de changement de la solution dans la prochaine itération. La mise à jour de la vitesse est influencée par les trois composantes suivantes :

- Une composante physique : la particule tend à suivre sa direction courante de déplacement. Ceci est contrôlé par le paramètre w appelé le facteur d'inertie ;
- Une composante cognitive : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée. Le paramètre c_1 contrôle le comportement cognitif de la particule ;
- Une composante sociale : la particule tend à se fier à l'expérience de ses congénères et, ainsi, à se diriger vers le meilleur site déjà atteint par ses voisins. Le paramètre c_2 contrôle l'aptitude sociale de la particule.

La combinaison des paramètres w, c_1 et c_2 permet de régler la balance entre les phases de diversification et d'intensification du processus de recherche. L'optimisation par essaim particulaire est à l'origine développée pour les problèmes d'optimisation à variables continues. Il est difficile d'appliquer le schéma PSO dans l'optimisation combinatoire à cause de diverses façons de définir les présentations et les opérateurs nécessaires, et à cause de la difficulté de trouver ceux qui sont adaptés au problème en question [76].

2.4 Problèmes de tournées de véhicules

Les problèmes de tournées de véhicules (VRP) ont été le sujet d'une recherche intensive durant plus de cinquante ans. Cet engouement peut en partie s'expliquer par les applications directes dans le domaine de la logistique, et à leur grande

difficulté. Le VRP implique la planification de routes de livraison à moindre coût, afin de servir un ensemble de clients dispersés géographiquement, tout en respectant les contraintes de capacité des véhicules. Il existe différentes extensions des problèmes de tournées prenant en compte des objectifs et contraintes opérationnelles des contextes d'application visés.

Dans cette section, nous introduisons quelques précisions utiles sur les problèmes de tournées en général. Nous discutons ensuite quelques variantes du problème étudiées dans la littérature. Nous terminons en nous attardant davantage sur les problèmes de tournées avec gains, qui présentent des aspects en lien avec les problématiques que nous traitons.

2.4.1 Présentation générale des problèmes de Tournées de Véhicules

Le Problème le plus simple, le plus ancien et le plus étudié des problèmes de tournées est celui du Voyageur de Commerce (*traveling salesman problem* - TSP [88]). Il s'agit d'un représentant habitant une ville donnée et qui doit visiter un ensemble de clients lors de sa tournée journalière. Il désire bien évidemment d'optimiser certains critères, en particulier minimiser la longueur du trajet total effectué. Le TSP est connu comme NP-Difficile par la réduction au HCP [88]. Le premier programme linéaire pour le TSP est proposé par Dantzig et al. [46]. Cette formulation sert toujours comme base des méthodes de résolution exactes pour le TSP. En particulier, les algorithmes récents basés sur le branch-and-cut peuvent résoudre des instances de grande taille du problème [2].

Si plusieurs représentants d'une même agence doivent se partager un ensemble de clients, le problème devient alors un multi-voyageurs de commerce (*multiple traveling salesmen problem* - MTSP [15]). Il est possible de ramener la résolution du MTSP à la résolution d'un TSP avec une multiplication de villes de départ à un facteur m.

Le problème de tournées de véhicules (*vehicle routing problem* - VRP [126], chapitres 2 - 6) est une généralisation du TSP obtenue lorsque K véhicules de capacité Q sont disponibles au nœud-dépôt. Chaque véhicule doit donc effectuer une tournée réalisable, c'est à dire quitter le dépôt, visiter une fois des clients dont la somme des demandes ne dépassent pas la capacité Q, avant de retourner au dépôt. Le but est de trouver des tournées de coût minimal permettant de livrer tous les clients et respectant les contraintes de capacité des véhicules.

Plus formellement, nous considérons un graphe complet $G = (V, E)$ où V est un ensemble comportant $|V| = n + 1$ nœuds. Le nœud $v_0 \in V$ représente le dépôt où stationnent m véhicules identiques de capacité Q. Les autres nœuds $v_i \in V \backslash \{v_0\}$ représentent des clients i caractérisés par une demande de produit non négative q_i. Les arcs $(i, j) \in E$ représentent la possibilité d'un trajet direct du client i au client j avec un coût de transport de $c_{i,j}$. Par défaut, le coût d'un trajet irréalisable entre deux nœuds ou un trajet bouclé sur la même ville est infini, c'est-à-dire $c_{i,j} = \infty$ si $(i, j) \notin E$ et $c_{i,i} = \infty$. L'objectif est de minimiser la somme des longueurs des

tournées telles que tous les clients soient servis et les contraintes de chargement de véhicules soient respectées.

Le VRP constitue un terrain d'essai privilégié pour de nombreuses méthodes exactes et heuristiques. Plusieurs méthodes exactes, principalement basées sur la méthode branch-and-cut, sont présentées dans les chapitres 2 et 3 de Toth and Vigo [126]. Comme méthodes approchées, nous pouvons citer les algorithmes de recherche tabu [60, 120], le resuit simulé [103] et l'optimisation par colonie de fourmis [30, 79, 138].

Dans les dernières années, la littérature s'est fortement orientée vers des problèmes de tournées de véhicules multi-attributs, pour lesquels les cas d'application pratiques posent toujours d'importants défis. L'objectif de la section suivante est de présenter quelques unes des variantes du VRP les plus courantes.

2.4.2 Diverses variantes des problèmes de tournées

Les cas d'application réels exigent souvent des contraintes additionnelles auxquelles sont soumis les opérationnels. On retrouve par exemple des problèmes impliquant plusieurs dépôts, une flotte de véhicules hétérogène, ou des fenêtres de temps associées aux clients. Ces attributs se traduisent en une vaste littérature, et il serait impossible de donner ici un état de l'art exhaustif de ces problèmes et des méthodes de résolution associées. Nous recommandons l'ouvrage de référence de Toth and Vigo [126] qui présentent en détail les différentes extensions du VRP ainsi que les méthodes associées, exactes et approchées.

2.4.2.1 Problème de tournées avec fenêtres de temps

le VRPTW (*vehicle routing problem with time windows*) est la variante du VRP la plus étudiée [126]. Des fenêtres de temps de visite sont associées aux clients et au dépôt. Chaque arc étant dans ce contexte caractérisé par une durée de trajet généralement assimilé à un coût. La disponibilité d'un client i est représentée par une fenêtre de temps définie par une date de service au plus tôt e_i et une date de service au plus tard l_i. Un véhicule se rendant au client i plus tôt que e_i doit attendre jusque début de la fenêtre de temps. Le temps d'attente résultant est pris en compte dans la contrainte de durée maximum de route. Une arrivée tardive après l_i rend la solution associée irréalisable. Certaines variantes, nommées *"fenêtres de temps souples"* autorisent des arrivées tardives et éventuellement en avance, sanctionnées au moyen de pénalités.

De nombreuses formulations et méthodes exactes sont proposées pour le VRPTW [75]. Nous pouvons citer quelques méthodes récentes de résolution exacte : le branch-and-cut [13], le branch-and-price [52] et le branch-and-price-and-cut [73]. Les métaheuristiques proposées pour le VRPTW sont les algorithmes de recherche tabu [39], recuit simulé [41] et les algorithmes génétiques [16].

2.4.2.2 Problème de tournées avec flotte limitée

Contrairement au cas de la flotte illimitée où une solution triviale consiste à associer une tournée à chaque client, Le m-VRPTW [91] impose une contrainte sur le nombre de véhicules disponibles. En effet, les moyens logistiques étant rarement infinis, une restriction de la flotte à m véhicules peut être introduite à la définition du problème. Cette contrainte a pour effet de rendre plus difficile la production de solutions réalisables. S'il existe une affectation permettant de satisfaire toutes les demandes sur l'horizon de planification, le problème est dit satisfiable et l'objectif est de trouver un ensemble des tournées respectant les contraintes de capacité des véhicules et des fenêtres de temps, tout en minimisant le coût total de transport lié aux arcs empruntés par les véhicules. Une recherche tabu a été proposée par Lau et al. [91] pour résoudre le m-VRPTW. Cette recherche est basée sur le concept de *Holding List* qui est une tournée particulière comportant l'ensemble des clients non routés. Les opérations de transfert et d'échange de clients considérés pour le voisinage sont effectuées sur la Holding List au même titre que sur les autres tournées. On retrouve le même principe sous forme de d' *Ejection Pool* dans les travaux de Lim and Xingwen [94]. Un modèle mathématique amélioré a été proposé par Wang et al. [136]. Le modèle contribue à l'initialisation d'un algorithme génétique conçu pour le m-VRPTW

2.4.2.3 Problème de tournées avec contrainte de distance maximale

Dans le DVRP (*Distance Constrained VRP*), une contrainte sur la distance maximale que peut parcourir chaque véhicule est posée. Ce type de contrainte permet de prendre en compte les possibilités d'épuisement du carburant ou l'autonomie limitée des véhicules imposant leur retour au dépôt pour ravitaillement. Les problèmes avec fenêtres de temps prennent implicitement ce type de contrainte en compte puisque la fenêtre de temps associée au dépôt limite la longueur des tournées. On ne trouve pas beaucoup de travaux traitant spécifiquement le DVRP. Laporte, Desrocher et Nobert [89, 90] ont produit des méthodes exactes pour ce problème. Prins [111] a notamment proposé un algorithme mémétique très efficace basé sur un codage de tournées sans séparateur et sur le découpage optimal de Beasley pour le décodage. Il convient de mentionner que la plupart des heuristiques et métaheuristiques pour le VRP fonctionnent également pour le DVRP, ou sont très facilement adaptables.

2.4.2.4 Problèmes de Tournées avec profits

Une entreprise est souvent amenée à traiter certaines demandes dans une période de temps donnée (par exemple une journée). Cependant, les ressources personnelles et matérielles de l'entreprise ont souvent des limites par rapport à la demande des clients. Les entreprises sont donc confrontées à une demande dépassant leurs moyens et doivent par conséquence identifier les clients à servir dans la période actuelle et ceux à traiter ultérieurement, en vue de maximiser leur profit. Cette problématique

forme une nouvelle classe des problèmes de tournées appelée problèmes de tournées avec profits (*routing problems with profits* - PRPs [51]) . Dans ce type de problème, il est impossible de servir tous les clients. Afin de les discriminer, une valeur P_i, qualifiée de *profit*, est associée à chaque client. Cette grandeur vise à quantifier l'intérêt que représente le service d'un client donné.

Tout comme les problèmes de tournées classiques, les routes construites doivent être de coût minimal. C'est pourquoi des tournées avec profits sont par nature bi-objectifs : les gains de profits à maximiser et les coûts des trajets à minimiser. Plusieurs variantes peuvent figurer dans cette classe en jouant sur la considération des deux objectifs déjà mentionnés. Le Problème de Tournées Sélectives (PTS) [51] s'intéresse à maximiser le gain récolté par tous les véhicules et à limiter, au niveau des contraintes, la longueur de chaque tournée par une distance maximale. Par opposition aux problèmes sélectifs, les problèmes de tournées avec quotas (PTQ) cherchent à minimiser la somme des trajets en se fixant un profit minimal à atteindre. Une autre variante du problème considère simultanément les deux objectifs [24]. La fonction à optimiser est donnée sous forme agrégée.

Dans cette thèse, nous nous sommes intéressés plutôt aux problèmes de nature sélective, plus particulièrement au problème de m-Tournées sélectives (PmTS). Compte tenu de l'origine historique de ce problème, nous préférons la dénomination de *Team Orienteering Problem* (TOP).

Le *Team Orienteering Problem* est abordé dans la littérature, pour la première fois, dans un article de Butt and Cavalier [31] sous le nom de *Multiple Tour Maximum Collection Problem*. La dénomination du TOP a été introduite par Chao et al. [33] pour décrire la course d'orientation en équipe. Dans ce sport d'extérieur, Plusieurs équipes sont en compétition afin de collecter le maximum de points possibles. Chaque membre d'une équipe doit, dans un temps imparti, visiter des lieux prédéterminés où il récupère des points pour son équipe. Les points ne peuvent être collectés qu'une seule fois par un seul membre de l'équipe. La course d'orientation en équipe peut être vu comme un problème de tournées avec une flotte homogène de m véhicules. Les véhicules partent d'un même point de départ d et arrivent au même point d'arrivée a et leur temps de trajet est limité à T_{max}. A chaque client, on associe un profit et chaque client est visité au maximum par un seul véhicule. L'objectif est de choisir les clients à servir et à organiser des services clientèles de façon à respecter la contrainte sur le temps de trajet.

Dans [77], de nouvelles contraintes temporelles ont été ajoutées à l'*Orienteering Problem* afin de retranscrire les exigences de ponctualité des clients. Dans le *Team Orienteering Problem with Time Windows* (TOPTW), chaque client a une durée de service caractérisant le temps pendant lequel un véhicule doit rester stationné pour assurer son service et une fenêtre de temps définissant la plage de temps pendant laquelle peut être positionnée la date de début de service.

En terme de méthodes exactes pour le TOP, les seuls travaux présents dans la littérature sont la génération de colonnes [32], le branch-and-price de [26] et récemment le branch-and-price-and-cut [108]. Parmi les métaheuristiques les plus efficaces pour le TOP, on trouve les recherches tabu et les recherches à voisinage

variable [4], les algorithmes de colonies de fourmis [80] et le path relinking [119].

A notre connaissance, la seule méthode exacte qui a traité le TOPTW a été proposée par Righini and Salani [114]. Les autres méthodes de la littérature sont essentiellement des méthodes approchées comme les algorithmes de colonies de fourmis [97] et les métaheuristiques basées sur des recherches locales [130, 85, 86].

Les domaines d'application du TOP les plus courants sont ceux liés au transport tels que la planification des tournées de services pour les techniciens [123] ou la planification de tournées provisoires pour le service de vente [127].

Motivé par des aspects applicatifs détaillés dans le chapitre 1, nous proposons des métaheuristiques pour résoudre le TOP et le TOPTW : une est basée sur le schéma de PSO et l'autre est basée sur les principes de GA/MA. Les expérimentations conduites sur les instances standards montrent l'efficacité de nos méthodes.

2.5 Conclusion

Dans ce chapitre nous avons défini dans quel contexte notre travail s'inscrit. Tout d'abord, nous avons rappelé certaines notions de base concernant l'optimisation combinatoire. Selon la difficulté d'élaboration d'algorithmes exacts, différentes classes de problèmes d'optimisation sont définies. Ensuite, nous avons présenté des méthodes de résolution exactes comme les approches branch-and-bound et la programmation linéaire. ainsi que des schémas heuristiques et métaheuristiques. A la fin de ce chapitre, une vue d'ensemble sur les problèmes de tournées de véhicules a été présentée. Les caractéristiques combinées au problème de VRP, forment un vaste domaine de la littérature. Les variantes du problème en rapport avec les sujets abordés dans cette thèse sont décrites. Dans les chapitres qui suivent, une description plus précise des problèmes abordés, leurs méthodes de résolution ainsi que leur intérêt théorique et pratique seront présentés.

Optimisation par Essaim Particulaire pour le TOP

Abstract

The team orienteering problem (TOP) is a particular vehicle routing problem in which the aim is to maximize the profit gained from visiting customers without exceeding a travel cost/time limit. This chapter proposes a new and fast evaluation process for TOP based on an interval graph model and a Particle Swarm Optimization inspired algorithm (PSOiA) to solve the problem. Experiments conducted on the standard benchmark of TOP clearly show that our algorithm outperforms the existing solving methods. PSOiA reached a relative error of 0.0005% whereas the best known relative error in the literature is 0.0394%. Our algorithm detects all but one of the best known solutions. Moreover, a strict improvement was found for one instance of the benchmark and a new set of larger instances was introduced.

3.1 Introduction

The term Orienteering Problem (OP), first introduced in [64], comes from an outdoor game played in mountainous or forested areas. In this game, each individual player competes with the others under the following rules. Each player leaves a specific starting point and tries to collect as many rewards as possible from a set of check points in a given time limit before returning to the same starting point. Each check point can reward each player at most once and each player is aware of the position of each check point as well as the associated amount of rewards. There always exists an optimal strategy to achieve the maximum amount of rewards. In general, finding such a strategy (or solving OP) is NP-Hard [64], the player should select a correct subset of check points together with determining the shortest Hamiltonian circuit connecting these points and the starting point. OP and its variants have attracted a good deal of attention in recent years [3, 19, 127, 132] as a result of their practical applications [34, 64, 92, 128] and their hardness [31, 54, 81]. Readers are referred to Vansteenwegen et al. [133] for a recent survey of these problems.

Adding the cooperative aspect to OP, without neglecting the competitive one, yields to the Team Orienteering Problem (TOP) [33]. In this problem, the players are partitioned into teams and players of a team work together to collect as many rewards as possible within the time limit. Each check point can reward each team at most once. The specific vehicle routing problem, analogous to this game that we also denote by TOP, is the problem where a limited number of vehicles are available to visit customers from a potential set, the travel time of each vehicle being limited by a time quota, each customer having a specific profit and being visited at most once. The aim of TOP is to organize an itinerary of visits so as to maximize the total profit. Solving this problem is also NP-Hard [33]. The applications of TOP include athlete recruiting [33], technician routing [24, 123] and tourist trip planning [132, 133]. In this chapter, we are interested in TOP as the core variant of OP for multiple vehicles. This work was motivated by several lines of research first put forward by Veolia Environnement [24, 25].

As far as we know, there are only three exact algorithms for TOP [26, 32, 108]. In contrast to exact solving approaches, a number of heuristics and metaheuristics have been developed for TOP. Two fast heuristics were developed by Butt and Cavalier [31] and by Chao et al. [34]. Tang and Miller-Hooks [123] proposed a tabu search embedded in an adaptive memory procedure. Two tabu search approaches and two versions of a Variable Neighborhood Search (VNS) algorithm were developed by Archetti et al. [4]. Those four methods make use of infeasible tours and of a repairing procedure. Among these, the slow version of the VNS (SVNS) gave very good results on the standard benchmark. Later, Ke et al. [80] developed four versions of an Ant Colony Optimization (ACO) approach. A guided local search and a skewed variable neighborhood search were then proposed by Vansteenwegen et al. [131, 132]. More recently, Bouly et al. [25] introduced giant tours, i.e. permutations of all customers, to represent solutions of TOP and designed an effective Memetic Algorithm (MA). The results of MA [25] were as good as those of SVNS [4] with several strict improvements. Souffriau et al. [119] submitted two versions of a Path Relinking (PR) approach and independently produced the strict improvements. Like [4], PR approach uses a repairing procedure during the relinking phase to deal with infeasible tours. Those tours are obtained from a gradual combination of each of the random generated solutions with the best ones. The slow version of the Path Relinking (SPR), despite its name, required very small computational times. It is also worth mentioning that Tricoire et al. [127] proposed a VNS algorithm for a generalized version of OP and provided their results on the original TOP instances. Furthermore, there are two methods based on Particle Swarm Optimization (PSO) designed to TOP: Bonnefoy [22] developed a PSO algorithm combined with a linear programming technique whereas Muthuswamy and Lam [99] introduced a discrete version of PSO (DPSO) to solve TOP.

In short, three methods stand out as the state-of-the-art algorithms for TOP: the slow version of the VNS (SVNS) in [4], the MA algorithm in [25] and the slow version of the PR (SPR) in [119]. Unlike the other two, MA proposed an interesting technique to represent the solutions of TOP, known as giant tours. This technique was previously introduced in [14] for the Vehicle Routing Problem (VRP). According

to a recent survey on heuristic solutions for variants of VRP [135], it is classified as an indirect representation of the solution space. Indeed, each giant tour represents a neighborhood of solutions from which the best one can easily be extracted by an evaluation process. A heuristic using this representation tends to have better visions during the search and a better chance to reach the global optimum. Several search algorithms exploiting this strategy have been discussed in [112] for the case of VRP and variants.

In this chapter, we propose an effective PSO-inspired algorithm (PSOiA) for TOP. This work is based on our preliminary study of a PSO-based memetic algorithm (PSOMA), which was communicated in [43]. The main contribution of our work is a faster evaluation process than the one proposed in [25]. This enables PSOiA and possibly further methods in the literature to examine a larger number of neighborhoods and explore faster the search space. Experiments conducted on the standard benchmark of TOP clearly show that PSOiA outperforms the existing solution methods of the literature. It achieves a relative error of 0.0005% and detects all but one of the best known solutions. Moreover, a strict improvement was found for one instance of the benchmark. The remainder of this chapter is organized as follows. Section 3.2 provides a formal formulation of TOP. PSOiA and the new optimal split procedure are described in Section 3.3. The dynamic management of the parameters and computational results on the standard benchmark are described in Section 3.4. In section 3.5, we introduce a new set of large instances and provide the respective results. Finally, some conclusions and further developments are discussed in Section 3.6.

3.2 Formulation of the problem

TOP is modeled with a graph $G = (V \cup \{d\} \cup \{a\}, E)$, where $V = \{1, 2, ..., n\}$ is the set of vertices representing customers, $E = \{(i, j) \mid i, j \in V\}$ is the edge set, d and a are respectively departure and arrival vertices for vehicles. Each vertex i is associated with a profit P_i, and each edge $(i, j) \in E$ is associated with a travel cost $C_{i,j}$ which is assumed to be symmetric and satisfying the triangle inequality. A tour R is represented as an ordered list of q customers from V, so $R = (R[1], \ldots, R[q])$. Each *tour* begins at the departure vertex and ends at the arrival vertex. We denote the total profit collected from a tour R as $P(R) = \sum_{i=1}^{i=q} P_{R[i]}$, and the total travel cost/time as $C(R) = C_{d,R[1]} + \sum_{i=1}^{i=q-1} C_{R[i],R[i+1]} + C_{R[q],a}$. A tour R is feasible if $C(R) \leq L$ with L being a predefined travel cost/time limit. The fleet is composed of m identical vehicles. A *solution* S is consequently a set of m (or fewer) feasible tours in which each customer is visited at most once. The goal is to find a solution S such that $\sum_{R \in S} P(R)$ is maximized. One simple way of reducing the size of the problem is to consider only *accessible* customers. A customer is said to be accessible if a tour containing only this customer has a travel cost/time less than or equal to L. For mixed integer linear programming formulations of TOP see [26, 32, 80, 108, 133].

3.3 A PSO-inspired algorithm

Particle Swarm Optimization (PSO) is a swarm intelligence algorithm proposed by Kennedy and Eberhart [82] with the basic idea of simulating the collective behavior of wild animals in the nature. PSO was first used for optimization problems in continuous space as follows. A set known as a *swarm* of candidate solutions, referred to as *particles*, is composed of positions in the search space. The swarm explores the search space according to Equations (3.1) and (3.2). In these equations, x_i^t and v_i^t are respectively the vectors of position and velocity of particle i at instant t. Three values w, c_1 and c_2, called respectively *inertia*, *cognitive* factor and *social* factor, are parameters of the algorithm. Two values r_1 and r_2 are random numbers generated in the interval $[0, 1]$. Each particle i memorizes its best known position up to instant t as x_i^{lbest}, and the best known position up to instant t for the swarm is denoted as x^{gbest}.

$$v_i^{t+1} = w \cdot v_i^t + c_1 \cdot r_1 \cdot (x_i^{lbest} - x_i^t) + c_2 \cdot r_2 \cdot (x^{gbest} - x_i^t) \quad (3.1)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (3.2)$$

With this design, PSO is highly successful at performing optimizations in continuous space [8, 76]. In contrast, when applied to problems of combinatorial optimization, PSO encounters difficulties in interpreting positions and velocities, as well as in defining position update operators. As a result, there are a variety of discrete PSO variants (DPSO) [9], and it is difficult to choose an appropriate variant for any given combinatorial optimization such as TOP.

3.3.1 Basic algorithm

Our PSO works with a population of particles, so called the *swarm* and denoted S. Each particle memorizes its current position, i.e. a representation of a solution, and its best known position, called *local best position*, according to an evaluation process. A basic iteration of the algorithm consists of updating the position of each particle in the swarm. In the standard PSO, this update is influenced by PSO parameters and it takes into account the current position, the local best position and the global best position. In our method, each particle also has a small probability ph to be moved out of its current position and transfered to a completely new position. This new position is generated using a randomized heuristic. Moreover, each new position has pm probability to be improved through a local search process. The algorithm is stopped after $itermax$ consecutive position updates have failed to give rise to new local best. Because $itermax$ is usually set to be proportional to $\frac{n}{m}$ [25, 43], then from now when we say the stopping condition is k, that means $itermax = k \cdot \frac{n}{m}$.

For convenience, the current, local best and global best positions of a particle x are denoted respectively $S[x].pos$, $S[x].lbest$ and $S[best].lbest$. The global scheme is summarized in Algorithm 3.1. Its components are detailed in the next sections.

Algorithm 3.1: Basic algorithm

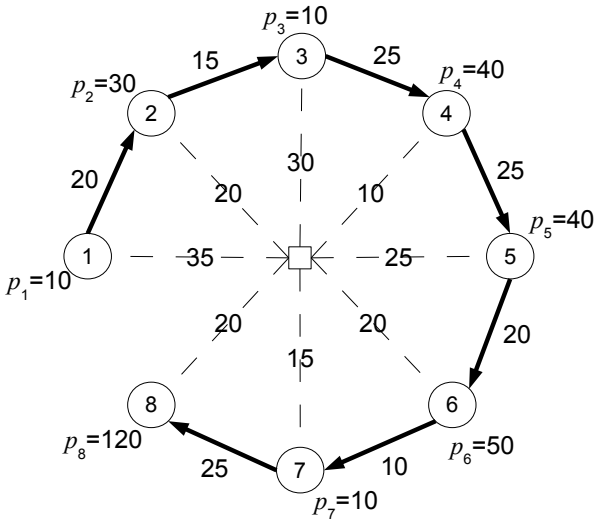
```
Data: S a swarm of N particles;
Result: S[best].lbest best position found;
begin
    initialize and evaluate each particle in S (see Section 3.3.3);
    iter ← 1;
    while iter ≤ itermax do
        foreach x in [1..N] do
            if rand(0,1) < ph then
                move S[x] to a new position (see Section 3.3.3);
            else
                update S[x].pos (see Section 3.3.5);
            if rand(0,1) < pm then
                apply local search on S[x].pos (see Section 3.3.4);
            evaluate S[x].pos (see Section 3.3.2);
            update lbest of S (see Section 3.3.6);
            if (update Rule 3 is applied) (see Section 3.3.6) then
                iter ← 1;
            else
                iter ← iter + 1;
```

3.3.2 Position representation and evaluation

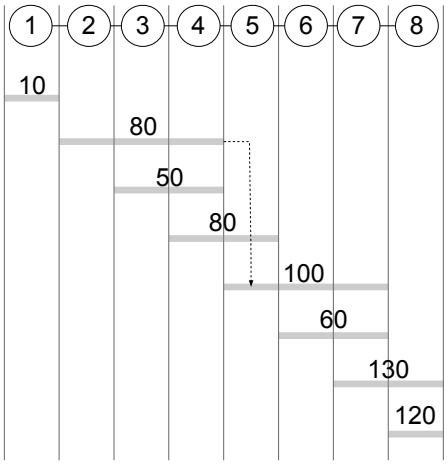
A position in our PSO is a permutation π of all accessible customers, usually referred to as a *giant tour*, in a particular problem scenario. The principle of the *split* technique that optimally extracts a solution from a giant tour was introduced by Bouly et al. [25] for TOP. The basic idea is the following. All possible subsequences of π, denoted by $(\pi[i], \ldots, \pi[i + l_i])$ or $\langle i, l_i \rangle_\pi$ for short, that can form a feasible tour of TOP are considered. For convenience, we use the term *extracted tours* or simply tours in this section to refer to these subsequences. The goal of a *split* procedure is then to find a set of m distinct tours (without shared customer) such that the sum of their profits is maximized. Such a procedure guarantees that if a set of tours forming an optimal solution for the TOP is currently present as subsequences in a permutation π^*, the application of the split procedure on π^* will return the optimal TOP solution.

The authors of [25] proposed a split procedure for TOP. The algorithm requires to find the longest path in an acyclic auxiliary graph. This graph represents the *successor* relations between extracted tours, i.e. the possibility of a tour to follow another in a valid solution. They also introduced the notion of *saturated* tours, i.e. a tour in which l_i is maximal (denoted by l_i^{max}), and proved that solutions containing only saturated tours are dominant. Therefore, only *saturated* tours were considered in their procedure and the number of arcs in the acyclic graph is reduced. The *worst*

case complexity of their procedure is $O(m \cdot n^2)$.



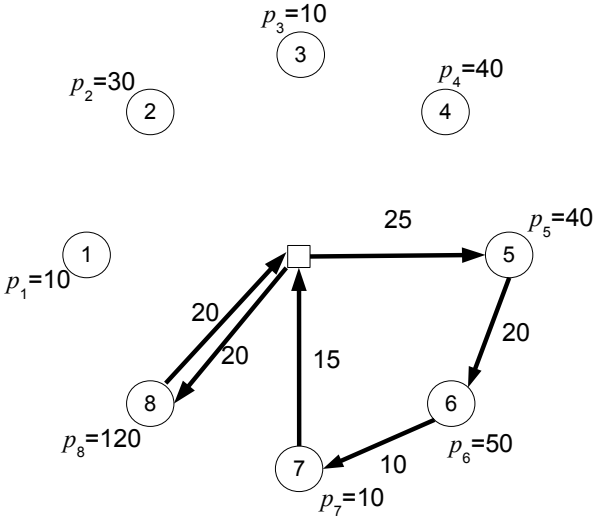
a) a split problem



b) the weighted interval model

		Γ	
I	*succ*(*I*)	1	2
1	2	130	**220**
2	5	130	**220**
3	5	130	**220**
4	6	130	**220**
5	8	130	**220**
6	8	130	180
7	0	130	130
8	0	**120**	120

c) the solving steps



d) a saturated optimal solution

Figure 3.1: The new evaluation process for the same split problem described in [25] with 8 customers, $m = 2$ and $L = 70$.

In this work, the limited number of saturated tours is exploited more efficiently to reduce the complexity of the evaluation process. Before going in the detail of our new split procedure, we recall the definition of a knapsack problem with conflicts (KPCG) [137] as follows. In a KPCG, we have a set of items to be put into a knapsack. A value and a volume are associated to each item. The knapsack has a limited volume, so it cannot generally hold all items. In addition to the knapsack volume, some items are in conflict with each other and they cannot be put in the knapsack together. The aim of the KPCG is to find a subset of items to fit into the knapsack such that the sum of their values is maximized. In such a problem, the conflicts between items are usually modeled with a graph, called *conflict graph*. We also recall the definition of an interval graph [124] as follows. A graph $G = (V, E)$

is called an interval graph if there is a mapping I from V to sets of consecutive integers (called *intervals*) such that for all i and j of V, $[i, j] \in E$ if and only if $I(i) \cap I(j) \neq \emptyset$. Then the following proposition holds for the split procedure of TOP.

Proposition 3.3.1. *The split procedure can be done optimally in $O(m \cdot n)$ time and space.*

Proof. Each possible tour extracted from a giant tour is in fact a set of positions of customers in the giant tour. Since these customers are adjacent in the giant tour, the positions are consecutive integers and the set of extracted tours can be mapped to the set of vertices of an interval graph X. Additionally, an edge of X (or a non-empty intersection between two sets of positions) indicates the presence of shared customers between the associated tours. As mentioned above, a split procedure looks for m tours without shared customer such that the sum of their profit is maximized. So this is equivalent to solve a knapsack problem with X as the conflict graph, a unitary volume for each item and m as the knapsack's volume. In this particular knapsack problem, the number of items is equal to the number of possible tours. This number is equal to n when only saturated tours are considered. Based on the work of Sadykov and Vanderbeck [116], we deduce that such a problem can be solved in $O(m \cdot n)$ time and space. □

Our new evaluation process is summarized as below. For each saturated tour starting with customer $\pi[i]$, we use $P[i]$ to denote the sum of profits of its customers. Its *first successor*, denoted by $succ[i]$, is computed as follows:

$$succ[i] = \begin{cases} i + l_i^{max} + 1 & \text{if } i + l_i^{max} + 1 \leq n \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

A two-dimensional array Γ of size $m \cdot n$ is used to memorize the maximum reachable profit during process. The algorithm then browses the saturated tours in reversed order, meaning from customer $\pi[n]$ to customer $\pi[1]$, and updates Γ based on the recurrence relation described in Equation 3.4.

$$\Gamma(i, j) = \begin{cases} \max\{\Gamma(succ[i], j-1) + P[i], \Gamma(i+1, j)\} & \text{if } 1 \leq i \leq n \text{ and } 1 \leq j \leq m \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

At the end, $\Gamma(1, m)$ corresponds to the profit of the optimal solution. A simple backtrack is then performed on Γ in order to determine the corresponding tours. That is to say if $\Gamma(succ[i], j-1) + P[i]$ is used over $\Gamma(i+1, j)$ in the relation, then the saturated tour starting with customer $\pi[i]$ belongs to the optimal solution.

Figure 3.1 depicts the same example of the split problem described in [25] but with the new evaluation process. More precisely, in this problem we have 8 customers with π = (1, 2, 3, 4, 5, 6, 7, 8), profits (10, 30, 10, 40, 40, 50, 10, 120), $L = 70$ and $m = 2$. According to the distances given in the figure, the saturated tours are $\langle 1, 0\rangle$, $\langle 2, 2\rangle$, $\langle 3, 1\rangle$, $\langle 4, 1\rangle$, $\langle 5, 2\rangle$, $\langle 6, 1\rangle$, $\langle 7, 1\rangle$ and $\langle 8, 0\rangle$ with profits 10, 80, 50, 80, 100, 60, 130 and 120 respectively. The interval model is shown in Figure 3.1.b and

the detail of the first successor relations as well as solving steps are given in Figure 3.1.c. The new algorithm actually returns the same solution composed of the same saturated tours (starting with customers 5 and 8) as expected in [25].

3.3.3 Randomized heuristics

Particle positions in the swarm, including local best positions, are initialized to a random sequence. In order to accelerate the algorithm, a small portion of the swarm containing N_{IDCH} particles will have their local best positions generated using a good heuristic. During the search, a faster heuristic is occasionally used to generate a completely new position for a particle. The heuristics that we use are randomized variants of the Iterative Destruction/Construction Heuristic (IDCH) of [25].

The core component of IDCH is a Best Insertion Algorithm (BIA). Our BIA considers a partial solution (which can be empty) and a subset of unrouted customers to be inserted in the solution. This constructive method then evaluates the insertion cost $\frac{C_{i,z}+C_{z,j}-C_{i,j}}{(P_z)^\alpha}$ of any unrouted customer z between any couple of successive customers i and j in a tour r. The feasible insertion that minimizes the cost is then processed and the method loops back to the evaluation of the remaining unrouted customers. If more than one possible insertion minimizes the insertion cost, one of them is chosen at random. This process is iterated until no further insertions are feasible, either because no tour can accept additional customers, or because all the customers are routed. The only parameter of BIA is α and it is set to 1 in [25, 119]. In this work, a random value of α is generated each time BIA is called. This generation makes our IDCH less predictable and actually a randomized heuristic. The computational method used to generate α is detailed in Section 3.4.

IDCH is described as follows. Firstly, BIA is called to initialize the current solution from scratch. On following iterations a small part of the current solution is destroyed by removing a limited random number (1, 2 or 3) of random customers from tours, and a 2-opt procedure is used to reduce the travel cost of tours. A reconstruction phase is then processed using a Prioritized Best Insertion Algorithm (PBIA). The destruction and construction phases are iterated, and each time a customer remains unrouted after the construction phase its priority is increased by the value of its associated profit. In the PBIA, the subset of unrouted customers with the highest priority is considered for an insertion using a BIA call. When no more of these customers can be inserted, unrouted customers with lower priorities are considered, and so on. The idea behind this technique is to explore solutions composed of high profit customers. IDCH memorizes the best discovered solutions so far and stops after a fixed number of Destruction/Construction iterations without improvement of this solution. This number is set to n for the fast version of IDCH. This version is used to generate a new position for a particle when it is moved out of its current position. For the slower version used to initialize the PSO, this value is set to n^2. In the slow version, after n iterations without improvement a diversification process is applied. This involves destroying a large part of the

solution while removing a number (bounded by n/m rather than by 3) of customers from tours then applying 2-opt to each tour to optimize the travel cost, and finally performing the reconstruction phase.

3.3.4 Improvement of positions through local search

In our PSO, whenever a new position, i.e. a new permutation, is found, it has a *pm* probability of being improved using a local search technique (LS). This LS contains 3 neighborhoods which were proved to be efficient for TOP [25]:

- *shift operator*: evaluate each permutation obtained by moving each customer i from its original position to any other position in the permutation.
- *swap operator*: evaluate each permutation obtained by exchanging every two customers i and j in the permutation.
- *destruction/repair operator*: evaluate the possibility of removing a random number (between 1 and $\frac{n}{m}$) of customers from an identified solution and then rebuilding the solution using BIA procedure described in the previous section.

The procedure is as follows. One neighborhood is randomly chosen to be applied to the particle position. As soon as an improvement is found, it is applied and the LS procedure is restarted from the new improved position. The LS is stopped when all neighborhoods are fully applied without there being any improvement. In addition, we enhanced the randomness of shift and swap operators. That is to say the possibilities of moving or exchanging customers in those operators are evaluated in random order.

3.3.5 Genetic crossover operator to update position

In combinatorial optimization, the particle position update of PSO can be interpreted as a recombination of three positions/solutions according to inertia, cognitive and social parameters. There are various ways of defining this kind of recombination operator [9]. In our approach, the recombination operator is similar to a genetic crossover whose core component is an extraction of l customers from a permutation π. To make sure that a customer can be extracted at most once from sequential calls of the core component, a set M is used to mark extracted customers from previous calls. The extracted subsequence is denoted π_M^l and the procedure is described as follows:

- Step 1 : generate a random location r in π and initialize π_M^l to empty.
- Step 2 : browse customers from $\pi[r]$ to $\pi[n]$ and add them to the end of π_M^l if they are not in M. If $|\pi_M^l|$ reaches l then go to Step 4, otherwise go to Step 3.
- Step 3 : browse customers from $\pi[r]$ down to $\pi[1]$ and add them to the beginning of π_M^l if they are not in M. If $|\pi_M^l|$ reaches l then go to Step 4.
- Step 4 : add customers from π_M^l to M.

With the core component, the position update procedure of particle x from the swarm S with respect to the three PSO parameters w, c_1 and c_2 is as follows:

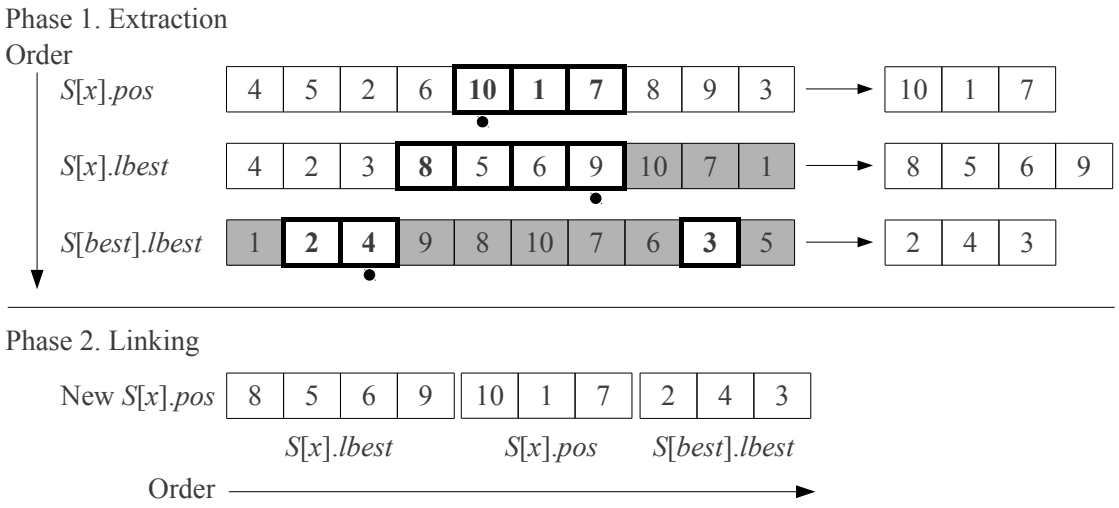


Figure 3.2: An example of position update for an arbitrary instance of ten customers. Black dots represent random generated locations r and shaded boxes represent marked customers from M during Phase 1.

- Phase 1 : apply sequentially but in a random order the core component to extract subsequences from $S[x].pos$, $S[x].lbest$ and $S[best].lbest$ with a common set M of customers to be skipped. M is initialized to the empty set and the desired numbers of customers to be extracted from $S[x].pos$, $S[x].lbest$ and $S[best].lbest$ are respectively $w{\cdot}n$, $(1{-}w){\cdot}n{\cdot}\frac{c_1.r_1}{(c_1.r_1+c_2.r_2)}$ and $(1{-}w){\cdot}n{\cdot}\frac{c_2.r_2}{(c_1.r_1+c_2.r_2)}$. Here r_1 and r_2 are real numbers whose values are randomly generated in the interval $[0, 1]$ with a uniform distribution. Real numbers obtained from those computations are truncated to integral values.
- Phase 2 : link three extracted subsequences in a random order to update $S[x].pos$.

To illustrate the update procedure, we consider an arbitrary instance of TOP with ten customers and an arbitrary particle x with $S[x].pos$ =(4, 5, 2, 6, 10, 1, 7, 8, 9, 3), $S[x].lbest$ = (4, 2, 3, 8, 5, 6, 9, 10, 7, 1) and $S[best].lbest$ = (1, 2, 4, 9, 8, 10, 7, 6, 3, 5). PSO parameters are $w = 0.3$, $c_1 = 0.5$ and $c_2 = 0.3$. Random variables r_1 and r_2 generated are respectively 0.5 and 0.5. Then the desired numbers of customers to be extracted for $S[x].pos$, $S[x].lbest$ and $S[best].lbest$ are respectively 3 ($= \lfloor 0.3{*}10 \rfloor$), 4 ($= \lfloor (1-0.3) * 10 * 0.5 * 0.5/(0.5 * 0.5 + 0.3 * 0.5) \rfloor$) and 3 ($= 10 - 3 - 4$). Random extraction order in Phase 1 is ($S[x].pos$, $S[x].lbest$, $S[best].lbest$) and random linking order in Phase 2 is ($S[x].lbest$, $S[x].pos$, $S[best].lbest$). Figure 3.2 gives an example of the update procedure that indicates the new position for the particle x of (8, 5, 6, 9, 10, 1, 7, 2, 4, 3).

Our particle position update procedure therefore works with the standard PSO parameters w, c_1 and c_2, the only restriction being that w has to be in the interval $[0, 1[$. Our PSO approach can be classified as PSO with position only, given that no velocity vector is used [105]. It is noteworthy to mention that the core component was created to adapt to a linear permutation order, but it can easily be adapted to a circular order by changing Step 3.

3.3.6 Swarm local best update

In some situations, PSO can be trapped in a local optimum, especially when all the local best positions of particles in the swarm are identical. In our approach, the fact that a particle can be randomly moved out of its current position reduces this premature convergence. However, the effect of this reduction is only partial because the probability to move a particle out of its current position is set to a small value. This setting is due to two main reasons: firstly, a frequent use of the IDCH heuristic to generate new positions is time-consuming and secondly, a frequent use of perturbing operations is undesired in a PSO algorithm [139].

So then to strengthen the diversification process, whenever a new position is found by a particle x in the swarm S, instead of updating $S[x].lbest$, the algorithm will search for an appropriate particle y in the swarm using a similarity measure and update $S[y].lbest$. The similarity measure is based on two criteria: the total collected profit and the travel cost/time of the identified solution. Two positions are said to be *similar* or *identical* if the evaluation procedure on these positions returns the same profit and a difference in travel cost/time that is lower than a value δ. Our update rules are based on Sha and Hsu [117] but simplified as follows. For convenience, the particle having the worst local best position of the swarm is denoted as $S[worst]$.

- Rule 1 : the update procedure is applied if and only if the performance of new position $S[x].pos$ is better than the worst local best $S[worst].lbest$.
- Rule 2 : if there exists a particle y in S such that $S[y].lbest$ is similar to $S[x].pos$, then replace $S[y].lbest$ with $S[x].pos$.
- Rule 3 : if no such particle y according to Rule 2 exists, replace $S[worst].lbest$ with $S[x].pos$. Each successful application of this rule indicates that a new local best has been *discovered* by the swarm.

The implementation of these rules was made efficient through the use of a binary search tree to sort particles by the performance of their local best positions using the two criteria. In the next section, the performance of our PSO on the standard benchmark for TOP is discussed.

3.4 Numerical results on the standard benchmark

PSOiA is coded in C++ using the Standard Template Library (STL) for data structures. The program is compiled with GNU GCC in a Linux environment, and all experiments were conducted on an AMD Opteron 2.60 GHz. In order to compare the performance of our approach with those of the existing algorithms in the literature, we use 387 instances from the standard benchmark for TOP [33]. These instances comprise 7 sets. Inside each set the original number of customers and customer positions are constant, however the maximum tour duration L varies. Therefore the number of accessible customers are different for each instance. The number of vehicles m also varies between 2 and 4.

3.4.1 Protocol and performance metrics

Our approach was tested using the same protocol as in [80, 99, 119]. For each instance of the benchmark, the algorithms were executed 10 times. The average and maximal scores as well as the average and maximal computational times were recorded. In order to evaluate separately the performance of different configurations or methods, the best known result in the literature for each instance, denoted by Z_{best}, is used as the reference score of the instance. These best results for all instances of the benchmark are collected from [4, 25, 43, 80, 119, 123] and also from our PSO algorithms, but not from Chao et al. [34] because the authors used a different rounding precision and some of their results exceeded the upper bounds given in [26].

For an algorithm tested on an instance, obtained solutions of 10 runs are recorded and we use Z_{max} and Z_{avg} to denote respectively the maximal and average scores of these runs. Then the relative percentage error (RPE) and the average relative percentage error (ARPE) are used to evaluate the performance of the algorithm. RPE is defined as the relative error between Z_{best} and Z_{max}. It was used in [99, 119] to show the performance of the algorithm over 10 runs.

$$RPE = \frac{Z_{best} - Z_{max}}{Z_{best}} \cdot 100 \tag{3.5}$$

ARPE is defined as the relative error between Z_{best} and Z_{avg}. It was used in [99] to show the robustness of the algorithm over 10 runs. In other words, a small value of ARPE indicates a higher chance of getting a good score (or a small RPE) for a limited number of runs of the algorithm on the instance. The instances, for which there is no accessible customer (or $Z_{best} = 0$) are discarded from the comparison. The number of instances is then reduced to 353.

$$ARPE = \frac{Z_{best} - Z_{avg}}{Z_{best}} \cdot 100 \tag{3.6}$$

For a set of instances, the respective average values of RPE and ARPE of the instances are computed to show the performance and robustness of the algorithm. For a benchmark composed of different sets, the average value of the latter ones on all the sets is computed to show the overall performance and robustness of the algorithm on the benchmark. As a complement measure for a benchmark, NBest is used to denote the number of instances in which Z_{best} are reached.

3.4.2 Parameter setting

Values of some parameters are directly taken from the previous studies of [25, 43]. Therefore, we did not do further experiments on those parameters:

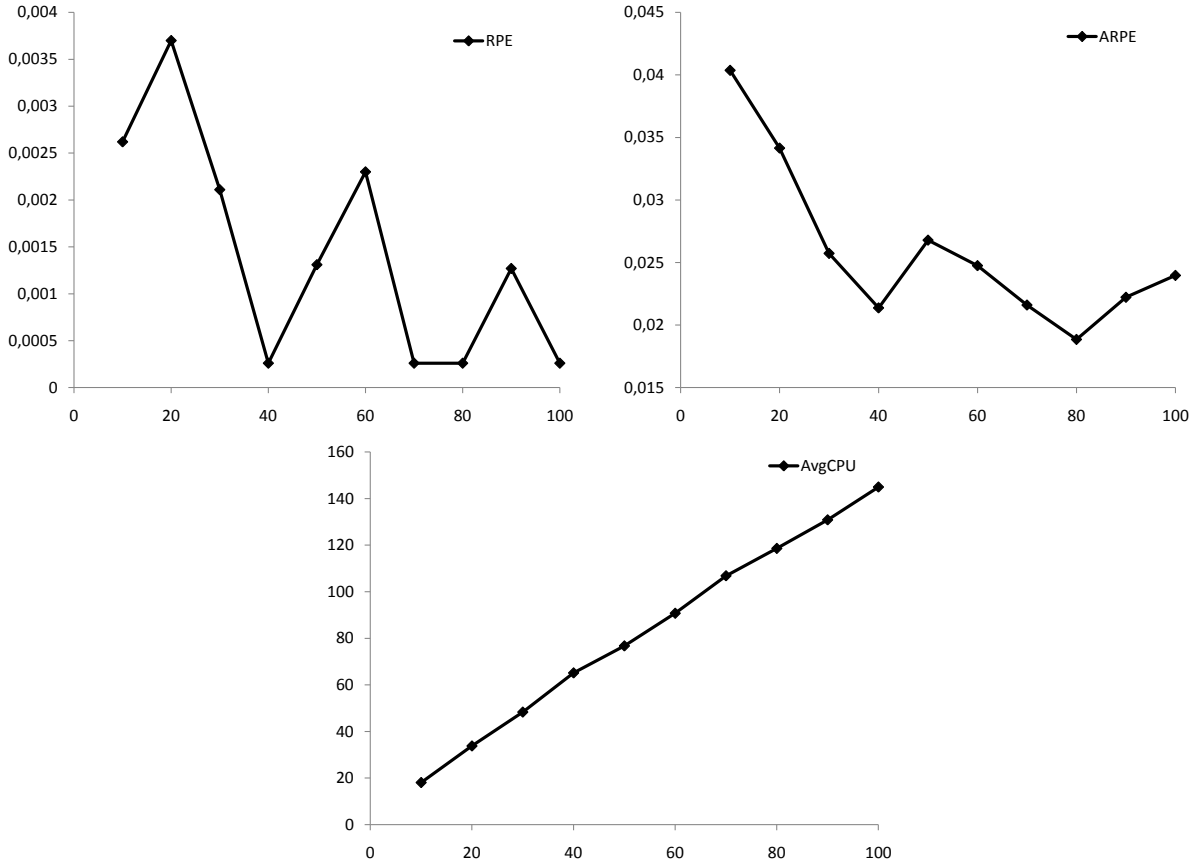


Figure 3.3: Performance of PSOiA in terms of the stopping condition k.

- N, the population size, is set to 40.
- N_{IDCH}, the number of local best positions initialized with the slow version of IDCH, is set to 5.
- pm, the local search rate, is set to $1 - \frac{iter}{itermax}$.
- δ, the similarity measurement of particles, is set to 0.01.
- c_1, c_2, the cognitive and social factors of the particles, are set to 0.5 ($c_1 = c_2 = 0.5$).
- w, the inertia parameter, decreases gradually as the algorithm proceeds. It is initialized to 0.9 and multiplied by 0.9 after each iteration of the PSO.
- α, the control parameter of intuitive criteria of the BIA heuristic, is generated as follows. Two random numbers r_1 and r_2 are first generated in $[0, 1]$ with a uniform distribution, then $\alpha = 1 + 2 \cdot \frac{r_1}{r_1 + r_2}$ is computed.

The most important parameter which could be up for discussion is the stopping condition k. We tested PSOiA on the 353 instances of the benchmark using varied values of k from 10 to 100 with steps of 10. In order to maximally exploit in these tests the crossover operator and the evaluation process, we set the probability ph of a particle to be moved out of its current position equal to 0.1. We will return to the ph parameter later (once k is fixed) to check whether it is over-tuned. Figures

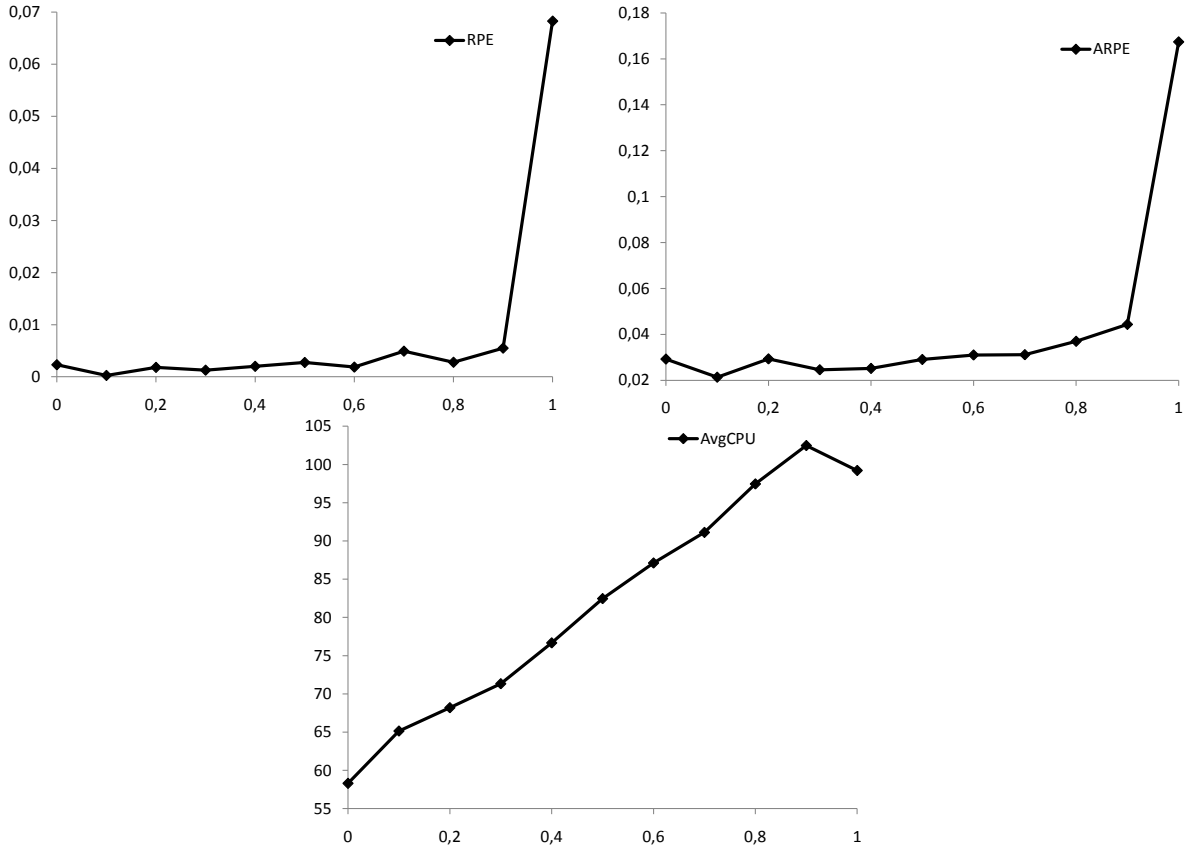


Figure 3.4: Performance of PSOiA in terms of the probability ph of a particle to be moved out of its current position.

3.3 illustrate the evolution of RPE, ARPE and the average computational time in terms of k. One may notice that from $k = 40$, the algorithm starts to provide the best RPE and interesting values of ARPE. On the other hand, the computational time linearly increases in terms of k, hence the value $k = 40$ were selected to present our final results of PSOiA.

Next, we set k to 40 and varied the value of ph from 0 to 1 with a step equal to 0.1. Figures 3.4 show the evolution of RPE, ARPE and the average computational time in terms of ph. In these tests, the computational time linearly increases in terms of ph (with a small exception for $ph = 1.0$) and value 0.1 is the right choice for the parameter.

3.4.3 Comparison with the literature

The results of PSOiA ($k = 40$) on instances of Chao's benchmark are then compared with the state-of-the-art algorithms in the literature:

- SVNS proposed by Archetti et al. [4], tested on an Intel Pentium 4 2.80 GHz,
- MA proposed by Bouly et al. [25], tested on an Intel Core 2 Duo 2.67 GHz,

Method	Year	RPE average for each data set					NBest
		4	5	6	7	avg	
SVNS	2007	0.0680	0.0267	0	0.0627	0.0394	134
ACO	2008	0.3123	0.0355	0	0.0064	0.0885	128
MA	2010	0.0548	0.0612	0	0.0571	0.0433	129
SPR	2010	0.1157	0.0465	0	0.0454	0.0519	126
DPSO	2011	2.0911	0.7828	0.3375	1.7618	1.2433	39
MA10	2011	0.0304	0.0612	0	0.0127	0.0261	146
PSOMA	2011	0.0262	0.0151	0	0.0211	0.0156	146
PSOiA	2012	0.0019	0	0	0	0.0005	156

Table 3.1: Performance comparison based on RPE average for each data set of the relevant instances.

- SPR proposed by Souffriau et al. [119], tested on an Intel Xeon 2.50 GHz,
- PSOMA (with $w = 0.07$, the best configuration) described in [43] as the preliminary study of this work, tested on an AMD Opteron 2.60 GHz.

On the comparison between computers in use, machine performances of PSOiA, PSOMA, MA/MA10 [25] and SPR [119] are almost the same: recent dual-core processors with clock frequency varying from 2.50 GHz to 2.67 GHz. SVNS [4] used a computer with higher clock frequency (2.8 GHz) but that was a Pentium 4. It is supposed to have a lower performance than the others.

In [119], the authors of SPR algorithm talk about the 157 relevant instances of sets 4, 5, 6 and 7 and show only their results on these instances. Therefore, we will provide the comparison focused on these 157 instances. We also noted that results of SVNS were taken from the website of the first author of [4]. These results were updated in 2008 and the rounding convention problem reported in [25, 80] was corrected. It also appears that these results are better than the ones published in the journal article [4]. Additionally, a different testing protocol which considered only 3 runs for each instance of the benchmark had been used for SVNS and MA. So in [43], the source code of MA [25] was received from the authors and turned to match the new testing protocol: 10 executions per instance. Results of this new test for MA is denoted by MA10.

Our results are also compared with the other swarm intelligence algorithms available in the literature:

- Sequential version of the Ant Colony Optimization (ACO) proposed by Ke et al. [80], tested on an Intel CPU 3.0 GHz,
- Discrete Particle Swarm Optimization (DPSO) proposed by Muthuswamy and Lam [99], tested on an Intel Core Duo 1.83 GHz.

Table 3.1 reports RPE averages for each data set of all methods. From this table, we observe that PSOMA (with very basic PSO components) already outperforms the other methods in the literature. This motivates our choice of testing the new optimal split procedure on PSO scheme instead of MA one. Regarding PSOiA, the

Method	Year	ARPE average for each data set				
		4	5	6	7	avg
SVNS	2007	n/a	n/a	n/a	n/a	n/a
ACO	2008	1.8663	0.8228	1.1754	0.5118	1.0941
MA	2010	n/a	n/a	n/a	n/a	n/a
SPR	2010	n/a	n/a	n/a	n/a	n/a
DPSO	2011	5.1956	3.7100	2.0073	4.1986	3.7779
MA10	2011	0.2068	0.0953	0.0169	0.1056	0.1061
PSOMA	2011	0.2851	0.0904	0	0.1790	0.1386
PSOiA	2012	0.1105	0.0336	0	0.0305	0.0436

Table 3.2: Robustness comparison based on ARPE average for each data set of the relevant instances.

Method	Average CPU time in seconds for each data set						
	1	2	3	4	5	6	7
SVNS	7.78	0.03	10.19	457.89	158.93	147.88	309.87
ACO	5.77	3.16	6.50	37.09	17.36	16.11	30.35
MA	1.31	0.13	1.56	125.26	23.96	15.53	90.30
SPR	n/a	n/a	n/a	36.74	11.99	8.96	27.28
DPSO	n/a	n/a	n/a	n/a	n/a	n/a	n/a
MA10	1.95	0.24	2.06	182.36	35.33	39.07	112.75
PSOMA	0.18	0.01	0.49	83.89	14.72	7.59	49.09
PSOiA	2.15	0.41	3.18	218.58	49.5	47.08	97.47

Table 3.3: Average CPU time for each data set of the standard benchmark.

Method	Maximal CPU time in seconds for each data set						
	1	2	3	4	5	6	7
SVNS	22	1	19	1118	394	310	911
ACO	n/a	n/a	n/a	n/a	n/a	n/a	n/a
MA	4.11	0.531	3.963	357.053	80.19	64.292	268.005
SPR	n/a	n/a	n/a	n/a	n/a	n/a	n/a
DPSO	n/a	n/a	n/a	n/a	n/a	n/a	n/a
MA10	8.59	1.16	6.34	635.75	113.58	96.89	443.59
PSOMA	4.35	0.03	4.88	466.65	78.12	48.77	350.86
PSOiA	10.61	2.20	10.93	1274.52	170.09	115.93	420.50

Table 3.4: Maximal CPU time for each data set of the standard benchmark.

Method	Year	Number of instances (%)				
		4	5	6	7	avg
SVNS	2007	n/a	n/a	n/a	n/a	n/a
ACO	2008	7	18	0	21	11
MA	2010	n/a	n/a	n/a	n/a	n/a
SPR	2010	n/a	n/a	n/a	n/a	n/a
DPSO	2011	0	2	0	0	1
MA10	2011	33	84	93	47	61
PSOMA	2011	26	76	100	33	55
PSOiA	2012	52	87	100	74	72

Table 3.5: Stability comparison based on the number of instances having zero APRE.

results are almost perfect with zero RPE for sets 5, 6, 7 and only one instance was missed for set 4 with a very small value of RPE. Table 3.2 reports ARPE averages for each data set of the standard benchmark. From that table, we observe that PSOMA is less robust than MA10 on data sets 4 and 7. However, it is more robust than MA10 on data sets 5 and 6. Finally, PSOiA is the most robust method. The ARPE average on all data sets of PSOiA is 0.0436% which almost equivalent to the RPE averages on all data sets of the state-of-the-art algorithms (SVNS, SPR and MA) reported in the literature (ranging from 0.0394% to 0.0519%) as shown in Table 3.1.

Tables 3.3 and 3.4 report the average and maximal CPU times respectively for each data set of all methods. From this table, we notice that ACO and SPR are fast methods. However, their performances are not as good as SVNS, MA/MA10, PSOMA and PSOiA as seen in Tables 3.1 and 3.2. SVNS is slower than the others. PSOMA is quite faster than MA10 but as mentioned above, MA10 is more robust than PSOMA. Computational efforts required for PSOiA and MA10 are almost the same. Based on a remark of [127] and our own verification, it is worthy to mention that the maximal CPU times of ACO method reported in [80] are in fact the maximal average CPU times, i.e. for each instance the average CPU time is computed, then the maximal value of these CPU times is reported for a whole set. Therefore, the maximal CPU times of ACO method are marked as n/a (not available) in Table 3.4.

Table 3.5 reports the number of instances (in percent) for which the value of ARPE is zero, which means that the results of all runs are identical or that the algorithm is stable. From this table, one may notice that PSOiA is stable in most cases (72%). Additionally, the performance analysis of SVNS, MA, SPR, PSOMA and PSOiA indicates that the results from data sets 4 and 7 are generally less stable than those from data sets 5 and 6. This can be explained by the differences between the features of those instances. Data sets 4 and 7 contain up to 100 customers for which both profits and positions are randomly distributed. On the other hand, data sets 5 and 6 have at most 64 customers arranged in a grid such that large profits are assigned to customers located far away from the depots.

Finally, detailed results of MA10, PSOMA and PSOiA for the 157 instances are reported in Tables 3.7, 3.8, 3.9 and 3.10. For each instance, columns CPU_{avg} report the average computational time in seconds of the ten runs. Complete results of the 353 tested instances are available at http://www.hds.utc.fr/~moukrim. According to these results, $p4.4.n$ is the only instance of the whole benchmark from which PSOiA was not able to find the best known solution. One unit of profit was missed for this instance. Furthermore, a strict improvement was detected for instance $p4.2.q$ with a new score of 1268 instead of 1267.

3.5 A set of larger instances for the team orienteering problem

From the previous section, we observe that PSOiA achieves a value of RPE of 0.0005%. Therefore, it would be very difficult to develop better heuristics for the current standard benchmark instances. In order to promote algorithmic developments for TOP, we introduce a new set of benchmark instances with a larger number of customers. Our new instances are based on the OP instances of Fischetti et al. [54] with the transformation of Chao et al. [33]. This transformation consists of designing the travel length limit of vehicles for TOP as $L^{TOP} = \frac{L^{OP}}{m}$. In this formulation, m is the number of vehicles of the new TOP instance and L^{OP} is the travel length limit of the vehicle of the former OP instance.

We used instances from the two classes described in [54] to generate TOP instances. According to the authors, the first class was derived from instances of the Capacitated Vehicle Routing Problem (CVRP) [35, 113] in which customer demands were transformed into profits and varied values of L^{OP} were considered. The second class was derived from instances of the Traveling Salesman Problem (TSP) [113] in which customer profits were generated in different ways: equal to 1 for each customer (gen1); using pseudo-random function so that the output values are in $[1, 100]$ (gen2); using distance-profit function such that large profits are assigned to nodes far away from the depots (gen3).

In total, 333 new instances were used in our test. It can be seen that PSOiA is very stable for a large part of those instances, especially the ones from the CVRP benchmark. So Table 3.11 reports the results of PSOiA for which the value of ARPE is non-zero. A complete specification, including the number of accessible customers n, the number of vehicles m, the travel length limit L and the way to generate the profits for the customers gen, is also given for each instance. The values in the last row corresponding to Z_{avg} and CPU_{avg} columns respectively indicate the ARPE and the average computational time on the set of instances. All tested instances and the other results are available on the previously mentioned website.

In addition, we also analyzed the computational behavior of PSOiA on the new instances according to the various generations of the profits (namely gen1, gen2 and gen3). In this analysis, for each TSP instance, three variants of TOP instances are available. This implies the same sample size of 93 instances per generation and

Generation	Number of instances with ARPE zero (%)	CPU_{avg}
gen1	81%	4123.84
gen2	73%	4764.23
gen3	75%	5357.32
	76%	4748.47

Table 3.6: Influence of profit generations on the stability of PSOiA

provides a fair comparison. Table 3.6 reports the number of instances (in percent) for which the ARPE is zero and the average computational time CPU_{avg} for each generation. From this table, one may notice that PSOiA is more stable and requires less computational effort on generation $gen1$ (equal profits) than on generations $gen2$ (random profits) and $gen3$ (large profits distributed to customers located far away). Finally, it should be noted that the sample size to analyze the stability of PSOiA according to the positions of the customers is not statistically large enough to reveal the detail.

3.6 Conclusion

This chapter presented an effective Particle Swarm Optimization approach for the Team Orienteering Problem. The approach uses giant tours to indirectly encode particle positions. A new fast evaluation process based on an interval graph model was proposed. This process enabled more iterations for the PSO without increasing the global computational time. Numerical results on the standard benchmark for TOP demonstrate the competitiveness of the algorithm. Our approach outperforms the prior methods both in terms of computational time and solution quality. Hence it improved considerably solving methods for TOP, a new strict improvement on one instance was detected and the newly attained relative error for all instances being 0.0005%. This success is due to the new accelerated split procedure, the good design of the recombination operator to update particle positions, the introduction of extra positions to the swarm, as well as the appropriate management of dynamic parameters. In summary, the results presented in this chapter are encouraging for the application of Particle Swarm Optimization to solve combinatorial problems, as already indicated in [9] and for the application/acceleration of optimal split procedures in dealing with vehicle routing problems, as already indicated in [48].

Table 3.7: Results for set 4 of the benchmark.

Instance	Z_{best}	MA10			PSOMA			PSOiA		
		Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}
p4.2.a	206	206	206	13.71	206	206	0.21	206	206	5.88
p4.2.b	341	341	341	51.8	341	341	0.65	341	341	39.21

continued on next page

Table 3.7 – continued

Instance	Z_{best}	MA10			PSOMA			PSOiA		
		Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}
p4.2.c	452	452	452	83.16	452	452	2.05	452	452	67.12
p4.2.d	531	531	530.7	143.67	531	530.8	29.77	531	531	124.29
p4.2.e	618	618	616.8	205.89	618	618	25.23	618	618	197.94
p4.2.f	687	687	679.6	204.92	687	681.4	109.12	687	687	322.64
p4.2.g	757	757	756	190.49	757	755.5	97.95	757	757	206.54
p4.2.h	835	835	828.1	245.31	835	826.1	126.94	835	833.6	257.24
p4.2.i	918	918	918	366.45	918	913.6	150.45	918	918	368.2
p4.2.j	965	965	962.6	300.78	965	963.1	167.88	965	965	258.36
p4.2.k	1022	1022	1020.7	370.53	1022	1020.2	180.83	1022	1021	350.07
p4.2.l	1074	1071	1070.8	281.38	1071	1066.9	160.98	1074	1072.3	357.41
p4.2.m	1132	1132	1129.4	303.22	1132	1129.9	201.95	1132	1130.6	321.08
p4.2.n	1174	1174	1172.7	374.27	1174	1170.3	158.34	1174	1172	427.5
p4.2.o	1218	1218	1216	306.18	1218	1211.7	154.13	1218	1210.9	415.43
p4.2.p	1242	1242	1241.2	311.23	1241	1238.4	198.04	1242	1239.5	347.47
p4.2.q	1268	1267	1264.3	313.03	1267	1264.6	192.02	1268	1266.2	588.22
p4.2.r	1292	1292	1288.7	352.5	1292	1287.8	178.19	1292	1289.9	470.01
p4.2.s	1304	1304	1301.8	297.88	1304	1302.1	190.02	1304	1303.8	486.19
p4.2.t	1306	1306	1306	292.66	1306	1306	167.27	1306	1306	408.65
p4.3.c	193	193	193	6.07	193	193	0.05	193	193	1.89
p4.3.d	335	335	335	29.92	335	335	1.08	335	335	16.6
p4.3.e	468	468	468	39.23	468	468	2.6	468	468	36.41
p4.3.f	579	579	579	107.55	579	579	6.3	579	579	72.88
p4.3.g	653	653	653	117.44	653	651.4	32.49	653	653	70.44
p4.3.h	729	728	724.7	138.95	729	724.7	60.24	729	729	194.18
p4.3.i	809	809	809	197.7	809	808.6	41.17	809	809	247.26
p4.3.j	861	861	859.5	175.98	861	857.6	85.71	861	860.9	229.11
p4.3.k	919	919	918.2	218.73	919	916.7	94.55	919	919	275.31
p4.3.l	979	979	975.2	206.99	979	977.4	89.28	979	976.9	258.33
p4.3.m	1063	1063	1057.9	231.16	1063	1058.4	103.42	1063	1062	281.42
p4.3.n	1121	1121	1115.9	197.23	1121	1115.6	104.57	1121	1118.4	309.03
p4.3.o	1172	1172	1169	306.3	1172	1169.7	145.44	1172	1172	371.72
p4.3.p	1222	1222	1219.4	301.81	1222	1222	123.77	1222	1222	284.61
p4.3.q	1253	1253	1250	220.44	1253	1250.2	112.58	1253	1252.2	448.69
p4.3.r	1273	1273	1270.2	218.38	1273	1269.5	115.4	1273	1269.4	288.72
p4.3.s	1295	1295	1293.8	255.25	1295	1291.5	115.6	1295	1289.5	278
p4.3.t	1305	1305	1303.7	210.95	1304	1301.1	124.38	1305	1304.3	305.85
p4.4.e	183	183	183	0.45	183	183	0.02	183	183	0.65
p4.4.f	324	324	324	16.5	324	324	0.2	324	324	8.61
p4.4.g	461	461	461	35.13	461	461	0.74	461	461	24.19
p4.4.h	571	571	571	52.23	571	567.1	5.56	571	571	36.74
p4.4.i	657	657	657	72.75	657	657	1.71	657	657	65.48

continued on next page

Table 3.7 – continued

Instance	Z_{best}	MA10			PSOMA			PSOiA		
		Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}
p4.4.j	732	732	732	95.06	732	731.2	9.2	732	732	81.35
p4.4.k	821	821	820.2	110.1	821	820.8	20.03	821	821	119.45
p4.4.l	880	880	879.1	113.9	880	879.1	54.34	880	879.5	101.6
p4.4.m	919	916	912.7	129.08	919	915.6	67.59	919	916.6	223.19
p4.4.n	977	969	965.5	197.89	969	964.2	72.07	976	967	257.14
p4.4.o	1061	1061	1057.7	185.94	1061	1051.6	91.63	1061	1060	208.36
p4.4.p	1124	1124	1119.8	226.38	1124	1115.9	102.16	1124	1122.7	193.14
p4.4.q	1161	1161	1161	195.51	1161	1159.4	94.86	1161	1161	252.98
p4.4.r	1216	1216	1210.2	226.09	1216	1201	94.76	1216	1206.2	260.06
p4.4.s	1260	1260	1256.9	191.25	1259	1257.1	131.53	1260	1257.5	256.15
p4.4.t	1285	1285	1283	175.01	1285	1284.5	100.62	1285	1282.3	161.33

Table 3.8: Results for set 5 of the benchmark.

Instance	Z_{best}	MA10			PSOMA			PSOiA		
		Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}
p5.2.h	410	410	410	50.52	410	410	1.34	410	410	51.98
p5.2.j	580	580	580	41.63	580	580	1.45	580	580	54.37
p5.2.k	670	670	670	41.12	670	670	1.63	670	670	60.78
p5.2.l	800	800	800	65.16	800	800	37.6	800	800	88.15
p5.2.m	860	860	860	62.88	860	860	33.52	860	860	90.5
p5.2.n	925	925	925	53.75	925	925	25.21	925	925	72.46
p5.2.o	1020	1020	1020	47.56	1020	1020	29.7	1020	1020	65.93
p5.2.p	1150	1150	1150	96.04	1150	1150	59.14	1150	1150	109.63
p5.2.q	1195	1195	1195	53.26	1195	1195	36.71	1195	1195	116.62
p5.2.r	1260	1260	1260	68.66	1260	1260	37.6	1260	1260	92.78
p5.2.s	1340	1330	1325	63.44	1340	1329.5	37.04	1340	1340	85.78
p5.2.t	1400	1400	1397	52.27	1400	1400	37.39	1400	1400	111.2
p5.2.u	1460	1460	1460	68.79	1460	1460	44.59	1460	1460	110.39
p5.2.v	1505	1505	1503.5	65.45	1505	1504.5	43.79	1505	1505	112.4
p5.2.w	1565	1560	1560	50.22	1560	1560	47.45	1565	1562.5	124.13
p5.2.x	1610	1610	1610	57.27	1610	1610	50.02	1610	1610	124.68
p5.2.y	1645	1645	1645	66.25	1645	1645	37.98	1645	1645	112.34
p5.2.z	1680	1680	1680	64.77	1680	1679	41.75	1680	1680	122.55
p5.3.k	495	495	495	30.34	495	495	1.36	495	495	33.27
p5.3.l	595	595	595	39.81	595	595	1.25	595	595	53.91
p5.3.n	755	755	755	41.9	755	755	1.87	755	755	48.68
p5.3.o	870	870	870	34.7	870	870	2.13	870	870	48.93
p5.3.q	1070	1070	1070	49.38	1070	1070	23.28	1070	1070	51.54
p5.3.r	1125	1125	1125	43.97	1125	1125	22.68	1125	1125	46.7

continued on next page

Table 3.8 – continued

Instance	Z_{best}	MA10			PSOMA			PSOiA		
		Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}
p5.3.s	1190	1190	1189	41.16	1190	1189	26.6	1190	1190	59.48
p5.3.t	1260	1260	1260	54.36	1260	1260	32.65	1260	1260	69.12
p5.3.u	1345	1345	1345	51.74	1345	1345	26.21	1345	1345	57.97
p5.3.v	1425	1425	1425	48.05	1425	1425	29	1425	1425	56.21
p5.3.w	1485	1485	1481.5	44.83	1485	1477	36.32	1485	1484.5	76.2
p5.3.x	1555	1555	1547.5	50.42	1555	1546.5	36.79	1555	1552	76.78
p5.3.y	1595	1590	1590	54.95	1595	1590.5	30.46	1595	1591	70.54
p5.3.z	1635	1635	1635	61.91	1635	1635	32.64	1635	1635	84.24
p5.4.m	555	555	555	22.29	555	555	1.46	555	555	20.76
p5.4.o	690	690	690	33.08	690	690	1.69	690	690	42.91
p5.4.p	765	760	760	46.44	765	760.5	27.36	765	760.5	48.35
p5.4.q	860	860	860	50.91	860	860	1.53	860	860	61.58
p5.4.r	960	960	960	64.85	960	960	1.39	960	960	76.4
p5.4.s	1030	1030	1029.5	39.19	1030	1029.5	18	1030	1030	30.82
p5.4.t	1160	1160	1160	48.1	1160	1160	2.09	1160	1160	47.63
p5.4.u	1300	1300	1300	66.82	1300	1300	2.1	1300	1300	67.26
p5.4.v	1320	1320	1320	37.87	1320	1320	1.77	1320	1320	97.97
p5.4.w	1390	1380	1380	29.24	1385	1381	19.37	1390	1386	40.5
p5.4.x	1450	1450	1450	47.34	1450	1448	25.75	1450	1450	63.41
p5.4.y	1520	1520	1520	46.84	1520	1520	32.06	1520	1520	102.12
p5.4.z	1620	1620	1620	50.43	1620	1620	37.84	1620	1620	86.55

Table 3.9: Results for set 6 of the benchmark.

Instance	Z_{best}	MA10			PSOMA			PSOiA		
		Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}
p6.2.d	192	192	192	11.37	192	192	0.14	192	192	5.76
p6.2.j	948	948	945.6	54.68	948	948	27.8	948	948	54.1
p6.2.l	1116	1116	1116	50.3	1116	1116	34.75	1116	1116	66.31
p6.2.m	1188	1188	1188	38.06	1188	1188	28.62	1188	1188	58.34
p6.2.n	1260	1260	1260	34.66	1260	1260	21.82	1260	1260	62.26
p6.3.g	282	282	282	8.3	282	282	0.17	282	282	5.85
p6.3.h	444	444	444	19.78	444	444	0.71	444	444	12.34
p6.3.i	642	642	642	33.84	642	642	1.3	642	642	31.16
p6.3.k	894	894	894	34.05	894	894	2.05	894	894	61.6
p6.3.l	1002	1002	1002	39.3	1002	1002	4.42	1002	1002	50.91
p6.3.m	1080	1080	1080	30.86	1080	1080	21.27	1080	1080	56.64
p6.3.n	1170	1170	1170	30.85	1170	1170	1.61	1170	1170	52.52
p6.4.j	366	366	366	7.28	366	366	0.18	366	366	5.23
p6.4.k	528	528	528	13.51	528	528	0.6	528	528	8.56

continued on next page

Table 3.9 – continued

Instance	Z_{best}	MA10			PSOMA			PSOiA		
		Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}
p6.4.l	696	696	696	18.96	696	696	1.12	696	696	27.37

Table 3.10: Results for set 7 of the benchmark.

Instance	Z_{best}	MA10			PSOMA			PSOiA		
		Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}
p7.2.d	190	190	190	12.81	190	190	0.11	190	190	4.36
p7.2.e	290	290	290	30.15	290	290	0.57	290	290	18.56
p7.2.f	387	387	386.4	59.79	387	386.4	7.86	387	387	40.48
p7.2.g	459	459	459	106.42	459	459	49.73	459	459	126.9
p7.2.h	521	521	520.6	113.88	521	521	32.42	521	521	188.66
p7.2.i	580	580	579.4	151.09	580	579.1	82.32	580	579.9	181.96
p7.2.j	646	646	646	159.48	646	645.3	54.9	646	646	134.9
p7.2.k	705	705	703.7	182.99	705	704.2	70.42	705	704.7	170.92
p7.2.l	767	767	767	205.35	767	767	73.7	767	767	210.89
p7.2.m	827	827	827	203.14	827	827	104.68	827	827	177.68
p7.2.n	888	888	887.9	261.33	888	887.6	142.88	888	888	186.65
p7.2.o	945	945	945	221.07	945	945	109.36	945	945	208.93
p7.2.p	1002	1002	1001.6	274.36	1002	999.2	103.42	1002	1001.8	241.83
p7.2.q	1044	1044	1043.8	247.36	1044	1039.2	130.83	1044	1043.7	181.26
p7.2.r	1094	1094	1094	232.16	1094	1091.3	126.52	1094	1094	182.32
p7.2.s	1136	1136	1136	258.83	1136	1134.5	127.36	1136	1136	228.1
p7.2.t	1179	1179	1176.3	280.06	1179	1174.1	157.24	1179	1179	277.18
p7.3.h	425	425	424.8	43.6	425	424.2	3.62	425	425	27.88
p7.3.i	487	487	487	70.87	487	487	5.96	487	487	45.65
p7.3.j	564	564	563.6	98.35	564	562.7	30.49	564	564	54.98
p7.3.k	633	633	632.7	130.31	633	632	56.7	633	633	88.79
p7.3.l	684	684	682.1	112.98	683	682	50.14	684	684	100.72
p7.3.m	762	762	762	158.38	762	760.9	64.26	762	762	127.04
p7.3.n	820	820	820	164.08	820	817.6	114.61	820	820	175.64
p7.3.o	874	874	871	173.12	874	872.5	84.87	874	874	196.91
p7.3.p	929	929	926	165.55	927	924.8	73.3	929	928	162.61
p7.3.q	987	987	987	205.16	987	980.6	107.93	987	987	168.7
p7.3.r	1026	1026	1022.4	218.64	1026	1021.7	101.17	1026	1022.6	203.02
p7.3.s	1081	1081	1079.7	245.28	1081	1079.5	123.09	1081	1081	242
p7.3.t	1120	1120	1118.7	266.55	1120	1118.2	138.14	1120	1118.4	151.73
p7.4.g	217	217	217	10.06	217	217	0.08	217	217	1.72
p7.4.h	285	285	285	13.28	285	285	0.16	285	285	4.44
p7.4.i	366	366	366	29.69	366	366	0.46	366	366	12.68
p7.4.k	520	520	518.4	61.08	520	518.2	14.42	520	518.2	39.94

continued on next page

Table 3.10 – continued

Instance	Z_{best}	MA10			PSOMA			PSOiA		
		Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}	Z_{max}	Z_{avg}	CPU_{avg}
p7.4.l	590	590	587.9	80.05	590	588.4	19.85	590	590	53.8
p7.4.m	646	646	646	109.07	646	646	33.59	646	646	100.05
p7.4.n	730	726	726	113.62	726	725.9	37.48	730	728.8	110.63
p7.4.o	781	781	779.4	133.82	781	779.3	54.97	781	780.4	93.27
p7.4.p	846	846	843.7	167.49	846	841.4	62.24	846	846	124.32
p7.4.q	909	909	907	165.07	909	907	77.59	909	908.7	134.35
p7.4.r	970	970	970	147.92	970	970	65.89	970	970	143.04
p7.4.s	1022	1022	1020.8	170.42	1022	1020.7	62.23	1022	1022	150.72
p7.4.t	1077	1077	1077	180.48	1077	1077	88.03	1077	1077	128.13

Table 3.11: Results of the new instances.

Instance	n	m	L	gen	PSOiA		
					Z_{max}	Z_{avg}	CPU_{avg}
cmt101c_m3	100	3	126.33	$gen2$	1300	1299	111.109
cmt151b_m3	150	3	116.67	$gen2$	1385	1373.8	754.007
cmt151c_m2	150	2	262.5	$gen2$	1963	1962	1799.641
cmt151c_m3	150	3	175	$gen2$	1916	1909.1	1376.236
cmt151c_m4	150	4	131.25	$gen2$	1880	1875.6	881.114
cmt200b_m2	199	2	191	$gen2$	2096	2088.2	4180.987
cmt200b_m3	199	3	127.33	$gen2$	2019	2005	2711.659
cmt200b_m4	198	4	95.5	$gen2$	1894	1889.7	1515.185
cmt200c_m2	199	2	286.5	$gen2$	2818	2810.1	7320.261
cmt200c_m3	199	3	191	$gen2$	2766	2751.2	4217.286
cmt200c_m4	199	4	143.25	$gen2$	2712	2700.6	3004.103
eil101b_m3	100	3	105	$gen2$	916	913.8	134.388
eil101c_m2	100	2	236	$gen2$	1305	1304.8	452.79
eil101c_m3	100	3	157.33	$gen2$	1251	1244.1	227.613
gil262a_m2	241	2	297.5	$gen2$	4078	4056.4	5907.285
gil262a_m4	112	4	148.75	$gen2$	3175	3174.2	271.83
gil262b_m2	249	2	594.5	$gen2$	8081	8061.1	7473.182
gil262b_m3	249	3	396.33	$gen2$	7585	7574.9	7276.798
gil262b_m4	241	4	297.25	$gen2$	6781	6742	4878.641
gil262c_m2	249	2	892	$gen2$	11030	11020	27500.87
gil262c_m3	249	3	594.67	$gen2$	10757	10714.6	14553.762
gil262c_m4	249	4	446	$gen2$	10281	10259.4	8472.009
bier127_gen1_m2	126	2	29570.5	$gen1$	106	104.8	1153.874
bier127_gen1_m3	126	3	19713.7	$gen1$	103	102.4	591.889
bier127_gen2_m2	126	2	29570.5	$gen2$	5464	5446.8	1132.566
bier127_gen2_m3	126	3	19713.7	$gen2$	5393	5376.2	648.084

continued on next page

Table 3.11 – continued

Instance	n	m	L	gen	PSOiA		
					Z_{max}	Z_{avg}	CPU_{avg}
bier127_gen2_m4	120	4	14785.2	$gen2$	5122	5119.2	657.569
bier127_gen3_m2	126	2	29570.5	$gen3$	2885	2884.3	1301.269
bier127_gen3_m3	126	3	19713.7	$gen3$	2706	2703.8	711.736
bier127_gen3_m4	120	4	14785.2	$gen3$	2402	2384.6	680.789
gil262_gen1_m3	210	3	396.333	$gen1$	101	100.9	1769.314
gil262_gen1_m4	102	4	297.25	$gen1$	78	77.1	155.755
gil262_gen2_m2	261	2	594.5	$gen2$	7498	7457.8	7356.652
gil262_gen2_m3	210	3	396.333	$gen2$	5615	5608.2	3304.551
gil262_gen3_m2	261	2	594.5	$gen3$	7183	7182.8	9129.303
gil262_gen3_m4	102	4	297.25	$gen3$	2507	2499.8	276.424
gr229_gen1_m4	227	4	441.25	$gen1$	223	220.8	11922.016
gr229_gen2_m3	228	3	588.333	$gen2$	11566	11551.3	14197.206
gr229_gen2_m4	227	4	441.25	$gen2$	11355	11255.5	18799.5
gr229_gen3_m3	228	3	588.333	$gen3$	8056	8051.6	14090.055
gr229_gen3_m4	227	4	441.25	$gen3$	7621	7600	11399.708
kroA150_gen2_m2	149	2	6631	$gen2$	4335	4334.4	892.981
kroA150_gen3_m3	127	3	4420.67	$gen3$	2726	2719.6	538.011
kroA200_gen1_m4	132	4	3671	$gen1$	81	80.4	560.285
kroB200_gen1_m2	199	2	7359.5	$gen1$	111	110.4	2344.534
kroB200_gen2_m2	199	2	7359.5	$gen2$	6185	6182.2	3467.258
kroB200_gen2_m4	128	4	3679.75	$gen2$	4944	4942.2	640.664
kroB200_gen3_m2	199	2	7359.5	$gen3$	4765	4757.8	6306.618
kroB200_gen3_m3	157	3	4906.33	$gen3$	3028	3016	1713.877
lin318_gen1_m2	317	2	10522.5	$gen1$	180	170.1	20667.243
lin318_gen1_m3	256	3	7015	$gen1$	149	148.6	9014.643
lin318_gen2_m2	317	2	10522.5	$gen2$	9544	9533.8	23804.82
lin318_gen2_m3	256	3	7015	$gen2$	7786	7782.1	9773.63
lin318_gen3_m2	317	2	10522.5	$gen3$	7936	7905.6	44029
lin318_gen3_m4	154	4	5261.25	$gen3$	3797	3796.4	1446.258
pr136_gen1_m2	131	2	24193	$gen1$	63	62.7	451.134
pr136_gen2_m2	131	2	24193	$gen2$	3641	3631.8	601.312
pr264_gen1_m4	118	4	6142	$gen1$	107	106.6	503.071
pr264_gen2_m2	131	2	12284	$gen2$	6635	6634.2	2048.2
pr264_gen2_m3	131	3	8189.33	$gen2$	6420	6410.7	938.394
pr264_gen2_m4	118	4	6142	$gen2$	5584	5564.5	590.787
pr264_gen3_m3	131	3	8189.33	$gen3$	2772	2770	1037.505
pr299_gen1_m2	251	2	12048	$gen1$	139	138.5	4775.928
pr299_gen1_m3	162	3	8032	$gen1$	111	110.1	1303.726
pr299_gen1_m4	112	4	6024	$gen1$	84	83.6	383.479
pr299_gen2_m3	162	3	8032	$gen2$	6018	5966.7	1446.047
pr299_gen2_m4	112	4	6024	$gen2$	4457	4453	593.41

continued on next page

Table 3.11 – continued

Instance	n	m	L	gen	PSOiA		
					Z_{max}	Z_{avg}	CPU_{avg}
pr299_gen3_m2	251	2	12048	$gen3$	5729	5728.6	11872.546
pr299_gen3_m3	162	3	8032	$gen3$	3655	3611	2705.815
pr299_gen3_m4	112	4	6024	$gen3$	2268	2258	455.639
rat195_gen2_m2	190	2	581	$gen2$	5148	5145.6	2156.983
rat195_gen3_m3	122	3	387.333	$gen3$	2574	2571.2	721.818
ts225_gen2_m2	217	2	31661	$gen2$	5859	5858.5	2998.427
rd400_gen2_m2	399	2	3820.5	$gen2$	12993	12787.5	77049.22
rd400_gen2_m3	399	3	2547	$gen2$	12645	12372.1	53707.14
rd400_gen2_m4	399	4	1910.25	$gen2$	12032	11953.5	42001.58
rd400_gen1_m2	399	2	3820.5	$gen1$	230	227.8	56767.29
rd400_gen1_m3	399	3	2547	$gen1$	222	221.7	62476.08
rd400_gen1_m4	399	4	1910.25	$gen1$	213	210.6	34744.8
rd400_gen3_m2	399	2	3820.5	$gen3$	12428	12274.1	96178.7
rd400_gen3_m3	399	3	2547	$gen3$	11639	11629.5	68074.77
rd400_gen3_m4	399	4	1910.25	$gen3$	10417	10383.1	48462.77
ARPE/CPUavg						0.46	11031.04

Algorithme mémétique pour le TOPTW

Abstract

The Team Orienteering Problem (TOP) is a variant of the vehicle routing problem. It is recently studied by many researchers and is described as follows: For a given set of vertices, each one associated with a score, the goal of the TOP is to maximize the sum of the scores collected by a fixed number of vehicles. More particularly, the Team Orienteering Problem with Time Windows (TOPTW) imposes the period of time of customer availability as a constraint to assimilate the real world situations. Split procedures have proved their efficiency for routing problems. Embedded in metaheuristics, their aim is to build a feasible routing solution by splitting a giant tour into trips. The Memetic Algorithm (MA) is population-based heuristic search approach that has been shown to be very effective for several combinatorial optimization problems. In this chapter, we present a memetic algorithm for TOPTW based on the application of split strategy to evaluate an individual. The effectiveness of the proposed MA is shown by many experiments conducted on benchmark problem instances available in the literature. The computational results indicate that the proposed algorithm competes with the heuristic approaches present in the literature and improves several best known solutions.

4.1 Introduction

Today, the significant increase in the energy prices is causing many problems to the companies, specially regarding the transportation of people, of goods or of information. These problems are commonly denoted as routing problems. One of the most challenging problems in logistics management is the distribution of finished products from depots to customers. The distribution problem is generally formulated as the vehicle routing problem (VRP). In the VRP, a fleet of vehicles is used to distribute the products from the depot to the customers. Each vehicle is referred to as a route, and each customer must be served in one route only. When operational requirements limit the total length of the route, decision makers must select the set of potential customers who are most convenient for him. In the

team orienteering problem (TOP), we are given a transportation network in which a start point and an end point are specified. Other points have associated scores and between all pairs of customers or between a customer and a depot we have a travel time. Given a fixed amount of time, the goal is to determine a path from start to end through a subset of locations in order to maximize the total path score. In the more general team orienteering problem with time windows (TOPTW) the vehicles have to comply with the additional constraint that serving a customer can only be started within its time window. It is legal to arrive before a time window "opens" but the vehicle must wait. This kind of time window is usually called a crisp or hard time window. In the next section, we briefly review the literature on OPTW and TOPTW.

4.2 Literature review

The Orienteering Problem (OP) was firstly introduced by Tsiligirides [128]. The roots of this problem trace back to the pioneering work of Golden et al. [64] who proved that the OP is NP-hard and used it to formulate and solve the home fuel delivery problem. The name "Orienteering Problem" originates from the sport game of orienteering described in [33]. Later, a new variant of the problem called Team Orienteering Problem (TOP) was introduced since it is widely seen in many real life situations, like for example the mobile tourist guide [132], the routing of technicians [24, 123] and fuel delivery problems [64].

Many heuristics have been successfully applied to TOP. There are four methods that can be considered as the state-of-the-art algorithms in the literature: a variable neighborhood search proposed by Archetti et al. [4], a memetic algorithm [25] , a path relinking approach [119] and a PSO-based memetic algorithm [43]. The survey of Vansteenwegen et al. [133] gives a review of the most important contributions on the orienteering literature.

Recently, The Orienteering Problem with Time Windows (OPTW) and the Team Orienteering Problem with Time Windows (TOPTW) have been the interest of many researchers. They are considered as the generalization of the OP and the TOP with the additional time constraints. In these problems, the service of a customer must be started within a time window $[e_i; l_i]$ defined by customer i. The vehicle cannot arrive earlier than time e_i and no later than time l_i. Vehicle arriving earlier than the earliest service time of a customer will incur waiting time. The first who considered the time windows in the OP were Kantor and Rosenwein [77]. They solved the problem with a tree heuristic that was more efficient than the classical insertion heuristics. The only exact method that we found was developed by Righini and Salani [114]. The computational time required by their method to solve large problem instances was very expensive. Therefore, most of the researchers focus on developing approximate methods. Montemanni and Gambardella [97] used an ant colony optimization to solve the problem, while Vansteenwegen et al. [130] present an iterated local search metaheuristic. In this method, an insert step is combined with a shake step to explore the research space more efficiently. Tricoire et al. [127] defined

the Multi-Period Orienteering Problem with Multiple Time Windows (MuPOPTW) as a new problem for scheduling the customer visits of sales representatives. The MuPOPTW is a generalization of OPTW and TOPTW, where customers may be visited on different time slots, and may have several time windows for each time slot. They propose an exact algorithm embedded in a variable neighborhood search method and provide experimental results for their method on standard benchmark of OPTW and TOPTW instances. Lin and Yu [95] presented a simulated annealing based heuristic approach to solve TOPTW. The method proposed by Labadie et al. [85] combines the greedy randomized adaptive search procedure (GRASP) with the evolutionary local search (ELS). ELS generates multiple distinct child solutions that are further improved by a local search procedure, while GRASP provides multiple starting solutions to ELS. Labadie et al. [86] introduced granular variant to a VNS algorithm in order to improve its efficiency. Firstly, each arc is evaluated with new cost taken into account traveling times, waiting times and profits. Then, an assignment problem is optimally solved and intervals of granularity are created. These intervals determine subset of promising arcs which will be considered during the node sequences construction in the local search procedure.

Memetic algorithms are known to be an effective approach in solving many hard combinatorial optimization problems [67]. Typically, a memetic approach repeatedly alternates between a recombination (or crossover) operator and a local optimization procedure to generate solutions located in promising regions in the search space. Many successful memetic approaches employ indirect representations of solutions. In order to be effective, an efficient decoder algorithm must be applied to extract one or several complete solutions. This widely applied methodology is in itself a structural problem decomposition.

In this chapter, a metaheuristic-based memetic algorithm (MA) is presented for the TOPTW. The proposed MA works with permutation encoding and uses adapted procedure to optimally split a sequence into a set of routes. The rest of the article is organized as follows. The next section is devoted to the formulation of TOPTW. Section 4.4 presents the detailed description of the proposed method including the solution representation, the optimal split procedure, and other components and parameters. In section 4.5, the effectiveness of the proposed algorithm is demonstrated by many computational results based on some benchmark problems. The conclusions are discussed in the final section 4.6.

4.3 Formulation of the problem

TOPTW is modeled with a graph $G = (V \cup \{d\}, E)$, where $V = \{1, 2, ..., n\}$ is the set of vertices representing customers, $E = \{(i, j) \mid i, j \in V\}$ is the edge set and d represents the depot that is the departure and arrival vertice for the vehicles. Each vertex i is associated with a profit P_i and a service time T_i. The visit of a vertex i can start only within a predefined time window $[e_i, l_i]$. The visitor cannot arrive later than the time l_i and in case he arrives earlier than e_i, he must wait before the service can start. For the depot vertex d, e_0 represents the earliest time

the visitor can leave the starting point and l_0 is the latest possible arrival time to the depot. Each edge $(i, j) \in E$ is associated with a travel cost $C_{i,j}$ which is assumed to be symmetric and satisfying the triangle inequality. A tour R is represented as an ordered list of q customers from V, so $R = (R[1], \ldots, R[q])$. Each *tour* begins and ends at the depot vertex. We denote the total profit collected from a tour R as $P(R) = \sum_{i=1}^{i=q} P_{R[i]}$, and the total travel cost or duration $C(R)$. A tour R is feasible if $C(R) \leq l_0$, l_0 being a latest possible arrival time to the depot, and if each customer is serviced within its time window. The fleet is composed of m identical vehicles. A *solution* S is consequently a set of m (or fewer) feasible tours in which each customer is visited at most once. The goal is to find a solution S such that $\sum_{R \in S} P(R)$ is maximized. One simple way of reducing the size of the problem is to consider only *accessible* customers. A customer is said to be accessible if a tour containing only this customer has a travel cost/time less than or equal to l_0 without violating its time window. For mixed integer linear programming formulations of TOPTW see [97] and [130].

4.4 Memetic algorithm

Memetic algorithm is a combination of an evolutionary algorithm and neighborhood-based local search framework [98]. The basic idea behind memetic approaches is to combine the advantages of the crossover that discovers unexplored promising regions of the search space, and local optimization that finds good solutions by concentrating the search around these regions. The proposed memetic algorithm is based on permutation encoding. The key feature of the proposed method is the split procedure that allows a reduction of the solution space exploration within the global optimization. We introduce an interesting way to represent solutions of TOPTW, known as giant tours. Each giant tour is in fact a neighborhood of solutions in the search space from which the optimal associated solution can be easily extracted by an evaluation process. Therefore, a heuristic using this representation tends to have better visions during the search and a better chance to reach the global optimum. Next, all the details of MA implementation are presented.

4.4.1 Chromosome and evaluation

The representation of our chromosome consists of an ordered list of all accessible customers in V called a *giant tour*. The giant tour is a permutation of n positive integers, such that each integer corresponds to a customer without trip delimiters. We try to extract m tours from the giant tour while respecting the order of the customers in the sequence. A tour from a permutation π is identified by its starting point i in the sequence and the number of customers following i , denoted l_i. A chromosome is evaluated using a tour splitting procedure which optimally partitions π into feasible routes. Using this strategy, the MA searches the set of possible giant tours to find one that gives an optimal solution after splitting.

Bouly et al. [25] proposed an optimal splitting procedure which is specific to the TOP. In their method, all possible subsequences of the *giant tour* that can form a feasible tour of TOP are considered. This extracted tours are used to build an auxiliary graph. Then, PERT/CPM algorithm is applied to look for the longest path under the cardinality constraint $2m+1$. This is efficiently addressed with a labeling technique. The value of the longest path found by this procedure corresponds to the highest profit that can be reached. The *worst case* complexity of their procedure is $O(m \cdot n^2)$.

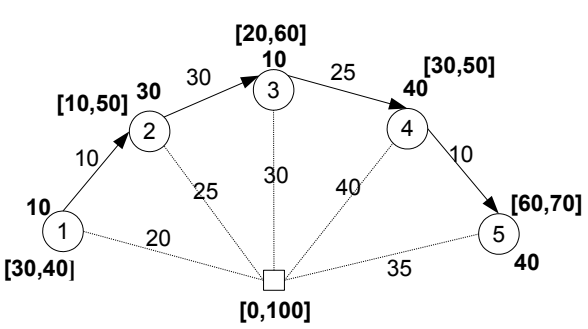
In [25], only tours of maximum length are considered. Meaning that all customers following i in the sequence are included in the tour as long as all constraints are satisfied, or until the end of the sequence is reached. Such a tour is called *saturated* tour. They proved that solutions containing only saturated tours are dominant. Therefore, only saturated tours were considered in their procedure. Later, Dang et al. [44] introduce a new evaluation procedure in which the limited number of saturated tours is exploited more efficiently to reduce the complexity of the evaluation process. In this work, the split procedure problem is reduced to a particular knapsack problem with conflicts (KPCG). KPCG, is a well-known combinatorial optimization problem [137]. It models a situation analogous to filling a knapsack with all or part of a given set of objects each having a weight and value, without exceeding the total weight limit supported by the knapsack. In addition to the knapsack capacity, some items are in conflict with each other and cannot be put in the knapsack together. The aim of the knapsack problem is to determine the objects selected to be placed in the knapsack in order to maximize the total value without exceeding the maximum weight. In such a problem, the conflicts between items are usually modeled with a graph, called *conflict graph*.

Before reviewing the main idea of the split procedure, we recall the definition of an interval graph [124] as follows. A graph $G = (V, E)$ is an interval graph if there is a mapping I between the vertex set of G and a collection of intervals in the real line such that two vertices of G are adjacent if their respective intervals intersect. Then, for all i and j of V, $[i, j] \in E$ if and only if $I(i) \cap I(j) \neq \emptyset$.

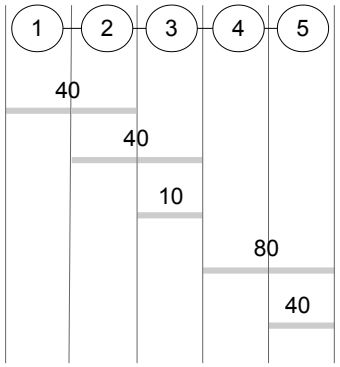
Proposition 4.4.1. *Given a TOPTW instance where m is the maximum number of available vehicles and π a permutation of n customers, the split procedure of π can be done optimally in $O(m \cdot n)$ time and space.*

Proof. The set of extracted tours from a giant tour can be mapped to the set of vertices of an interval graph X. An edge in X indicates the presence of shared customers between the associated tours. Since a split procedure looks for m tours without any shared customer such that the sum of their profit is maximized. So this is equivalent to solve a knapsack problem where X represents the conflict graph, m is the knapsack's capacity, and the weight of each item is reduced to one. In this particular knapsack problem, the number of items is equal to the number of possible tours. This number is equal to n when only saturated tours are considered. Such a problem can be solved in $O(m \cdot n)$ time and space [116]. □

We have extended the split procedure for TOPTW to tackle time windows.



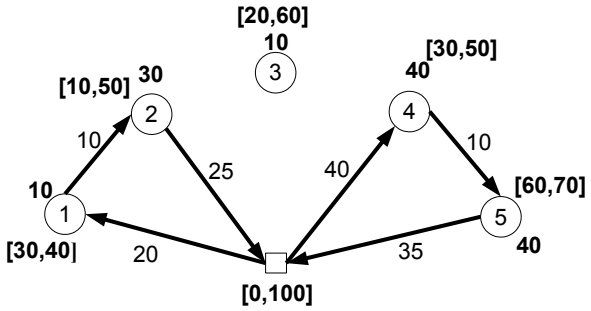
a) a splitting problem with $m = 2$



b) the weighted interval model

			Γ	
i	P(i)	*succ*(*i*)	1	2
1	40	3	**80**	**120**
2	40	4	**80**	**120**
3	10	4	**80**	90
4	80	-	**80**	0
5	40	-	40	0

c) the solving steps



d) a saturated optimal solution

Figure 4.1: An example of splitting problem.

Time windows affect only the construction of the interval model. When defining a saturated tour R starting with x, we should make sure that each customer is served within its time window and that $C(R) \leq l_0$, where l_0 is the latest possible arrival time to the depot and $C(R)$ the total travel duration. So, a waiting time is added each time the vehicle arrives at a customer before the beginning of his time windows. If the vehicle arrives too late, which is not allowed, the subsequence is not included in X. The evaluation process is summarized as follows. For each saturated tour starting with customer $\pi[i]$, we use $P[i]$ to denote the sum of profits collected from its customers. Its *first successor*, denoted by $succ[i]$, is computed as follows:

$$succ[i] = \begin{cases} i + l_i^{max} + 1 & \text{if } i + l_i^{max} + 1 \leq n \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

A two-dimensional array Γ of size $m \cdot n$ is used to memorize the maximum reachable profit during process. The algorithm then browses the saturated tours in reversed

order, or in other words, from customer $\pi[n]$ to customer $\pi[1]$, and updates Γ based on the recurrence relation described in Equation 4.2.

$$\Gamma(i,j) = \begin{cases} \max\{\Gamma(succ[i], j-1) + P[i], \Gamma(i+1, j)\} & \text{if } 1 \leq i \leq n \text{ and } 1 \leq j \leq m \\ 0 & \text{otherwise} \end{cases} \quad (4.2)$$

At the end, $\Gamma(1, m)$ corresponds to the profit of the optimal solution. A simple backtrack is then performed on Γ in order to determine the corresponding tours. That is to say if $\Gamma(succ[i], j-1) + P[i]$ is used over $\Gamma(i+1, j)$ in the relation 4.2, then the saturated tour starting with customer $\pi[i]$ belongs to the optimal solution.

The first graph of Figure 4.1 shows a sequence $S = (1, 2, 3, 4, 5)$ where each customer has a profit and a time window given in the square brackets. To simplify, we assume that the service times are set to 0, the number of the vehicles m used is equal to 2, and the maximum operation time l_0 is 100. The interval model is given in the second part of the Figure 4.1. The first interval $[1, 2]$ for example with weight 40 corresponds to the collected profit of the trip $(d, 1, 2, a)$. Vehicle leaves the depot at time 0, waits 10 units of time at node 1 before leaving it to serve node 2 at time 40. The customer 3 cannot be included in the trip, since its time window is already closed when a vehicle reaches it at time 70. The other intervals $[i, j]$ of the graph are similarly defined. The details of the first successor relations as well as the solving steps are given in the Figure 4.1.c The maximum score obtained with two tours is equal to 120. Finally, we gives the optimal solution obtained by the algorithm in Figure 4.1.d. It is composed of two tours starting respectively with customer 1 and 4.

4.4.2 Population

To generate the initial population, 75 percent of the population is created by random permutations of n customers nodes. The remaining 25 percent is generated by a fast and simple heuristic procedure to reduce significantly the algorithm time. That's why, the heuristic method proposed by Bouly et al. [25], called Iterative Destruction/Construction Heuristic(IDCH), has been modified and applied to TOPTW. First, we build a feasible solution by inserting at every iteration an unrouted customer. This process is performed using Best Insertion Algorithm (BIA). In this algorithm, the insertion which provides the minimal increase in cost to the current solution, is accepted. To improve the initial solution constructed we use destruction/repair operator. Initially, IDCH removes a limited random number of customers $D \in (1, 2, 3)$ from the current solution. In order to reduce the travel cost of tours, 2-opt* exchanges are combined with Or-opt exchanges. These intra-route and inter-route exchanges were introduced in [110] as powerful heuristics for problems with time windows. In the 2-opt*, two clients i and j from different routes are selected. The exchange that involve links $(i, i+1)$ and $(j, j+1)$ is performed if the move maintains the feasibility of the solution and improves its travel cost. The Or-Opt considers a sequence of one, two or three consecutive customers in the

current solution, and moves the sequence at another location in the same route. In the next step, we rebuild the solution by re-inserting unrouted customers in all possible way. To ensure that constraints on insertion are performed in $O(1)$, we record for each already included customer i, its waiting time $Wait_i$ and the maximum delay allowed for the service $Maxshift_i$. All feasible insertions of each unserved customer u between two couple of adjacent customers i and j are evaluated. This is done according to a suitable cost function $C(u) = Shift_u/(P_u)^\alpha$ where $Shift_u = (C_{i,u} + Wait_u + T_u + C_{u,j} - C_{i,j})$. The feasible insertion that minimizes the cost is then processed. In addition, priority coefficient w_u is associated to each customer u. Whenever the customer is not routed through a construction phase its priority is increased by the value of its profit. The customer u that has a lager w_u is more likely to be inserted. When a limited number of iterations $iter_{perturb} = n$ is reached is reached without a strict improvement, a method of diversification is performed. Diversification stands for random moves that can deteriorate the current solution by removing a large number of customers $D_{perturb} \in [1, n/m]$. The destruction and construction phases are iterated until $iter_{max} = n^2$ iterations without improvement. The algorithm of the IDCH heuristic is given in Algorithm 4.1.

Algorithm 4.1: IDCH algorithm

```
Data: s empty solution;
V vector of n customers
Result: s* best solution found;
begin
    apply priority best insertion (s, V);
    iter ← 1 ;
    while iter ≤ iter_max do
        if iter modulo iter_pertub = 0 then
            do partial destruction(s, D_perturb) ;
        else
            do partial destruction(s, D) ;
        apply 2-Opt*/Or-Opt neighborhoods(s);
        U ← get unrouted clients (V, s);
        apply priority best insertion (s, U);
        if Profit(s) ≥ Profit(s*) then
            s* ← s;
            iter ← 1;
        else
            iter ← iter + 1 ;
```

4.4.3 Selection and crossover

The crossover operator is very important in a Genetic or a Memetic Algorithm. This operator allows the construction of new solutions from existing ones and plays a great part in the behavior of the algorithm. When selecting individuals, as parents, to produce the offspring, we favor those with higher solution quality and diversity. In this research, the binary tournament method [111] is adopted to select a couple of parents among the population. Two chromosomes are randomly selected in the population, and the chromosome with the best evaluation becomes the first parent. The tournament is repeated for the second parent. These parents are then combined using the linear order crossover or LOX [111]. The LOX first chooses two cut points randomly and passes the section enclosed by the cut points from one parent to child. Then, the unpassed customers are placed in the unfilled positions using the order of their occurrence in the other parent.

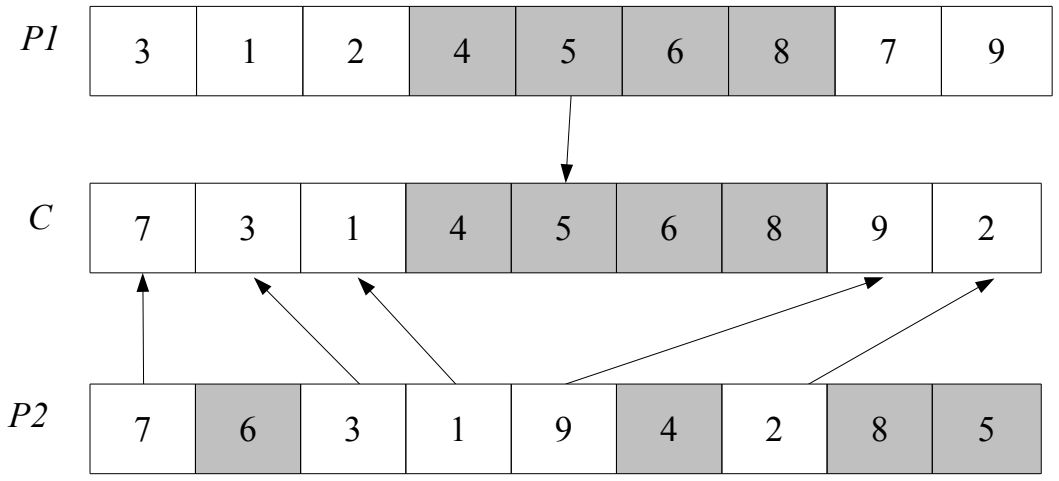


Figure 4.2: Illustration of the LOX crossover operator.

The example in Figure 4.2 shows how LOX constructs the child C. Given two parents $P1$ and $P2$ and assuming that the cut points are at positions $i = 4$ and $j = 6$, the subsequence $P1(i)...P1(j)$ is copied into $C(i)...C(j)$. Finally, $P2$ is swept from the beginning to complete C with the missing nodes.

4.4.4 Local search engine

The local search engine, which is the main component of memetic algorithms, indicates cultural evolution in the population. The local search procedures presented here are applied in order to improve the quality of every new child obtained from the crossover. We first present the set of local search operators included in the Memetic Algorithm, then we explain how they are managed.

- *2-opt* operator*: two routes r_1 and r_2 are devided into two parts. Then the first part of r_1 is connected to the second part of r_2, while the first part of r_2 is connected to the second part of r_1.

- *Or-opt operator*: consider a sequence of one, two or three consecutive customers in the the current solution, and move the sequence to another location in the same route.
- *destruction/repair operator*: first, a random number of customers (between 1 and $\frac{n}{m}$) is removed from an identified solution. Then, the lowest possible insertion cost $\frac{Shift_i}{(P_i)^\alpha}$ of each unrouted customer is evaluated. The visit with the lowest ratio will be selected for insertion.
- *shift operator*: a customer is removed from its current position and is relocated at another one. Every possible insertion position for every customer is considered.
- *swap operator*: positions of every two customers in the sequence are exchanged.

The call to the previous operators is controlled as follows. When a new child is computed with the crossover operator, the local search scheme is applied with a probability *pm*. Neighborhoods are selected in a random order. The research in a given neighborhood is stopped as soon as a better solution is found. Then a new neighborhood ,different from the previous one, is chosen randomly. This process is repeated until no further improvement can be made.

4.4.5 Population update

When an offspring solution s_{new} is created by the crossover operator presented in section 4.4.3 and improved by the local search algorithm described in section 4.4.4, we decide if the improved offspring should be inserted into the population and which existing solution of the population should be replaced. Basically, our decision is made based on both: the solution quality and the distance between solutions in the population. The update procedure is applied if and only if the performance of new solution s_{new} is better than the worst individual. Population is a sorted list of solutions according to two criteria: the total collected profit and the travel cost/time. Two solutions are said to be similar or identical if the evaluation procedure returns the same profit and a difference in travel cost/time lower than a value δ. If there is a solution s similar to s_{new}, then s is replaced with s_{new}. Otherwise the worst individual is deleted and the new solution is inserted into the population.

4.4.6 Basic algorithm

The proposed Memetic Algorithm associates all the elements described above. Algorithm 4.2 presents a synthetic view of the whole process. The algorithm starts with an initial set of solutions, called population. Each solution in the population is called a chromosome and represents a point in the search space. IDCH heuristic is used to generate a small part of a good initial solution while the remainder is generated randomly. During each iteration, two parents are selected and crossover operator is applied to create a new solution. The obtained child chromosome has a probability *pm* of being mutated using a set of LS techniques repeatedly. Finally, it is inserted within the population according to its fitness evaluation. The

Algorithm 4.2: Basic algorithm

```
Data: POP a population of N solutions;
Result: S_POP[N − 1] best solution found;
begin
    initialize and evaluate each solution in POP (see Section 4.4.2);
    while NOT(stoppingcondition) do
        select 2 parents POP[p1] and POP[p2] using binary tournament;
        C ← LOX(POP[p1], POP[p2]) ;
        if rand(0, 1) < pm then
            apply local search on C (see Section 4.4.4);
        if f(C) ≥ f(POP[0])(see Section 4.4.5) then
            if ∄p‖(f(POP[p]) = f(C)) then
                eject POP[0] from POP ;
                reset stopping condition ;
            else
                update stopping condition;
            insert or replace C in right place in POP ;
        else
            update stopping condition;
```

stopping criterion for MA is after reaching a maximal number of iterations without improvement. That is to say after reaching the number of iterations where the child chromosome simply replaces an existing chromosome in the population, or when its evaluation is worse than the worst chromosome in the current population.

4.5 Numerical results

We use instances that are taken from the http://www.mech.kuleuve - n.be/en/cib/op as test benchmarks to test our new proposed algorithm. 56 instances were designed by Solomon [118] and 20 others were designed by Cordeau et al. [38]. These instances originally designed for the VRPTW and the Multi Depot Periodic VRPTW(MDPVRPTW). They have been adapted by considering the demands of the customers as node profits and the number of tours $m \in \{1, 2, 3, 4\}$. Solomon's 100-customer instances are the most popular test set. They are organized into 6 classes: C1 and C2 are clustered instances, RC1 and RC2 are those where requests are partially clustered and partially random, and finally R1 and R2 have randomly distributed requests. Instances ending with 1 are based on narrow time windows and those ending with 2 are based on wide time windows. In the instances introduced by Cordeau, the number of vertices varies between 48 and 288 and all customers are considered active in the same day. Vansteenwegen et al. [130] use the original instances of Solomon and Cordeau to introduce a new set of benchmark instances. The number of vehicles considered allows to visit all customers that is why the

parameter	description	value
N	population size	40
N_{IDCH}	size of population generated with IDCH	5
k	stopping condition	10
pm	local search rate	$1 - \frac{iter}{itermax}$
δ	similarity measurement	0.01
α	control parameter of intuitive criteria	$1 + 2 \cdot \frac{r_1}{r_1+r_2}$, where r_1 and r_2 are two random numbers of BIA heuristic generated in $[0, 1]$ with a uniform distribution

Table 4.1: Parameter values

optimal solutions of these instances are known since they are equal to the sum of customers' profits. Travel time between two customers is assumed to be equal to the travel distance. It is rounded down to the first decimal for the Solomon's instances and to the second decimal for the Cordeau's instances. The whole algorithmic approach was implemented in C++ using the Standard Template Library (STL) for data structures and was compiled using the GNU GCC compiler on an AMD Opteron 2.60 GHz in a Linux environment.

4.5.1 Parameter setting

The parameters of the proposed algorithm are selected after several testing. A number of different alternative values were tested and the ones selected are those that gave the best computational results concerning both the quality of the solution and the computational time needed to achieve this solution. When the population is initialized, five chromosomes are generated by the IDCH heuristic and the rest are generated randomly. The similarity measurement of individuals δ is set to 0.01 and the local search rate pm is calculated as: $1 - \frac{iter}{itermax}$ where $iter$ is the number of consecutive iterations without improvement. The algorithm stops when $iter$ reaches $itermax = k * n/m$. The cost function $C(u) = Shift_u/(P_u)^\alpha$ of the BIA heuristic uses a random value of α generated in $[1, 3]$. This control parameter makes our IDCH less predictable and actually a randomized heuristic. Moreover, the score becomes more relevant than the time consumption when deciding which unrouted client is the most promising to insert. If α is set to 1, the obtained results are worse.

Finally only two parameters are required to be tunned, they are the stopping condition k and the population size N. We select the most representative subset of the problem characteristics (problem size, distribution of customer location, and time windows characteristic). Computational experiments were conducted on a subset of 40 instances which consists of 6 problems of the Solomon's instances and 4 problems the Cordeau's instances with $m = \{1, 2, 3, 4\}$. The value of k and N were varied from 10 up to 50 with steps of 10. This results 25 different (k, N)

settings to be tested. The algorithm was run 5 times on different randomly generated seeds for each instance. For an overall performance comparison between different configurations, we use two following measures. The first one is the relative gap to the best known solutions, denoted $rpe(\%)$ and the second is the average computational time in seconds CPU_{avg}. Details results for each parameter combination is reported in Table 4.2 and illustrated in Figure 4.3.

As there exist multiple Pareto optimal solutions, we calculate the *compromise point* wich gives a good trade off between algorithm performance and computational time. To identify this point, we introduce a fictitious point (also called *ideal point*) $P^0 = (0.06, 501.35)$. Then we determine the next best point P that minimizes the euclidean distance to P^0. Consequently we adopt the parameter settings $(10, 40)$ for computational experiments. The summary of selected parameters are given in Table 4.1.

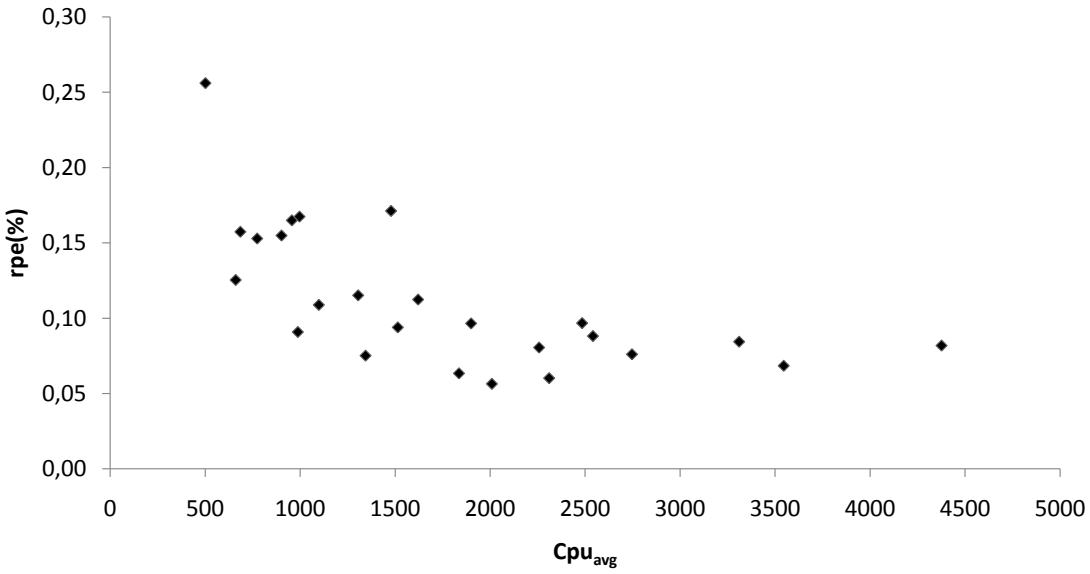


Figure 4.3: Pareto front solutions obtained with different settings of the stopping condition k and the population size N

4.5.2 Performance comparison

In order to investigate the performance of the proposed MA for TOPTW, we compare it with:

- Ant Colony System (ACS) introduced by [97], tested on a Dual AMD Opteron 250 2.4GHz processor,
- Iterated Local Search (ILS) proposed by [130], tested on Intel Core 2 with 2.5 GHz,
- Exact Algorithm embedded in a Variable Neighborhood Search (VNS) proposed by [127] , tested on 2.4GHz CPU,
- Simulated Annealing approach (SA) proposed by [95], tested on Intel Core 2 with 2.5 GHz CPU ,

k	N	$rpe(\%)$	CPU_{avg}
10	10	0.26	501.35
10	20	0.13	660.61
10	30	0.16	685.02
10	40	0.16	956.37
10	50	0.17	996.39
20	10	0.15	773.73
20	20	0.11	1098.18
20	30	0.09	1513.87
20	40	0.17	1478.26
20	50	0.11	1620.80
30	10	0.15	901.72
30	20	0.12	1305.28
30	30	0.10	1899.70
30	40	0.06	2009.10
30	50	0.10	2483.89
40	10	0.09	987.77
40	20	0.06	1835.80
40	30	0.06	2310.32
40	40	0.09	2540.97
40	50	0.07	3545.12
50	10	0.08	1344.04
50	20	0.08	2257.49
50	30	0.08	2746.46
50	40	0.08	3310.32
50	50	0.08	4375.72

Table 4.2: Performance on a small subset of instances with various parameters settings

Instance	ACS		ILS		VNS		GRASP-ELS		SA		GVNS		MA	
Set	rpe	cpu_{avg}	rpe	cpu_{avg}	rpe	cpu_{avg}	rpe	cpu_{avg}	rpe	cpu_{avg}	rpe	cpu_{avg}	rpe	cpu_{avg}
m=1														
c100	0	6.34	1.11	0.33	0	98.39	0	22.59	0	21.07	0.56	166.46	0	0.98
r100	0	383.40	1.90	0.19	0	89.10	0.11	3.51	0.11	23.34	1.72	29.43	0	5.38
rc100	0	143.21	2.92	0.23	0	65.21	0.33	1.99	0	22.19	1.88	9.80	0	1.59
c200	0.40	342.61	2.28	1.71	0	560.17	0.40	32.18	0.13	37.49	0.55	192.40	0	122.40
r200	2.18	1556.70	2.89	1.66	0.40	1065.82	0.59	11.18	1.29	45.83	2.44	33.82	−0.52	236.10
rc200	1.23	1544.55	3.43	1.63	0.07	869.41	1.37	8.21	0.96	50.25	2.53	16.01	−0.02	201.52
pr01-pr10	1.05	1626.61	4.72	1.75	0	822.07	0.73	5.03	0.97	112.21	0.54	12.37	−0.02	485.98
pr11-pr20	10.73	887.66	9.11	1.98	0.93	1045.93	1.70	7.90	3.25	162.40	2.71	24.22	0.39	903.08
m=2														
c100	0.15	818.00	0.94	1.08	0	87.98	0	70.94	0	26.42	0.47	139.53	0	70.09
r100	0.34	1559.36	2.27	0.87	0.06	63.46	0.92	7.97	0.14	36.63	1.10	60.34	−0.12	45.98
rc100	0.38	1375.78	2.47	0.71	0.23	55.16	1.46	4.66	0.19	40.48	0.78	20.31	0	46.33
c200	1.27	1398.10	2.54	3.46	0.51	545.65	0.09	29.26	1.18	53.66	0.25	33.79	0	164.93
r200	3.11	2735.15	2.69	2.27	0.20	1015.08	0.28	17.58	0.53	91.40	0.62	14.73	−0.57	634.67
rc200	2.64	2342.72	4.08	2.20	0.43	804.83	0.59	17.14	1.18	80.10	1.62	12.76	−0.60	355.97
pr01-pr10	2.35	1889.66	5.99	4.76	0.63	524.83	0.87	19.46	2.21	173.93	0.57	39.09	−0.44	1291.54
pr11-pr20	4.79	2384.81	7.65	5.21	1.04	618.78	2.21	28.77	3.66	201.63	0.98	82.44	−0.24	2144.27
m=3														
c100	0.11	1043.24	2.44	1.50	0	85.49	0.13	86.74	0.22	35.26	0.34	165.01	0	70.77
r100	0.55	1668.86	1.78	1.67	0.21	61.91	0.89	13.86	0.38	56.07	1.21	73.93	−0.01	58.56
rc100	1.19	1476.81	3.14	1.11	0.36	60.62	1.83	8.65	0.64	42.80	0.91	33.68	−0.01	54.72
c200	0.55	1413.11	1.98	2.08	0.16	196.80	0.45	26.75	0.35	53.93	0.64	55.42	−0.10	104.73
r200	0.13	1171.65	0.30	1.36	0.03	321.65	0	2.49	0.08	41.95	0.11	6.97	0	74.22
rc200	0.37	1607.85	1.37	1.73	0.04	404.01	0.06	8.34	0.20	58.98	0.25	7.41	−0.07	212.43
pr01-pr10	3.01	2163.80	6.57	9.24	1.50	473.20	1.31	40.55	2.33	197.01	0.35	85.90	−0.33	1416.21
pr11-pr20	5.19	2383.29	8.91	9.69	1.48	517.48	2.00	42.95	3.51	251.83	0.72	150.73	−0.71	2388.19
m=4														
c100	0.47	1056.05	2.93	2.57	0.09	81.87	0.50	84.58	0.36	49.51	0.85	133.22	−0.19	106.15
r100	0.99	1652.54	3.25	2.60	0.24	61.17	0.88	24.18	0.67	58.38	1.15	84.74	−0.11	79.46
rc100	0.92	1854.00	3.07	1.98	0.34	58.47	1.43	13.35	0.26	68.13	0.85	36.91	−0.24	57.66
c200	0	7.70	0	1.00	0	104.78	0	0.01	0	41.76	0	0.55	0	0.04
r200	0	126.46	0	0.87	0	150.74	0	0.03	0	39.71	0	0.27	0	0.10
rc200	0	646.72	0	1.24	0	164.56	0	0.03	0	40.15	0	0.88	0	0.15
pr01-pr10	2.34	2447.70	6.63	14.07	1.40	403.17	1.42	45.75	1.76	255.57	0.60	127.33	−1.12	1807.40
pr11-pr20	4.18	2583.50	7.16	13.74	0.90	408.01	1.20	65.33	2.57	283.98	0.64	232.64	−2.23	2784.70
Average	1.65	1401.79	3.38	3.09	0.36	375.62	0.74	22.60	0.96	88.30	0.87	64.34	−0.23	524.00

Table 4.3: Performance comparison based on RPE average for each data set of the standard benchmark.

Table 4.4: Strict improvements

Instance	m	New BK	Instance	m	New BK	Instance	m	New BK
r202	1	930	r111	3	774	pr03	3	1014
r203	1	1028	rc104	3	835	pr04	3	1298
r204	1	1093	r201	3	1450	pr05	3	1500
r206	1	1032	rc201	3	1699	pr06	3	1516
r207	1	1078	rc205	3	1719	pr08	3	1141
r208	1	1118	c103	4	1210	pr09	3	1282
r209	1	962	c108	4	1140	pr10	3	1586
r210	1	1002	r103	4	928	pr12	3	1004
r211	1	1051	r104	4	975	pr13	3	1159
rc202	1	938	r106	4	906	pr14	3	1376
rc204	1	1143	r107	4	950	pr15	3	1694
rc206	1	899	r108	4	995	pr17	3	842
rc208	1	1054	r111	4	953	pr18	3	1289
r104	2	552	r112	4	974	pr19	3	1433
r107	2	538	rc102	4	909	pr20	3	1722
r108	2	560	rc103	4	975	pr02	4	1083
c204	2	1490	rc104	4	1065	pr03	4	1247
r201	2	1260	rc107	4	987	pr04	4	1596
r202	2	1353	pr06	1	591	pr05	4	1859
r203	2	1430	pr11	1	353	pr06	4	1895
r205	2	1402	pr13	1	467	pr07	4	876
r206	2	1451	pr15	1	708	pr08	4	1392
r209	2	1423	pr02	2	715	pr09	4	1623
r210	2	1440	pr04	2	928	pr10	4	1967
rc201	2	1386	pr05	2	1103	pr12	4	1134
rc202	2	1527	pr08	2	834	pr13	4	1394
rc203	2	1640	pr09	2	909	pr14	4	1691
rc204	2	1718	pr10	2	1145	pr15	4	2085
rc205	2	1462	pr13	2	846	pr16	4	2058
rc206	2	1552	pr15	2	1238	pr17	4	935
rc207	2	1608	pr18	2	955	pr18	4	1558
rc208	2	1705	pr19	2	1041	pr19	4	1789
c103	3	990	pr20	2	1251	pr20	4	2121
r104	3	778	pr02	3	945			

- Greedy Randomized Adaptive Search procedure with Evolutionary Local Search (GRASP/ELS), proposed by [85], tested on Intel Pentium 4 processor 3.00 GHz,
- Granular Variant of Variable Neighborhood Search (GVNS), proposed by [86], tested on Pentium(R) 4 with 3 GHz processor.

The results of GVNS, GRASP-ELS and ACS were obtained with 5 runs of the algorithm on each instance. VNS was run 10 times per instance while ILS and SA were executed only once. We noted that results of VNS were taken from the website of the authors $http://prolog.univie.ac.at/research/op/$. It also appears that these results are better than the ones published in the journal article [127]. The quality of the produced solutions is given in terms of the relative percentage error (RPE), that is $RPE = \frac{BKS - Z_{max}}{BKS} \cdot 100$ where Z_{max} denotes the maximal score obtained over different runs and BKS the best known solution value in the literature. The Table 4.3 summarizes the comparison and reports the relative percentage error (RPE) and the average computational time in seconds CPU_{avg} for each instance set. Table 4.4 details the new best-known solutions found by MA. For each instance we give the number of vehicles and the new best score obtained. Finally, detailed results are reported in Tables 4.54.14 and compared with results attained by the state-of-the-art algorithms in the literature. For each instance, we give the best solution (BKS) obtained by all methods including the ones found in this work, the max score and the average computational time in seconds.

As presented in Table 4.3. MA produces the best relative gap which is equal to $-0,23\%$ and generates 33% of best found solutions The first conclusion that can be drawn from these tables is that MA is very competitive compared to the others methods. It outperforms the other methods and improves 101 instances for which the optimal solution remains unknown. However, one should note that MA is far more time consuming. Actually, on the largest instances. MA needs more time to get good quality solutions. The reason appears to be that a lot of time is consumed by local-search operators. This is necessary to take entirely advantage of the MA component.

4.6 Conclusion

In this chapter, a memetic algorithm was proposed for the team orienteering problem with time windows. The main contribution of this work is to show that the use of an indirect representation of solution with an adapted splitting procedure improve the effectiveness of the algorithm. The proposed algorithm integrates several optimization methods, including heuristic approaches, a crossover operator, a local search optimization procedure and a quality-and-diversity based population updating strategy. The computational results obtained prove the efficiency of our memetic algorithm for TOPTW in comparison with the existing ones. The algorithm brings further improvements and has allowed the identification of new best known solutions. The method is also very flexible in the sense that it can address many problem variants with a unified methodology and common parameter

settings. Future work will focus on extending the methodology to a wider array of vehicule routing problems with time windows.

Table 4.5: Results for Solomon's instances with m = 1.

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
c101	320	320	320	0.11	c201	870	870	870	51.78
c102	360	360	360	0.72	c202	930	930	930	181.43
c103	400	400	400	3.23	c203	960	960	960	175.22
c104	420	420	420	3.13	c204	980	980	972	116.93
c105	340	340	340	0.01	c205	910	910	910	84.47
c106	340	340	340	0.01	c206	930	930	930	167.48
c107	370	370	370	0.95	c207	930	930	930	120.39
c108	370	370	370	0.59	c208	950	950	950	81.53
c109	380	380	380	0.07					
r101	198	198	198	1.72	r201	797	797	797	93.63
r102	286	286	286	0.72	r202	930	930	925	218.72
r103	293	293	293	5.43	r203	1028	1026	1025.2	307.58
r104	303	303	303	3.10	r204	1093	1093	1093	265.24
r105	247	247	247	3.24	r205	953	953	953	128.66
r106	293	293	293	0.57	r206	1032	1032	1032	280.29
r107	299	299	299	12.66	r207	1078	1078	1077.6	273.05
r108	308	308	308	3.90	r208	1118	1118	1118	287.77
r109	277	277	277	3.35	r209	962	961	958.6	283.15
r110	284	284	284	16.01	r210	1002	1001	1000.6	257.27
r111	297	297	297	4.15	r211	1051	1051	1050.8	201.77
r112	298	298	298	9.65					
rc101	219	219	219	0.06	rc201	795	795	795	168.82
rc102	266	266	266	0.08	rc202	938	938	938	180.08
rc103	266	266	266	0.74	rc203	1003	1000	998.4	192.17
rc104	301	301	301	1.91	rc204	1143	1143	1139.8	182.18
rc105	244	244	244	1.68	rc205	859	859	859	179.29
rc106	252	252	252	2.84	rc206	899	895	895	187.04
rc107	277	277	277	1.04	rc207	983	983	983	298.39
rc108	298	298	298	4.32	rc208	1054	1053	1049.6	224.20

Table 4.6: Results for Solomon's instances with m = 2.

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
c101	590	590	590	47.76	c201	1460	1450	1450	94.16
c102	660	660	656	90.89	c202	1470	1470	1470	168.87

continued on next page

Table 4.6 – continued

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
c103	720	720	720	65.20	c203	1480	1480	1480	250.24
c104	760	760	760	74.76	c204	1490	1490	1490	285.21
c105	640	640	640	85.90	c205	1470	1470	1470	91.48
c106	620	620	620	72.12	c206	1480	1480	1480	101.14
c107	670	670	670	72.15	c207	1490	1490	1486	175.98
c108	680	680	680	42.40	c208	1490	1490	1490	152.33
c109	720	720	720	79.64					
r101	349	349	349	34.57	r201	1260	1254	1251	215.14
r102	508	508	508	32.28	r202	1353	1352	1348.8	505.76
r103	522	522	521.8	62.17	r203	1430	1428	1422.6	744.78
r104	552	552	552	40.25	r204	1458	1458	1458	833.92
r105	453	453	453	48.06	r205	1402	1402	1398	512.66
r106	529	529	529	39.02	r206	1451	1451	1451	750.12
r107	538	538	538	51.59	r207	1458	1458	1458	547.01
r108	560	560	560	52.21	r208	1458	1458	1458	484.26
r109	506	506	505.8	48.18	r209	1423	1420	1416	661.56
r110	525	525	525	47.29	r210	1440	1438	1432.6	965.19
r111	544	544	544	44.89	r211	1458	1458	1458	760.97
r112	544	544	544	51.23					
rc101	427	427	427	45.65	rc201	1386	1386	1384	160.25
rc102	505	505	505	54.80	rc202	1527	1523	1517.4	277.43
rc103	524	524	524	35.39	rc203	1640	1640	1637.2	419.30
rc104	575	575	575	57.11	rc204	1718	1718	1716	645.00
rc105	480	480	480	43.55	rc205	1462	1462	1458.8	258.05
rc106	483	483	483	38.23	rc206	1552	1550	1539.2	192.07
rc107	534	534	534	51.83	rc207	1608	1607	1603.2	380.99
rc108	556	556	556	44.11	rc208	1705	1705	1694.8	514.69

Table 4.7: Results for Solomon's instances with m = 3.

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
c101	810	810	810	56.56	c201	1810	1810	1810	1.85
c102	920	920	920	65.87	c202	1810	1810	1810	457.64
c103	990	980	980	88.10	c203	1810	1810	1810	601.62
c104	1030	1030	1030	88.00	c204	1810	1810	1810	531.17
c105	870	870	870	45.69	c205	1810	1810	1810	368.33
c106	870	870	870	76.11	c206	1810	1810	1810	456.40
c107	910	910	910	57.17	c207	1810	1810	1810	10.44
c108	920	920	920	74.08	c208	1810	1810	1810	12.20

continued on next page

Table 4.7 – continued

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
c109	970	970	970	85.39					
r101	484	484	484	36.50	r201	1450	1441	1440.6	307.26
r102	694	694	694	52.03	r202	1458	1458	1458	507.45
r103	747	747	746.2	76.37	r203	1458	1458	1458	0.53
r104	778	778	778	68.67	r204	1458	1458	1458	0.06
r105	620	619	619	51.06	r205	1458	1458	1458	0.31
r106	729	729	728.4	69.24	r206	1458	1458	1458	0.13
r107	760	760	759.8	64.40	r207	1458	1458	1458	0.07
r108	797	797	797	65.12	r208	1458	1458	1458	0.08
r109	710	710	710	45.46	r209	1458	1458	1458	0.19
r110	737	737	736.8	61.78	r210	1458	1458	1458	0.30
r111	774	774	773.4	46.89	r211	1458	1458	1458	0.06
r112	776	776	776	65.21					
rc101	621	621	621	42.64	rc201	1699	1698	1698	300.21
rc102	714	714	711.8	41.00	rc202	1724	1724	1724	357.77
rc103	764	764	760.6	62.32	rc203	1724	1724	1724	0.57
rc104	835	835	835	70.56	rc204	1724	1724	1724	0.18
rc105	682	682	682	53.12	rc205	1719	1719	1719	413.72
rc106	706	706	706	50.03	rc206	1724	1724	1724	325.00
rc107	773	773	773	47.58	rc207	1724	1724	1724	301.87
rc108	795	795	795	70.48	rc208	1724	1724	1724	0.13

Table 4.8: Results for Solomon's instances with m = 4.

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
c101	1020	1020	1020	65.73	c201	1810	1810	1810	0.04
c102	1150	1150	1150	86.73	c202	1810	1810	1810	0.04
c103	1210	1210	1210	155.53	c203	1810	1810	1810	0.06
c104	1260	1260	1258	229.04	c204	1810	1810	1810	0.05
c105	1060	1060	1060	56.53	c205	1810	1810	1810	0.03
c106	1080	1080	1078	78.42	c206	1810	1810	1810	0.03
c107	1120	1120	1120	78.11	c207	1810	1810	1810	0.04
c108	1140	1140	1136	98.76	c208	1810	1810	1810	0.04
c109	1190	1190	1190	106.46					
r101	611	611	611	39.64	r201	1458	1458	1458	0.32
r102	843	843	841	71.49	r202	1458	1458	1458	0.24
r103	928	928	928	92.51	r203	1458	1458	1458	0.05
r104	975	975	974.2	107.51	r204	1458	1458	1458	0.07
r105	778	778	776.8	67.09	r205	1458	1458	1458	0.04

continued on next page

Table 4.8 – continued

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
r106	906	906	906	80.09	r206	1458	1458	1458	0.05
r107	950	950	950	89.77	r207	1458	1458	1458	0.06
r108	995	994	992.8	89.17	r208	1458	1458	1458	0.08
r109	885	885	883.8	54.19	r209	1458	1458	1458	0.05
r110	915	915	907.2	97.16	r210	1458	1458	1458	0.04
r111	953	952	949.2	89.19	r211	1458	1458	1458	0.07
r112	974	974	971.4	75.69					
rc101	811	811	811	53.18	rc201	1724	1724	1724	0.56
rc102	909	909	905.8	64.24	rc202	1724	1724	1724	0.11
rc103	975	975	974.8	53.69	rc203	1724	1724	1724	0.06
rc104	1065	1065	1065	66.28	rc204	1724	1724	1724	0.05
rc105	875	875	875	47.14	rc205	1724	1724	1724	0.29
rc106	909	909	908.2	54.40	rc206	1724	1724	1724	0.05
rc107	987	987	986	63.15	rc207	1724	1724	1724	0.05
rc108	1025	1025	1025	59.19	rc208	1724	1724	1724	0.06

Table 4.9: Results for Cordeau's instances with m = 1.

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
pr01	308	308	308	1.25	pr11	353	353	353	25.68
pr02	404	404	404	16.79	pr12	442	441	441	113.55
pr03	394	394	394	17.59	pr13	467	467	467	171.25
pr04	489	489	489	41.11	pr14	567	555	554.4	762.21
pr05	595	595	595	255.98	pr15	708	708	708	1298.59
pr06	591	591	590.8	2485.64	pr16	674	650	645.2	1396.42
pr07	298	298	298	3.20	pr17	362	362	362	34.96
pr08	463	463	463	12.80	pr18	539	539	539	249.72
pr09	493	493	493	247.51	pr19	562	560	553	1305.86
pr10	594	594	591.8	1777.97	pr20	667	648	648	3672.54

Table 4.10: Results for Cordeau's instances with m = 2.

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
pr01	502	502	502	9.28	pr11	566	566	565.2	21.19
pr02	715	713	712.2	132.33	pr12	774	768	763.8	267.39
pr03	742	742	738	213.80	pr13	846	846	841	395.61

continued on next page

Table 4.10 – continued

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
pr04	928	928	927.4	752.19	pr14	1017	994	993	1566.21
pr05	1103	1101	1096.2	1225.57	pr15	1238	1233	1227.8	2459.94
pr06	1076	1076	1073.2	4658.92	pr16	1224	1198	1187.4	8314.33
pr07	566	566	566	23.97	pr17	652	646	646	32.00
pr08	834	834	833	317.37	pr18	955	955	953.6	425.84
pr09	909	909	908.6	1585.38	pr19	1041	1040	1036.2	1536.54
pr10	1145	1145	1142.2	3996.52	pr20	1251	1251	1249.4	6423.65

Table 4.11: Results for Cordeau's instances with m = 3.

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
pr01	622	622	620.2	10.51	pr11	654	654	654	17.52
pr02	945	944	943.2	228.38	pr12	1004	999	996.8	236.82
pr03	1014	1014	1010	337.76	pr13	1159	1159	1158	684.75
pr04	1298	1298	1287.8	1219.08	pr14	1376	1367	1362.2	1794.54
pr05	1500	1499	1496.2	2263.41	pr15	1694	1688	1679.6	3399.83
pr06	1516	1515	1513.4	5241.36	pr16	1668	1651	1644	7909.67
pr07	744	744	744	34.06	pr17	842	839	838.6	62.72
pr08	1141	1140	1140	410.98	pr18	1289	1289	1280.6	471.29
pr09	1282	1279	1274.2	1327.29	pr19	1433	1421	1418	2579.06
pr10	1586	1583	1576.2	3089.22	pr20	1722	1713	1695	6725.71

Table 4.12: Results for Cordeau's instances with m = 4.

Instance	BKS	MA			Instance	BKS	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
pr01	657	657	657	9.81	pr11	657	657	657	20.33
pr02	1083	1080	1078	138.85	pr12	1134	1134	1131.4	191.78
pr03	1247	1247	1243.6	346.83	pr13	1394	1395	1385.8	712.72
pr04	1596	1595	1584.4	1466.62	pr14	1691	1685	1674.4	1598.24
pr05	1859	1859	1844.4	3157.15	pr15	2085	2080	2072.2	3781.57
pr06	1895	1898	1884.4	6419.55	pr16	2058	2053	2039.2	8876.34
pr07	876	876	876	37.64	pr17	935	935	931	62.74
pr08	1392	1383	1375.4	452.98	pr18	1558	1557	1543.4	688.26
pr09	1623	1622	1617.4	2114.14	pr19	1789	1784	1773	3409.73
pr10	1967	1961	1950.6	3930.48	pr20	2121	2112	2099.4	8505.26

Table 4.13: Results for new Solomon's instances.

Instance	*Opt*	MA			Instance	*Opt*	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
c101	1810	1810	1810	17.64	c201	1810	1810	1810	0.04
c102	1810	1810	1810	28.95	c202	1810	1810	1810	0.04
c103	1810	1810	1810	3.61	c203	1810	1810	1810	0.06
c104	1810	1810	1810	0.24	c204	1810	1810	1810	0.05
c105	1810	1810	1810	14.74	c205	1810	1810	1810	0.03
c106	1810	1810	1810	44.59	c206	1810	1810	1810	0.03
c107	1810	1810	1810	3.64	c207	1810	1810	1810	0.04
c108	1810	1810	1810	3.53	c208	1810	1810	1810	0.04
c109	1810	1810	1810	1.12					
r101	1458	1458	1458	6.31	r201	1458	1458	1458	0.31
r102	1458	1458	1458	4.98	r202	1458	1458	1458	8.89
r103	1458	1458	1456.2	108.60	r203	1458	1458	1458	0.52
r104	1458	1455	1449.8	246.14	r204	1458	1458	1458	14.19
r105	1458	1458	1458	5.22	r205	1458	1458	1458	0.30
r106	1458	1458	1458	12.26	r206	1458	1458	1458	0.18
r107	1458	1458	1457.8	153.76	r207	1458	1458	1458	48.73
r108	1458	1458	1457.6	129.25	r208	1458	1458	1458	0.85
r109	1458	1458	1456.2	146.04	r209	1458	1458	1458	0.21
r110	1458	1457	1455.2	201.73	r210	1458	1458	1458	0.30
r111	1458	1458	1457.2	169.23	r211	1458	1458	1458	220.38
r112	1458	1458	1456.4	259.11					
rc101	1724	1724	1721.8	110.61	rc201	1724	1724	1724	0.60
rc102	1724	1721	1718.6	143.84	rc202	1724	1724	1724	19.92
rc103	1724	1724	1724	11.40	rc203	1724	1724	1724	0.72
rc104	1724	1724	1724	1.96	rc204	1724	1724	1724	0.21
rc105	1724	1721	1719.8	97.34	rc205	1724	1724	1724	0.36
rc106	1724	1718	1715.4	138.96	rc206	1724	1724	1724	0.87
rc107	1724	1724	1724	8.10	rc207	1724	1724	1724	1.27
rc108	1724	1724	1724	16.67	rc208	1724	1724	1724	0.11

Table 4.14: Results for new Cordeau's instances.

Instance	*Opt*	MA			Instance	*Opt*	MA		
		Z_{max}	Z_{avg}	CPU_{avg}			Z_{max}	Z_{avg}	CPU_{avg}
pro01	657	622	619.6	10.18	pro06	3671	3671	3671	56.73
pro02	1220	1213	1210.2	167.49	pro07	948	940	940	50.28
pro03	1788	1785	1784.2	784.00	pro08	2006	2006	2006	2.73
pro04	2477	2477	2477	39.14	pro09	2736	2736	2736	3.19
pro05	3351	3351	3351	9.48	pro10	3850	3850	3850	3.05

Nouvelles bornes inférieures pour le VRPTW

Abstract

The Vehicle Routing Problem with Time Windows (VRPTW) consists in determining the routing plan of vehicles with identical capacity in order to supply the demands of a set of customers within predefined time windows. This complex multi-constrained problem has been widely studied due to its industrial, economic, and environmental implications. In this work, we are interested in reducing the number of vehicles instead of the total distance traveled. We first analyze several lower bounds based on incompatibilities between customers and capacity constraints of the vehicles. Then, we introduce an adaptation of Energetic Reasoning algorithm for VRPTW with a limited fleet. The proposed approach focuses on some time-intervals and exploits time constraints, incompatibility graph and the "bin packing" aspect of the problem in order to obtain a new valid lower bounds on the fleet size needed. It can also be used to get time windows adjustments for m-VRPTW. Finally, we provide the results of computational experiments on the standard benchmark of Solomon. Our algorithms improved the results of classical lower bound techniques and allows us to prove the optimality of 23 instances.

5.1 Introduction

The vehicle routing problem (VRP) is a classical combinatorial optimization problem that has become a key component of distribution and logistics management. The objective of the VRP consists in the determination of a vehicle scheduling strategy to minimize the number of routes and the corresponding total travel distance or cost. The problem was first introduced by Dantzig and Ramser [45]. Additional constraints which make the resulting problems align better with real-life applications were associated with the classical VRP and hundreds of models and algorithms have been proposed for different versions of this problem.

In this chapter, we deal with the case where vehicle capacity is limited and customers impose time windows constraints. This extended version is called vehicle routing problem with time windows (VRPTW) [55]. In VRPTW, a set of customers must be served by a fleet of vehicles located in a single depot. A quantity of goods

should be delivered to each customer whose service takes an amount of time. Every customer imposes a time window during which they can be served. The earliest possible time when the service of customer i can begin (i.e. the start of the time window) is denoted e_i while the latest possible time is denoted l_i. A vehicle is allowed to arrive at a customer i before e_i, but it must wait until that time to deliver the goods. Since deliveries cannot be split, a customer is always served by a single vehicle. All vehicles are identical and have a maximum capacity C. The aim is to plan routes with the minimum cost. These routes should start from and finish in a unique depot such that all the time windows and capacity constraints are respected.

VRPTW has a wide range of applications in distribution management. Common examples are newspaper delivery, beverage and food delivery, commercial and industrial waste collection [65]. It has been a subject of an intensive research focused mainly on heuristic and metaheuristic approaches. In the next section, we present a brief literature review of vehicle routing problem with time windows.

5.2 Literature review

VRPTW was first introduced by Solomon [118]. Both exact and heuristic algorithms have been used to solve VRPTW. The main part of the literature on exact methods is concerned with the variant of the problem where the number of available vehicles is not fixed. A review of the exact methods up to 2002 is reported in [39]. Kallehauge [75] gave a detailed analysis of existing formulations. More recently, Baldacci et al. [7] reviewed mathematical formulations, relaxations and recent exact methods. They reported the computational comparison of the methods of Jepsen et al. [73], Desaulniers et al. [47] and Baldacci et al. [6] that are considered the most effective exact methods in the literature. These approaches have significantly improved the quality of the lower bounds of instances with up to 100 customers. The key factor of their success is the effective combination between the set partitioning formulation and column generation based algorithms.

Since, VRPTW is an NP-Hard problem [93], the computational times for exact methods can be very high, even for instances with a moderate size. This has been the motivation for researches aimed at developing approximation techniques. These techniques are classified into three main categories: heuristics, metaheuristics and hybrid methods. Heuristics are generally used to provide upper bounds for exact algorithms and to generate initial solutions for metaheuristic approaches. They are based on routes construction and routes improvement using different criteria for customer selection and insertion [100, 104]. Unlike classical improvement methods, metaheuristics usually incorporate mechanisms to continue the exploration of the search space after a local minimum is encountered. The first metaheuristics used to solve VRPTW are Tabu search [115, 121].

There has been a significant increase in interest in solving VRPTW using genetic algorithms (GA). Most of these works present a hybridization of a GA with different

construction heuristics [20, 18], local searches [125, 109, 74] and other metaheuristics such as tabu search [58, 59, 68, 71] and ant colony algorithm [17]. Methods proposed in Homberger and Gehring [70, 71] and Gehring and Homberger [58, 59] can be considered to be Pareto optimal in terms of solution quality and time consumption. These approaches seem to produce the best results among genetic algorithms. We refer to Bräysy et al. [29] for detailed descriptions of evolutionary methods applied on VRPTW. Two other surveys given by Bräysy and Gendreau [27, 28] describe basic features of heuristics and metaheuristics researches and discuss experimental results on Solomon's benchmark test problems.

It is worth pointing out that the literature concerning VRPTW is split according to the objective considered. While exact methods usually minimize distance, most heuristics consider a hierarchical objective which first minimizes the number of vehicles used and then the total distance. Thus, a solution that employs fewer vehicles is always better than another using more, even if the total time of the first solution is worse. Only if two solutions have the same number of routes, their total times are compared. A few of approximate algorithms consider the total distance as the main objective. The best performing recent heuristics are the hybrid genetic algorithm of Jung and Moon [74], the column generation heuristic of Alvarenga et al. [1] and the memetic algorithm of Labadie et al. [84]. A new optimization framework was later developed by Ursani et al. [129] for the distance minimization objective only. This framework is an iterative procedure between optimization and de-optimization phases and uses a genetic algorithm as an optimization methodology.

A third stream of research focuses on solving VRPTW as a multi-objective problem in which both vehicles and cost are considered depending on the needs of the user [122] [102].

5.3 Problem formulation

In the following, we present a mixed integer formulation for VRPTW. The problem is modeled by an oriented graph $G = (V^+, E)$, where $V^+ = \{0, 1, 2, ..., n\}$ is the vertex set representing the set of customers $V = \{1, 2, ..., n\}$ and the depot 0. $E = \{(i, j) : i \neq j, i, j \in V^+\}$ is the edge set. The capacities of all vehicles are equal and are denoted by Q. A demand q_i, a service time s_i and a time window $[e_i, l_i]$ are associated to each vertex $i \in V$. If the vehicle v arrives earlier than e_i, it must wait before the service can start. Each edge $(i, j) \in E$ is associated with a travel cost $\delta_{i,j}$ which satisfies the triangle inequality. The vehicle must start and finish its tour at the depot. Each customer must be served within a predefined time window and assigned to exactly one vehicle. The total size of deliveries for customers assigned to the same vehicle must not exceed the vehicle capacity Q and the travel cost/time $C(R)$ of each tour R must not exceed l_0 which is the latest possible arrival time to the depot.

The model involves three types of variables: the binary routing variables $x_{ij} \in \{0, 1\}\,(i, j \in V^+)$, the scheduling variables $w_i \geq 0\,(i \in V)$ and the vehicle load

variables $y_i (i \in V)$. The routing variables x_{ij} is one if a vehicle traverses the arc $(i, j) \in E$. The scheduling variable w_i denotes the time the vehicle arrives at customer $i \in V$. y_i denotes the vehicle load at departure from customer i. The formulation is as follows:

$$\min \sum_{i \in V} x_{0i} \tag{5.1}$$

subject to:

$$\sum_{j \in V^+} x_{ij} = 1 \qquad \forall i \in V \tag{5.2}$$

$$\sum_{j \in V^+} x_{ij} - \sum_{j \in V^+} x_{ji} = 0 \qquad \forall i \in V^+ \tag{5.3}$$

$$w_j \geq w_i + x_{ij}(\max(\delta_{i,j} + s_i, e_j - l_i)) - (1 - x_{ij})(l_i - e_j) \qquad \forall i, j \in V \tag{5.4}$$

$$e_i \leq w_i \leq l_i \qquad \forall i \in V \tag{5.5}$$

$$y_j \geq y_i + q_j - (1 - x_{ij})(Q - q_j) \qquad \forall i, j \in V \tag{5.6}$$

$$q_i \leq y_i \leq Q \qquad \forall i \in V \tag{5.7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V^+ \tag{5.8}$$

The objective (5.1) is to minimize the total number of vehicles used to serve the customers. Constraints (5.2) and (5.3) define the routing network and the constraints (5.4) and (5.5) guarantee the connectivity of each tour and ensure that the time windows are respected. We assume that the time windows are ajusted such that $e_i = \max(e_i, \delta_{0,i})$ and $l_i = \min(l_i, l_0 - (\delta_{i,0} + s_i))\ \forall i \in V$. Constraints (5.6) and (5.7) ensure that the vehicle's capacity is not exceeded. Also constraints (5.6) eliminate subtours in a manner similar to (5.4). Finally, (5.8) are integral constraints.

5.4 Lower bounding techniques

To get lower bounds on the vehicle number of VRPTW, we used the approaches described below.

5.4.1 Incompatibilities between customers

The first lower bound is deduced from the compatibility constraints: let i and j be two customers. If there is no feasible route with i and j then they define an incompatible pair of nodes denoted by $i||j$. To identify such a situation, we used the following conditions:

1. $(e_i + s_i + \delta_{i,j} > l_j) \wedge (e_j + s_j + \delta_{j,i} > l_i) \Rightarrow i||j$: the customers i and j cannot be in the same route due to their time windows constraints.

2. $(C(R_1) > l_0) \wedge (C(R_2) > l_0)$ where $R_1 = (0, i, j, 0) \wedge R_2 = (0, j, i, 0) \Rightarrow i||j$: the travel cost of the tour with i and j exceeds the cost limit l_0.
3. $q_i + q_j > Q \Rightarrow i||j$: the sum of the demands is greater than the vehicle capacity.

Using these conditions, we construct the graph of incompatibilities between customers defined as: $G^V_{inc} = (V, N_V)$ with $V = \{1, ..., n\}$ and $N_V = \{(i, j) : i||j\}$. The minimum number of routes to be used LB_{Clique} is the size of the maximum clique in the graph G^V_{inc}. Finding a clique with the greatest cardinality involves the use of an exact method with exponential worst case performance. Nevertheless, our experiments showed that the maximum clique can be identified in a fraction of a second using the heuristic algorithm developed by Dang and Moukrim [42]. The details of implementation are given in Section 5.5.

5.4.2 Vehicle capacity constraints

The second bound is based on a relaxation of time windows constraints. When we consider only the capacity constraints, VRPTW is reduced to Bin Packing problem (BPP). Each vehicle is a bin with a fixed size Q. Each demand q_i is an item that should be put in one bin. We are interested in a lower bound of the required number of bins for a given set of items. Since BPP is NP-hard, we use instead an obvious lower bound given by $LB_{Capacity} = \sum_{i \in V} q_i/Q$. In order to improve $LB_{Capacity}$, we add incompatibility constraints to BPP. A Heuristic algorithm can be applied to get a lower bound on the resulting bin packing problem with conflicts noted LB_{BPPC}. The algorithm starts by computing a maximal clique on the extended conflict graph. In this graph, two vertices are connected by an edge if they are incompatible. A bin is initialized for every item in the clique, then the remaining customers are assigned to the bins.

5.4.3 New lower bounds inspired from Energetic Reasoning

In this section, we present a brief overview of Energetic Reasoning, then we discuss its adaptation to VRPTW.

5.4.3.1 Energetic Reasoning

Energetic Reasoning (ER) is one of the most powerful propagation algorithms. It has been originally developed by Erschler et al. [50] for the Cumulative Scheduling Problems (CuSP). The idea is to propose a smart way to consider time and resource constraints simultaneously in a unique reasoning. In that context, the energy is generally defined by multiplying a time duration by the resource quantity of a given time interval. Considering the quantities of energy supplied by the resources and consumed by the tasks within given intervals, the energetic approach aims to develop satisfiability tests and time-bound adjustments to ensure that either a given schedule is not feasible or some necessary conditions, satisfying any feasible schedule, must be derived. Adjustment techniques aim to tighten these time windows by considering

the resource constraints and by removing the parts in the time windows that cannot lead to a feasible solution. Since its inception, Energetic Reasoning has gained popularity and has been used for solving more complex scheduling problems [12, 11, 10, 101].

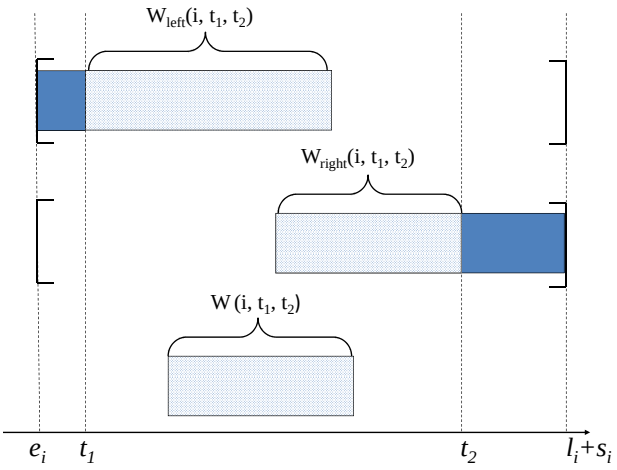


Figure 5.1: The work of an activity.

In order to keep the same notation used for vehicle routing problem, we describe the CuSP as follows. We consider a set V of n activities to be scheduled on a resource of quantity m. Each activity i has a release time e_i, a latest start time l_i and a processing time s_i. Moreover, the activity i requires a constant amount b_i of resource throughout its processing. We will deal here only with the case where $b_i = 1, \forall i \in V$. This is equivalent to the problem of scheduling n activities on m identical parallel machines. For ease of presentation, we denote this problems as PMSP.

Given a time interval $[t_1, t_2]$, with $t_1 < t_2$, the part of an activity i that must be processed between t_1 and t_2 is called *work* of i in the time interval $[t_1, t_2]$. To compute this *work*, the activities are either *left-shifted* or *right-shifted* on their time window, which means that, they can start either at their release date e_i, or at their latest start time l_i. Thus, the work of an activity i over $[t_1, t_2]$ is equal to the minimum between its *left work* and its *right work*. For convenience, the left work, the right work and the work of an activity i over $[t_1, t_2]$ are denoted respectively $W_{left}(i, t_1, t_2)$, $W_{right}(i, t_1, t_2)$ and $W(i, t_1, t_2)$. They are formally defined as follows:

$$W_{left}(i, t_1, t_2) = min(t_2 - t_1, s_i, max(0, e_i + s_i - t_1)) \quad (5.9)$$

$$W_{right}(i, t_1, t_2) = min(t_2 - t_1, s_i, max(0, t_2 - l_i)) \quad (5.10)$$

$$W(i, t_1, t_2) = min(W_{left}(i, t_1, t_2), W_{right}(i, t_1, t_2)) \quad (5.11)$$

Finally, we define the total work $W(t_1, t_2)$ as the sum of the works of all the activities. If this work is greater than the available energy then no feasible solution exists.

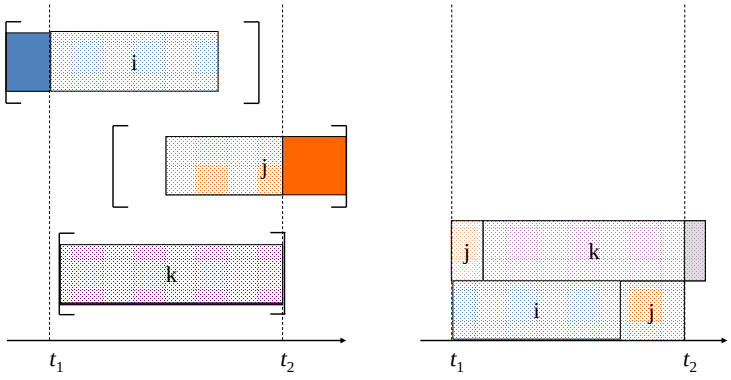


Figure 5.2: An example of an infeasible instance: insufficient energy to serve the three customers.

Proposition 5.4.1. ***Satisfiability test***

if $\exists[t_1, t_2]$, $W(t_1, t_2) > m * (t_2 - t_1)$ *then the instance is infeasible.*

Now we introduce the concept of *slack* of an activity i on a time interval $[t_1, t_2]$. This slack corresponds to the available energy that can be used to process i. It can be defined also as the maximum amount of time that might be allocated to i during the time interval $[t_1, t_2]$. Let $SL(i, t_1, t_2)$ be the slack of $V \setminus \{i\}$, then:

$$SL(i, t_1, t_2) \quad = \quad m \cdot (t_2 - t_1) - W(t_1, t_2) + W(i, t_1, t_2) \tag{5.12}$$

Thus, following the definition previously introduced, the slack allows to adjust the time-bounds of activities. For example, if the right work of the activity $i \in V$ is strictly greater than $SL(i, t_1, t_2)$, the activity i cannot be right-shifted, i.e., it cannot start at its latest start time. Hence, only a part of activity i, smaller than or equal to the slack $SL(i, t_1, t_2)$ can be processed on $[t_1, t_2]$. More precisely, the adjustment scheme can be defined as:

Proposition 5.4.2. ***Release date adjustments***

$\forall i \in V$, *if* $W_{left}(i, t_1, t_2) > SL(i, t_1, t_2)$ *then* $e_i \leftarrow max(e_i, t_2 - SL(i, t_1, t_2))$

Proposition 5.4.3. ***Lastest start time adjustments***

$\forall i \in V$, *if* $W_{right}(i, t_1, t_2) > SL(i, t_1, t_2)$ *then* $l_i \leftarrow min(l_i, t_1 - s_i + SL(i, t_1, t_2))$

Note that one crucial point to apply efficiently Energetic Reasoning is to determine the relevant time-intervals on which it may be useful to both check feasibility conditions and time-bound adjustments. Baptiste et al. [12] have proved that the only relevant intervals $[t_1, t_2]$ that need to be considered are those where $t_1 \in T_1$ and $t_2 \in T_2$ such as $t_1 < t_2$, $T_1 = \{e_i, i \in V\} \cup \{l_i, i \in V\} \cup \{e_i + s_i, i \in V\}$

and $T_2 = \{l_i + s_i, i \in V\} \cup \{e_i + s_i, i \in V\} \cup \{l_i, i \in V\}$. The satisfiability test and time bound adjustment algorithm run in $O(n^3)$. The detailed steps are summarized in Algorithm 5.1.

Algorithm 5.1: Energetic Reasoning: satisfiability test and time bound adjustment

```
Data: I PMSP instance;
Result: I_adj PMSP instance with time bound adjustment;
begin
    initialization;
    T_1 = {e_i, i ∈ V} ∪ {l_i, i ∈ V} ∪ {e_i + s_i, i ∈ V};
    T_2 = {l_i + s_i, i ∈ V} ∪ {e_i + s_i, i ∈ V} ∪ {l_i, i ∈ V};
    foreach t_1 ∈ T_1 do
        foreach t_2 ∈ T_2 such as t_1 < t_2 do
            W ← 0;
            foreach i ∈ V do
                W ← W + W(i, t_1, t_2);
            if W > m * (t_2 − t_1) then
                Infeasible instance;
            else
                foreach i ∈ V do
                    SL ← m * (t_2 − t_1) − W + W(i, t_1, t_2) ;
                    if SL < W_left(i, t_1, t_2) then
                        e_i ← max(e_i, t_2 − SL);
                    if SL < W_right(i, t_1, t_2) then
                        l_i ← min(l_i, t_1 + SL − s_i);
```

5.4.3.2 From m-VRPTW to PMSP

Our approach is to relax a VRPTW instance, where a limited number of vehicles is given, in order to obtain a PMSP instance. A trivial relaxation of an m-VRPTW instance can be done by ignoring travel times, customer demands and vehicle capacities. We obtain a PMSP where the vehicles are considered as m identical parallel machines, the number of activities is equal to the number of customers n and each activity i has to be processed for s_i units of time by only one machine. The processing of activity i cannot be started before its release date e_i, and each activity i has a delivery time $l_i + s_i$.

In vehicle routing problems, travel times are not negligible compared to the service times. Ignoring the travel time would undervalue the energy consumed. Therefore, few adjustments could be performed and Energetic Reasoning becomes inefficient. Better results are obtained by considering the time that a vehicle needs

to travel in order to visit each customer. First, the travel time $\delta_{i,j}$ between the customers i and j is updated to eliminate the waiting time at the customer j.

$$\delta_{i,j} \leftarrow max(\delta_{i,j}, e_j - (l_i + s_i)) \qquad \forall i \in V \;\; j \in V^+ \tag{5.13}$$

Then, the number of potential successors of the customer i is reduced. This is performed by eliminating the transition $\delta_{i,j}$ if j cannot be served after i due to its time windows:

$$if(e_i + s_i + \delta_{i,j} > l_i) \qquad then \;\; \delta_{i,j} \leftarrow \infty \qquad \forall i \in V^+ \;\; j \in V^+ \setminus \{i\} \tag{5.14}$$

Before giving the detail of our travel evaluation procedure, we note by I' the PMSP instance derived from the m-VRPTW instance I. We associate I' with a graph $G' = (V', E')$ which is built by performing the following transformations:

1. The number of vertices is increased by introducing m artificial origin nodes V_o and m artificial destination nodes V_d. Let n' be the size of V', $n' = n + 2 * m$ and $V' = V \cup V_o \cup V_d$ with $V = \{1, ..., n\}$, $V_o = \{n+1, ..., n+m\}$ and $V_d = \{n+m+1, ..., n+2*m\}$. Then, the arc set is defined by $E' = E \cup \{(i,j) : i, j \in V', i \neq j, i \in V' \setminus V \;\; j \in V' \setminus V$.
2. The distance matrix $\Delta = (\delta_{i,j})$ is extended to $\Delta' = (\delta'_{i,j})$ which is associated with E' such as:

$$\delta'_{i,j} = \begin{cases} \delta_{i,j} & (i, j \in V), \\ \delta_{0,j} & (i \in V_o, j \in V), \\ \delta_{i,0} & (i \in V, j \in V_d), \\ \infty \quad (i \in V', j \in V_o) \; or \; (i \in V_d, j \in V') \end{cases}$$

3. The set of vertices V' in G' denotes the $2 * n + m$ activities assigned to I'. The range of the possible start dates of an activity $i \in V'$ is derived from the customer's time windows including the time window associated to the depot $[0, l_0]$.

$$e'_i = \begin{cases} e_i & (i \in V), \\ 0 & (i \in V_o \cup \in V_d) \end{cases} \quad l'_i = \begin{cases} l_i & (i \in V), \\ 0 & (i \in V_o), \\ l_0 & (i \in V_d) \end{cases}$$

4. The processing time s'_i is calculated according to the service time of the customer i and the time that the vehicle has to travel and wait in order to visit the next customer j. To include a travel cost $\delta'_{i,j}$ in the service time of an activity i, the customer j should immediately be served after the customer i by the same vehicle. This requires prior knowledge of the m-VRPTW solution. Since we cannot exactly define the energy allocated to routes, we make a default evaluation of the energy consumed during travel. The time allocated to the activity "serve a customer" consists of the service time s_i and the travel time that the vehicle will necessary perform to reach the next customer. For each customer i, we denote by $\phi^{min}_{toSuccessor}(i)$ the minimal travel time to reach

any other customer. To account for the travel time from the depot to the first customer in each route, we define $\delta^{min}_{0,1}, ..., \delta^{min}_{0,m}$ the values of the m smallest travel time from the depot to customers.

$$\phi^{min}_{toSuccessor}(i) \leftarrow min_{j \in V'}\{\delta'_{i,j}\} \qquad \forall i \in V \tag{5.15}$$
$$\phi^{min}_{toSuccessor}(i) \leftarrow \delta^{min}_{0,i-n} \qquad \forall i \in V_o \tag{5.16}$$
$$\phi^{min}_{toSuccessor}(i) \leftarrow 0 \qquad \forall i \in V_d \tag{5.17}$$

$$s'_i \leftarrow s_i + \phi^{min}_{toSuccessor}(i) \qquad \forall i \in V \tag{5.18}$$
$$s'_i \leftarrow 0 \qquad \forall i \in V_d \tag{5.19}$$
$$s'_i \leftarrow \phi^{min}_{toSuccessor}(i) \qquad \forall i \in V_o \tag{5.20}$$

$$\delta'_{i,j} \leftarrow max(0, \delta'_{i,j} - \phi^{min}_{toSuccessor}(i)) \qquad \forall i \in V', j \in V' \tag{5.21}$$

The minimization in (5.15) determines the time needed to go from i to its closest neighbor or to the depot (m destination nodes). Note that, the activities associated to the artificial destination nodes V_d have no processing time (5.19), while the time associated to the artificial origin nodes V_o are given using the m smallest distances from the depot (5.16)(5.20).

The minimum travel time $\phi^{min}_{toSuccessor}(i)$ is subtracted from each element of the row i of the distance matrix Δ' (5.21); that constant is added to the processing time of the activity i (5.18), and it is therefore subtracted from the total distance of any solution. This is because every solution must include one and only one customer from this row. We call *reducing* the row, the process of subtracting the smallest element from the remaining ones. This process was introduced by Little et al. [96] to solve the well known Traveling Salesman Problem.

Next, we apply the same argument to the resulting matrix, by considering the minimal travel time to arrive from any other customer j noted $\phi^{min}_{FromPredecessor}(i)$ (5.22). This time is added at the beginning of the activity i. For this reason, the bounds of the corresponding time window are shifted (5.23) (5.24). After these reducing operations, the distance matrix Δ' contains at least one zero in each row and each column. The example in Figure 5.4 illustrates the proposed m-VRPTW relaxation.

$$\phi^{min}_{FromPredecessor}(i) \leftarrow min_{j \in V'}\{\delta'_{j,i}\} \qquad \forall i \in V \cup V_d \tag{5.22}$$

$$e'_i \leftarrow max(0, e'_i - \phi^{min}_{FromPredecessor}(i)) \qquad \forall i \in V \cup V_d \tag{5.23}$$
$$l'_i \leftarrow max(0, l'_i - \phi^{min}_{FromPredecessor}(i)) \qquad \forall i \in V \cup V_d \tag{5.24}$$

Once the transformations are performed, we apply the same satisfiability test on the relaxed m-VRPTW instance I', using Algorithm 5.1. A new lower bound

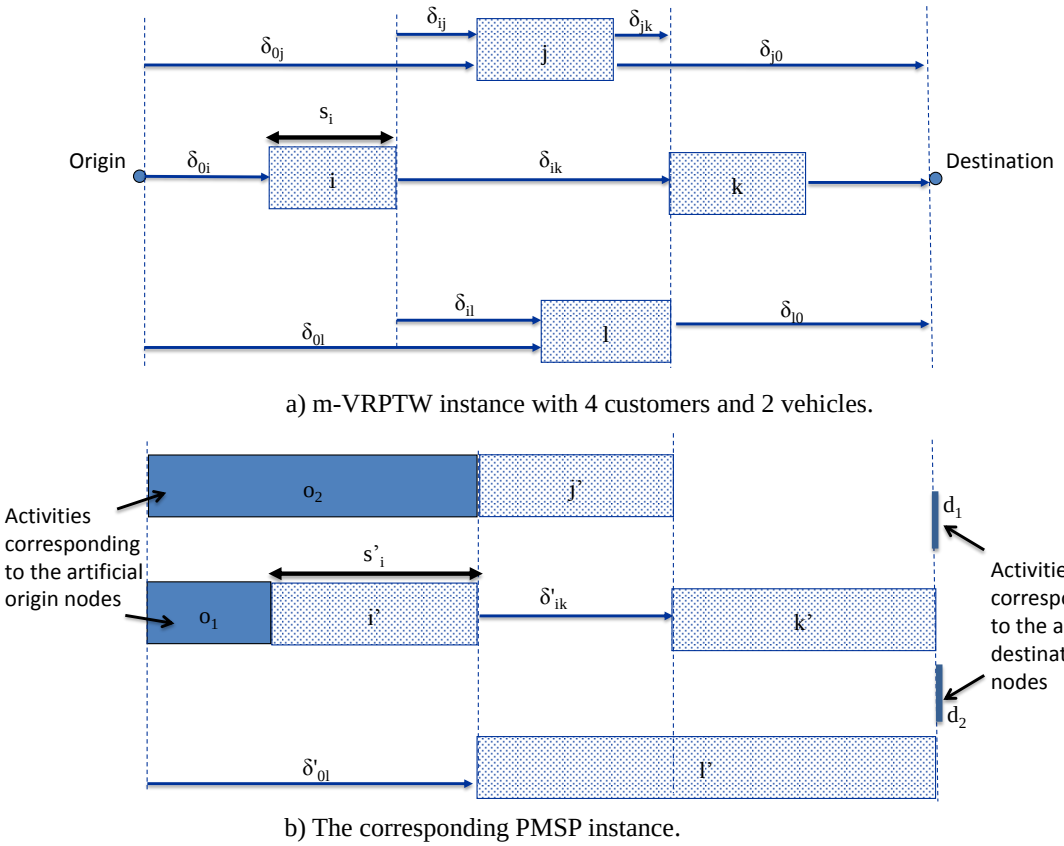


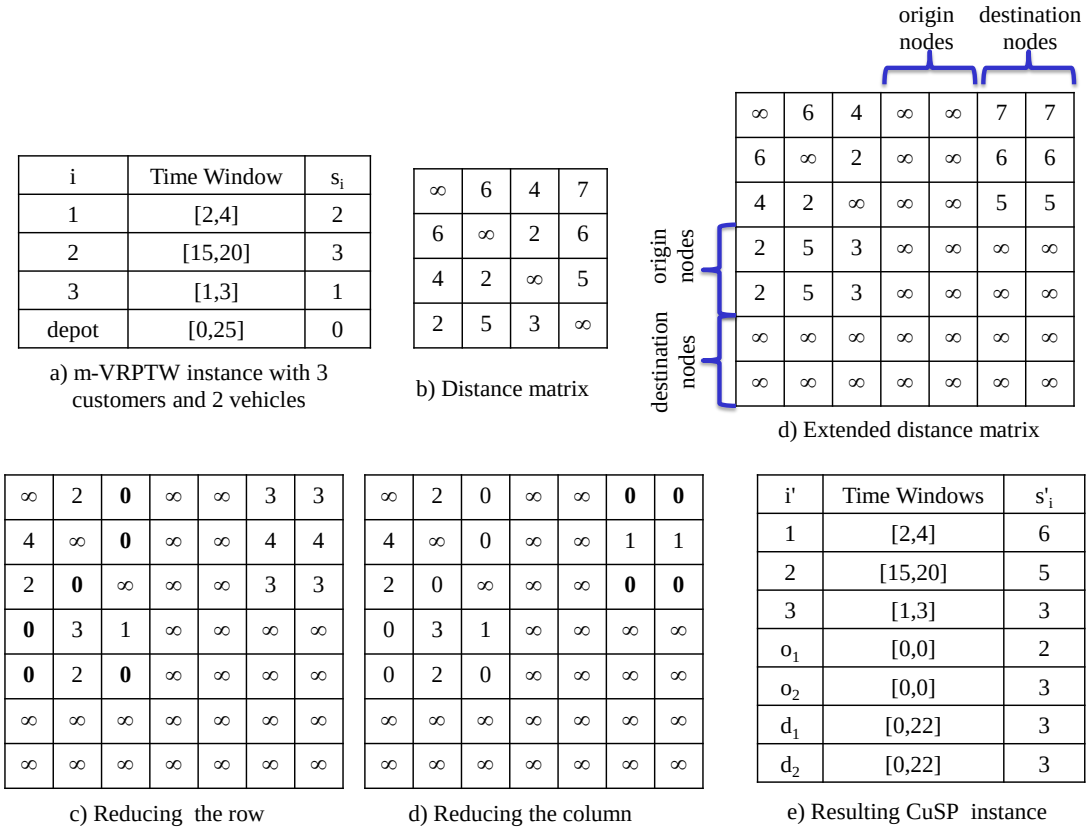
Figure 5.3: An example of m-VRPTW instance relaxed to PMSP instance.

is obtained in the following way: starting from a trial value m, feasibility tests are carried out to detect an infeasibility (that is, the vehicle number cannot be less than m). If an infeasibility is detected, then $m + 1$ is a valid lower bound. The process is reiterated until no infeasibility is detected.

According to the evaluation procedure of travel times, we distinguish two possible lower bounds LB_{eval1} and LB_{eval2}. The former is obtained if the travel times to the successors are considered before the remaining travel times from the predecessors whereas the latter is obtained by reversing the order of considered travels.

5.4.4 Using bin-packing lower bounds into Energetic Reasoning

We extend Energetic Reasoning, using the bin-packing problem with conflicts (BPPC), to get tighter lower bounds for VRPTW. On each time-interval $[t_1, t_2]$, we compute the mandatory parts of activities and then deduce an associated bin-packing instance. The decision version of BPPC that we use can be formulated as follows: given a set of items of different weights and a graph where the nodes represent items and the edges represent conflicts between pairs of items; is there a packing of these items in less than m bins with a capacity Q ?



i	Time Window	s_i
1	[2,4]	2
2	[15,20]	3
3	[1,3]	1
depot	[0,25]	0

a) m-VRPTW instance with 3 customers and 2 vehicles

∞	6	4	7
6	∞	2	6
4	2	∞	5
2	5	3	∞

b) Distance matrix

∞	6	4	∞	∞	7	7
6	∞	2	∞	∞	6	6
4	2	∞	∞	∞	5	5
2	5	3	∞	∞	∞	∞
2	5	3	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞

d) Extended distance matrix

∞	2	**0**	∞	∞	3	3
4	∞	**0**	∞	∞	4	4
2	**0**	∞	∞	∞	3	3
0	3	1	∞	∞	∞	∞
0	2	**0**	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞

c) Reducing the row

∞	2	0	∞	∞	**0**	**0**
4	∞	0	∞	∞	1	1
2	0	∞	∞	∞	**0**	**0**
0	3	1	∞	∞	∞	∞
0	2	0	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞

d) Reducing the column

i'	Time Windows	s'_i
1	[2,4]	6
2	[15,20]	5
3	[1,3]	3
o_1	[0,0]	2
o_2	[0,0]	3
d_1	[0,22]	3
d_2	[0,22]	3

e) Resulting CuSP instance

Figure 5.4: An example of reducing the rows and the columns of the extended distance matrix.

We now state the link between a necessary condition for the existence of a solution and the solution of a BPPC instance: let $I'(V', G_{inc}, m)$ denote a relaxed instance of VRPTW where V' is the set of activities, m the number of available vehicles and G_{inc} the graph of incompatibilities between activities. Let $[t_1, t_2]$ be a time-interval, we assume that the corresponding mandatory parts of activities have been computed: $W(i, t_1, t_2), \forall i \in \{1, ..., n'\}$. Then, $BPPC(I', G_{inc}, t_1, t_2)$ denotes the packing instance which is associated to the scheduling instance I' and the time-interval $[t_1, t_2]$. $BPPC(I', G_{inc}, t_1, t_2)$ is made of n' items with the size $W_i = W(i, t_1, t_2), \forall i \in \{1, ..., n\}$, and m bins. The size of the bin is equal to the length of the time-interval $Q = t_2 - t_1$. Then, deciding whether all mandatory parts of the activities can be scheduled within $[t_1, t_2]$ in I' is equivalent to determine for $BPPC(I', G_{inc}, t_1, t_2)$ if all items can be packed into the available bins.

Property 5.4.1. *If there exists a time-interval $[t_1, t_2]$, such that $BPPC(I', G_{inc}, t_1, t_2)$ has no solution, then there is no solution to the initial problem $I'(V, G_{inc}, m)$.*

Notice that methods based on exact solving of BPPC can only be used either on trivial instances or on small-size instances. Therefore, it is more interesting to use tests that ensure that the minimum number of bins needed to pack all items

is strictly bigger than m. Such tests are deduced from calculating lower bounds of BPPC. As stated in Section 5.4.3.2, Energetic Reasoning uses two procedures to determine the processing time of activities. This leads to define two new lower bounds LB_{REBPPC_eval1} and LB_{REBPPC_eval2}.

5.5 Numerical results

We tested our algorithms on the well known instances introduced by Solomon [118]. The benchmark comprises 6 sets ($R1, C1, RC1, R2, C2, RC2$). Each data set contains 100 customers who have specific euclidean coordinates. Customer's locations are determined using a random uniform distribution for the problem sets $R1$ and $R2$, but are restricted to be within clusters for the sets $C1$ and $C2$. Sets $RC1$ and $RC2$ have a combination of clustered and randomly placed customers. Sets $R1$, $C1$ and $RC1$ have a short scheduling horizon with tight time windows, while $R2$, $C2$ and $RC2$ are based on wide time windows.

Results comparison can be a hard task because two types of methods in the literature deal with VRPTW: heuristic methods usually calculate distances up to two digits, while exact methods calculate up to a single digit only. Exact methods typically use a monolithic total distance objective and only few of Solomon's instances were solved to optimality when the number of vehicles is considered as the primary objective [13]. Therefore, our results are not directly comparable to those methods. We adopt the same double digit precision used in heuristics methods for results comparison. Our algorithm is coded in C++ and all experiments were conducted on an Intel Core i7-2620M 2.70GHz. CPLEX 12.4 was used as MIP solver to provide solutions (or lower bounds) for BPPC sub problems identified in Energetic Reasoning (see Section 5.4.4). We limit the solving time of CPLEX to 1 mn for each BPPC instance.

Computational results are presented in Table 5.1. The performance of lower bounds described in this chapter is presented in columns: LB_{Clique}, LB_{RE_eval1}, LB_{RE_eval2}, LB_{REBPPC_eval1}, LB_{REBPPC_eval2}, $LB_{Capacity}$, LB_{BPPC}. The column UB represents the overall best-published heuristics upper bounds. These results are reported in Solomon's web page: http://web.cba.neu.edu/ msolomon/problems.htm. They are derived from 23 different algorithms that include exact, heuristic, and metaheuristic algorithms. The CPU time (sec.) needed to get a lower bound of the minimum number of vehicles is given in columns Cpu for each technique. The maximum of the lower bounds is reported in column LB. A Lower bound marked with (*) indicates that, for the instance, the best-published solution is proved to be optimal.

In general, the proposed techniques proved the optimality of 23 instances among the 56 instances tested and gave near optimal solution for the rest. The average performance of $LB_{Capacity}$ is consistently better (except for $R1$) than LB_{Clique}, but LB_{Clique} gives superior results for the time-constrained problem in each data set. LB_{Clique} outperforms $LB_{Capcaity}$ in three problems in $R1$ and one problem in $RC1$

by a margin of 43%. The poor quality of LB_{Clique} is attributed to the structure of the data sets. This bound addresses capacity and time incompatible pairs of customers but Solomon's data sets do not favor such pairs. First, there are no capacity incompatible pairs because all customer demands are far less than the vehicle capacity. Second, there is a small number of time incompatible pairs. This is because the data sets were created by placing the time windows of customer i symmetrically around the time to travel from the depot to i ($\delta_{0,i}$). For clustered and semi-clustered problems, customers in the group have approximately the same distance from the depot. Moreover, they are usually within a small distance from each other and have overlapping time windows. These factors increase the likelihood of incompatible pairs.

On the other hand, the four new lower bounds: LB_{RE_eval1}, LB_{RE_eval2}, LB_{REBPPC_eval1} and LB_{REBPPC_eval2} produced consistent results across all data sets. Compared to LB_{Clique} they gave better bounds in 47 instances, were dominated in 4 instances and gave the same lower bound in the remaining 5. As expected, when Energetic Reasoning is combined to BPPC in LB_{REBPPC_eval1} and LB_{REBPPC_eval2}, the results outperform the bounds produced by LB_{Clique}, LB_{RE_eval1} and LB_{RE_eval2}. This is because, the incompatibilities are considered at each time interval examined by ER. In the same manner, LB_{BPPC} improves the classical $LB_{Capacity}$ by taking into account the conflicts between customers. LB_{REBPPC_eval1} and LB_{REBPPC_eval2} produced improved lower bounds for 6 instances compared to the maximum bound that we can obtain using LB_{Clique} and $LB_{Capacity}$. These results confirm that the association of ER to BPPC is very efficient for VRPTW. To conclude, the overall performance of the lower bounding procedures has been encouraging. LB_{BPPC} gave good results in capacity constrained problems while LB_{REBPPC_eval1} and LB_{REBPPC_eval2} improves many lower bounds of time constrained problems. A challenging area for future research is the development of lower bounds that work well for problems that are equally time windows and capacity constrained.

5.6 Conclusion

In this chapter, we introduced several combinatorial optimization methods which can be used to get lower bounds for the Vehicle routing problem with time windows (VRPTW). Investigating the concept of Energetic Reasoning, we were able to propose new lower bounding techniques based on the transformation of m-VRPTW instance to PMSP problem. The numerical results confirm the contribution brought by the new proposed techniques. With a very fast computing time, we were able to prove the optimality of several heuristic solutions and provide a reasonable approximation of the optimal vehicles number required to visit all customers. This suggests that our lower bounds techniques can produce a good estimation of the fleet size quickly. The next step is to develop an exact algorithm (dynamic programming or branch-and-bound) using the approaches presented in this work to find optimal solutions.

Instance	LB_{Clique}		LB_{RE_eval1}		LB_{RE_eval2}		LB_{REBPPC_eval1}		LB_{REBPPC_eval2}		$LB_{Capacity}$		LB_{BPPC}		LB	UB
	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU				
c101	10	0,08	10	0,57	10	0,56	10	2,99	10	2,37	10	0	10	0,01	10*	10
c102	9	0,07	8	1,25	9	1,17	9	1720,68	9	1629,16	10	0	10	0,02	10*	10
c103	7	0,04	8	1,61	8	1,45	8	1468,63	8	1352,98	10	0	10	0,01	10*	10
c104	4	0,02	8	1,49	8	1,32	8	926,2	8	816,9	10	0	10	0,02	10*	10
c105	5	0,03	10	0,57	10	0,5	10	77,95	10	271,93	10	0	10	0,02	10*	10
c106	6	0,03	9	0,59	10	0,53	10	63,1	10	18,4	10	0	10	0,02	10*	10
c107	1	0,01	9	0,77	9	0,63	9	2147,04	9	1243,9	10	0	10	0,02	10*	10
c108	1	0,01	9	0,98	9	0,92	9	763,58	9	547,66	10	0	10	0,02	10*	10
c109	1	0	8	0,91	8	0,83	8	5441,78	8	4011,96	10	0	10	0,01	10*	10
c201	2	0	3	0,44	3	0,39	3	0,08	3	0,08	3	0	3	0,02	3*	3
c202	2	0	3	0,52	3	0,52	3	0,01	3	0,02	3	0	3	0,02	3*	3
c203	2	0	3	0,52	3	0,47	3	0,02	3	0,03	3	0	3	0,02	3*	3
c204	1	0	3	0,61	3	0,58	3	0,02	3	0,02	3	0	3	0,01	3*	3
c205	1	0	3	0,4	3	0,38	3	0,05	3	0,05	3	0	3	0,02	3*	3
c206	1	0	3	0,43	3	0,44	3	0,07	3	0,12	3	0	3	0,03	3*	3
c207	1	0	3	0,42	3	0,41	3	0,07	3	0,07	3	0	3	0,03	3*	3
c208	1	0,01	3	0,34	3	0,31	3	0,05	3	0,07	3	0	3	0,02	3*	3
r101	18	0,23	15	4,41	15	3,65	18	104,45	18	178,2	8	0	18	0,03	18	19
r102	17	0,15	13	6,27	12	4,48	17	70,98	17	330,92	8	0	17	0,02	17*	17
r103	13	0,07	8	4,52	8	4,42	13	0,41	13	0,45	8	0	13	0,02	13*	13
r104	8	0,04	8	1,34	8	1,28	8	782,94	8	826,48	8	0	8	0,01	8	9
r105	7	0,14	11	3,33	11	2,6	11	11349,9	11	8620,3	8	0	8	0,01	11	14
r106	7	0,11	9	3,19	9	2,91	9	6979,6	9	6491,38	8	0	8	0,02	9	12
r107	6	0,06	8	1,99	8	1,95	8	2040,94	8	1899,86	8	0	8	0,02	8	10
r108	5	0,03	8	1,3	8	1,24	8	490,29	8	498,02	8	0	8	0,01	8	9
r109	4	0,05	9	1,71	9	1,55	9	1990,73	9	1857,15	8	0	8	0,02	9	11
r110	3	0,03	8	1,54	8	1,44	8	955,89	8	844,12	8	0	8	0,02	8	10
r111	6	0,05	8	1,84	8	1,83	8	2459,38	8	2269,61	8	0	8	0,01	8	10
r112	1	0	8	1,06	8	1,03	8	519,59	8	479,96	8	0	8	0,01	8	9
r201	2	0	3	1,15	3	1,15	3	277,34	3	302,54	2	0	2	0,01	3	4
r202	2	0	2	1,36	2	1,29	2	285,34	2	226,89	2	0	2	0,02	2	3
r203	1	0	2	1,29	2	1,27	2	181,65	2	217,95	2	0	2	0,02	2	3
r204	1	0	2	0,66	2	0,64	2	0,01	2	0,03	2	0	2	0,01	2*	2
r205	1	0	2	0,88	2	0,83	2	163,35	2	151,81	2	0	2	0,02	2	3
r206	1	0,01	2	1,16	2	1,09	2	204,64	2	215,15	2	0	2	0,02	2	3
r207	1	0	2	0,59	2	0,59	2	0,01	2	0,01	2	0	2	0,02	2*	2
r208	1	0	2	0,61	2	0,62	2	0,01	2	0,01	2	0	2	0,01	2*	2
r209	1	0	2	0,87	2	0,83	2	202,51	2	192,65	2	0	2	0,02	2	3
r210	1	0	2	1,04	2	1,02	2	226,08	2	228,63	2	0	2	0,02	2	3
r211	1	0	2	0,4	2	0,4	2	0,86	2	1,08	2	0	2	0,02	2*	2
rc101	8	0,21	11	2,44	11	1,96	11	28317,4	11	23560,4	9	0	9	0,02	11	14
rc102	7	0,13	9	2,51	9	2,21	9	30014,7	9	23113,3	9	0	9	0,01	9	12
rc103	7	0,08	8	2,21	8	2,11	8	9696,94	8	7828,77	9	0	9	0,01	9	11
rc104	6	0,02	7	1,78	7	1,68	7	1476,53	7	1376,3	9	0	9	0,02	9	10
rc105	12	0,11	9	2,9	9	2,6	12	27152,8	12	20416,6	9	0	12	0,01	12	13
rc106	4	0,13	9	1,24	9	1	9	1749,83	9	1317,5	9	0	9	0,02	9	11
rc107	4	0,05	8	1,58	8	1,38	8	1565,92	8	1367,51	9	0	9	0,02	9	11
rc108	3	0,02	7	1,55	7	1,38	7	1520,95	7	1293,38	9	0	9	0,02	9	10
rc201	1	0	3	0,98	3	0,91	3	263,11	3	280,27	2	0	2	0,01	3	4
rc202	1	0	2	1,11	2	1,07	2	265,43	2	260,78	2	0	2	0,02	2	3
rc203	1	0	2	0,95	2	0,92	2	209,49	2	188,1	2	0	2	0,02	2	3
rc204	1	0	2	0,87	2	0,87	2	314,65	2	263,38	2	0	2	0,02	2	3
rc205	2	0,01	2	1,47	2	1,46	2	584,4	2	568,62	2	0	2	0,02	2	4
rc206	1	0	2	0,7	2	0,68	2	165,55	2	150,75	2	0	2	0,01	2	3
rc207	1	0	2	0,72	2	0,73	2	196,4	2	187	2	0	2	0,02	2	3
rc208	1	0	2	0,63	2	0,61	2	131,75	2	119,98	2	0	2	0,02	2	3

Table 5.1: Lower bound results and CPU times

Conclusion

Dans cette thèse, nous avons mis en place des outils de capitalisation d'informations et d'aide à la décision intervenant dans le cadre de la réduction de l'empreinte écologique : économies d'énergie et réduction des émissions de Gaz à Effet de Serre. Un éventail large de domaines tels que le transport de marchandises, les déplacements des personnes, la réhabilitation des bâtiments a été considéré. Les problèmes rencontrés en pratique nous ont conduits à nous intéresser aux méthodes d'optimisation liées aux problèmes de tournées de véhicules. Notre objectif réside dans le développement de méthodes de résolution efficaces qui tirent profit des travaux menés en optimisation combinatoire et qui peuvent être intégrées au sein d'outils d'aide à la décision adaptés aux caractéristiques et spécificités du cas réel.

Tout d'abord, nous avons présenté des contributions apportées pour la résolution de problèmes de tournées de véhicules sélectives plus ou moins contraints. Notre approche est basée sur une méthode d'extraction. Il s'agit de représenter une solution d'un problème d'optimisation comme une formulation d'un problème plus simple à résoudre. En général, le sous problème identifié peut être résolu en un temps polynomial. Cette approche permet d'exploiter des cas particuliers spécifiques et pertinents du problème traité pour élaborer des méthodes de résolution qui s'avèrent particulièrement efficaces.

Dans un premier temps, nous nous sommes intéressés au problème de tournées avec profits (TOP). L'approche d'extraction proposée est connue dans la littérature pour les problèmes de tournées, sous la dénomination *découpage optimal*. Nous avons développé une procédure de découpage optimal dédiée au TOP. En particulier, nous avons exploité plus efficacement la notion de tournées saturées et montré que l'on pouvait améliorer la complexité des découpages existants. Nous avons ensuite proposé une méthode basée sur la programmation dynamique pour résoudre le sous problème extrait en un temps polynomial. Nous avons intégré cette approche d'extraction dans le cadre d'un algorithme basé sur le schéma PSO. Plusieurs composants ont été intégrés au schéma, notamment, l'introduction des *extra solutions* générées par une heuristique itérative randomisée basée sur le principe de construction/destruction. Avec une étude étendue des paramètres, nous avons montré qu'en comparaison à la littérature, le schéma PSO permet d'obtenir de meilleures solutions. En effet, nous avons obtenu toutes les meilleures solutions de la littérature sauf une. De plus, une nouvelle amélioration stricte a été obtenue sur une instance du benchmark. A notre connaissance, il s'agit de la toute première adaptation du schéma PSO pour le TOP avec succès. Nous avons testé notre approche sur de nouvelles instances dérivées des problèmes de TSP et CVRP. Ces

instances comportent de 100 jusqu'à 400 clients et présentent des caractéristiques intéressantes en termes de distribution des clients et génération de profits. La stabilité de notre PSO a été une autre fois confirmée sur l'ensemble des 333 nouvelles instances.

Dans la suite de ce développement, nous avons adapté le nouveau découpage optimal proposé en considérant les fenêtres de temps comme contraintes supplémentaires. Nous avons procédé à l'élaboration d'un algorithme mémétique pour résoudre le TOPTW. Pour tester son efficacité, plusieurs techniques d'optimisation de type recherches locales (shift, swap, destruction/repair, OrOpt et 2Opt*) ont été développées. Les résultats sur les instances du benchmark standard de Solomon et Cordeau ont montré l'efficacité de notre algorithme qui s'est avéré compétitif par rapport aux algorithmes de l'état de l'art avec des améliorations strictes sur plusieurs instances.

Nous nous sommes intéressés également aux problèmes de tournées classiques avec fenêtres de temps. Nous avons tout d'abord proposé un algorithme de prétraitement visant principalement à détecter des infaisabilités pour déduire des bornes inférieures sur le nombre de véhicules requis. Le calcul de la borne est effectué par une adaptation du Raisonnement Énergétique aux problèmes de tournées de véhicules. Cette adaptation tient à la transformation du problème de tournées de véhicules avec fenêtres de temps à un CuSP construit de sorte à ce que les temps de trajets minimaux soient intégrés aux tâches. Sur les instances ciblées du benchmark de Solomon, la méthode proposée nous a permis de prouver l'optimalité d'un certain nombre d'instances.

A la fin de ce mémoire, nous avons montré l'aspect applicatif des développements précédents à travers la réalisation d'un prototype de portail de services mobilité/habitat. L'utilisateur exploite l'outil développé comme une "boîte à idées" lui exposant différents scénarios, améliorant l'empreinte écologique de son profil tout en tenant compte des différentes contraintes. Les détails concernant l'architecture et les développements liés au prototype ont été fournis en annexe. Pour des raisons de confidentialité, nous n'avons pas pu exposer les modèles utilisés ni les algorithmes mis en place.

L'ensemble de nos recherches ont donné lieu à un article publié dans une revue internationale (EJOR - European Journal of Operational Research), à un article publié dans une conférence internationale avec comité de lecture et à deux articles en cours de rédaction. Les contributions présentées dans ce mémoire ouvrent également de nombreuses perspectives de recherches.

Etant donnée la qualité des résultats obtenus avec le schéma PSO, nous envisageons de tester notre approche de découpage sur d'autres variantes de problèmes de tournées en intégrant par exemple les contraintes de synchronisation. Nous souhaitons aussi tester une nouvelle tendance en optimisation qui consiste à utiliser des solveurs de la programmation en nombre entier dans un contexte heuristique pour produire des solutions de bonne qualité dans un temps de calcul restreint.

Le problème du VRPTW a été largement traité dans la littérature. La plupart des méthodes exactes développées se sont intéressées à la minimisation de la somme des distances des tournées mais peu de méthodes cherchent à résoudre optimalement le problème de minimisation du nombre de véhicules. Nous souhaitons améliorer les bornes inférieures proposées afin de les utiliser dans une formulation linéaire en nombres entiers. Nous nous intéressons aussi au développement d'une méthode de résolution basée sur la génération d'inégalités valides exploitant le graphe d'incompatibilités entre clients et la notion de satisfiabilité du raisonnement énergétique.

D'un point de vue expérimental, nous envisageons étendre l'outil développé afin de couvrir les autres périmètres (habitat, consommation). Cette future étape entraînera son amélioration par l'ajout d'autres contraintes dans le modèle et la proposition d'autres algorithmes plus performants exploitant l'ensemble de nos recherches académiques.

Portail du Moteur d'Adaptation

A.1 Cadre général

Le Portail du Moteur d'Adaptation (PMA) est un outil d'aide à la décision conçu pour mobiliser et mettre en valeur les nouvelles compétences et initiatives [23]. Il produit des scénarios d'évolution de la consommation d'un particulier afin de le projeter dans le temps en lui montrant les changements possibles de son empreinte écologique. Le PMA permet de dialoguer avec le monde extérieur, soit de faire le

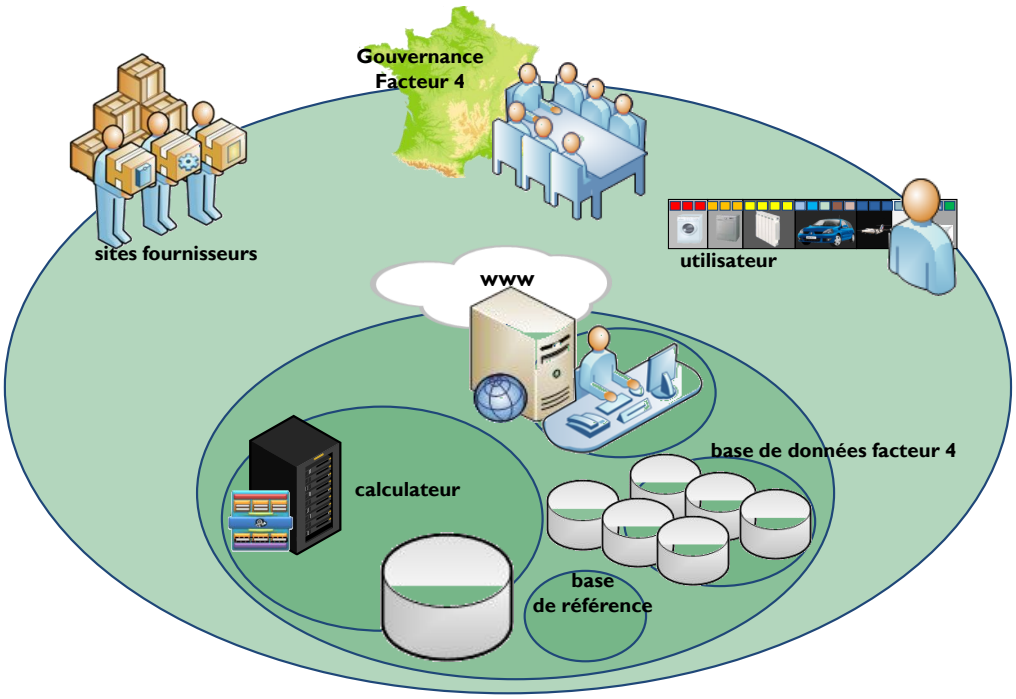


Figure A.1 – Positionnement de l'outil.

lien entre calculateur et consommateur. Il se compose de :

1. Calculateur de scénarios : il s'agit de l'ensemble des méthodes de manipulation des profils et production des plans d'amélioration. Son implémentation repose

sur des modèles issus de la Recherche Opérationnelle. Il est accessible via une interface objet à exploiter pour mettre en œuvre les différentes fonctionnalités.

2. Base de données facteur 4 (BDF4) : permet de sauvegarder l'ensemble des informations fournies par l'utilisateur sur les moyens qu'il utilise. Elle est par la suite enrichie avec différents moyens alternatifs proposés par des sites fournisseurs. On qualifie de fournisseur, toute entité capable de proposer des produits et des services élaborés dans une démarche de réduction de l'empreinte écologique. Ces produits et services sont référencés et caractérisés dans la base de donnée sous forme de composants. Un composant est la brique élémentaire d'un profil. Sa structure permet de collecter les informations nécessaires à la construction de scénarios manipulables par le calculateur.
3. Base de référence : contient les procédures de calcul de l'empreinte écologique associée aux différents composants sauvegardée dans la BDF4. La détermination de ces procédures de calcul a été effectuée en collaboration avec des organismes experts (ASPA, Gest'Energie).

Nous avons opté pour une architecture orientée services où plusieurs technologies sont manipulées pour faire communiquer les différentes applications A.2.

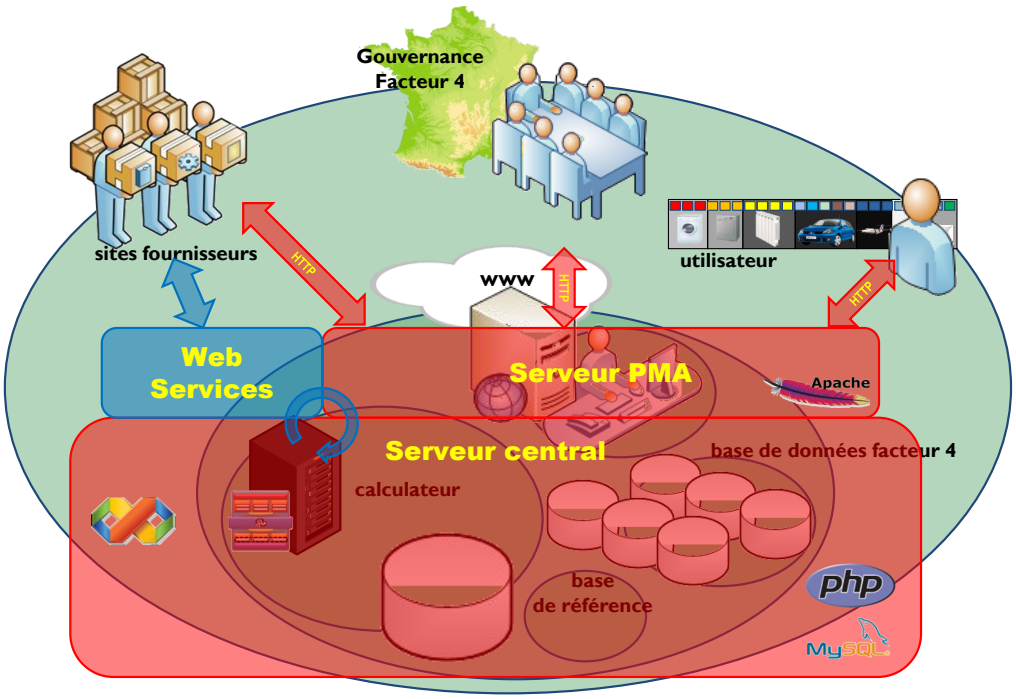


Figure A.2 – Architecture informatique.

L'ensemble des cas d'utilisation à considérer au niveau du PMA est décrit dans la Figure A.3. L'utilisateur crée un compte lui donnant accès à la constitution d'un profil. Il doit remplir un certain nombre de formulaires pour constituer son « profil par défaut », qu'il pourra ensuite affiner et enrichir en intégrant des composants

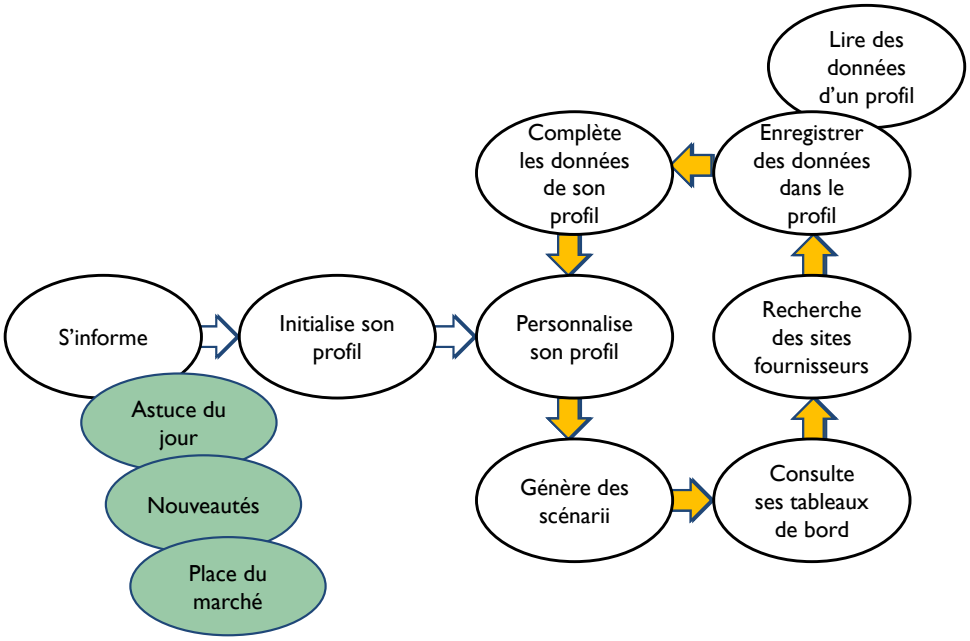


Figure A.3 – Cas d'utilisation.

proposés par des sites fournisseurs. En utilisant ces données, le calculateur propose un pack de solutions sous forme de décisions améliorantes planifiées dans un horizon de temps. L'outil est capable de quantifier les réductions des émissions et la consommation énergétique afin de les valoriser sous forme de points **CoO** (CoO = « Coo »pération et/ou CO_2). Il s'agit d'une monnaie incitative parallèle utilisée pour rémunérer et encourager les efforts réalisés par les particuliers.

A.2 Description du portail

Une fois sur la page d'accueil du site, l'utilisateur crée un nouveau compte s'il s'agit de sa première connexion. Un pseudo et un mot de passe lui seront attribués afin de l'identifier lors de ses futurs accès. Un ensemble de fonctionnalités est mis à sa disposition pour personnaliser son profil et consulter les différentes offres disponibles. A travers la cartographie du PMA, il peut visualiser à la fois les points particuliers du profil (le domicile, les lieux de travail et les lieux d'activités des divers membres adhérents), ainsi que des points généraux propres à la localisation du territoire (les commerces locaux, les lieux d'auto partage, les parkings, les points relais). L'utilisateur décrit l'ensemble de ses déplacements à travers un agenda mobilité A.4. Il peut aussi consulter les détails de chaque activité à réaliser et les alternatives des trajets proposées. On note que les différentes possibilités de déplacement sont fournies par des sites spécialisés en mobilité comme les SIM (Système d'Information

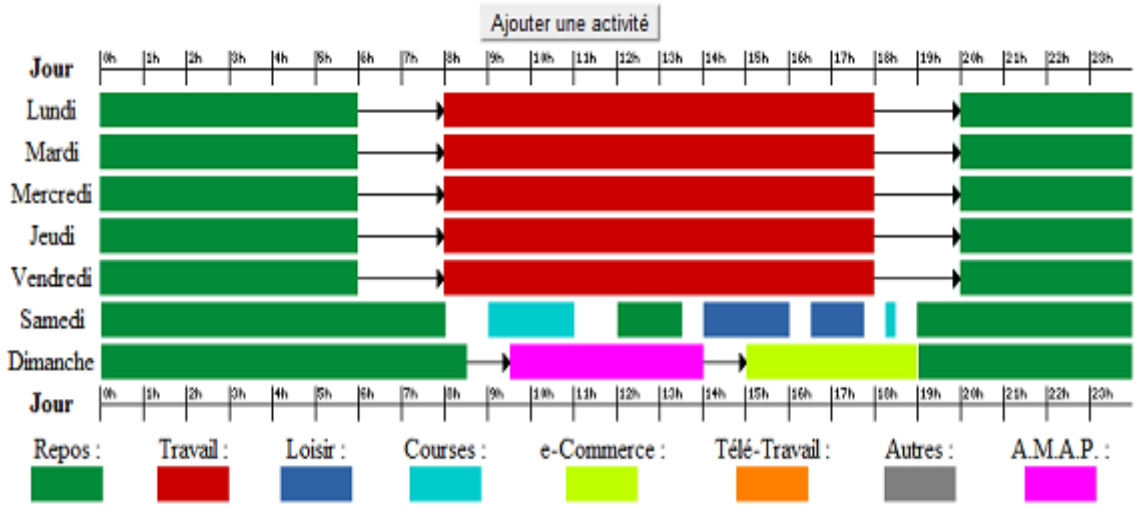


Figure A.4 – Agenda mobilité.

Multimodal) A.5.

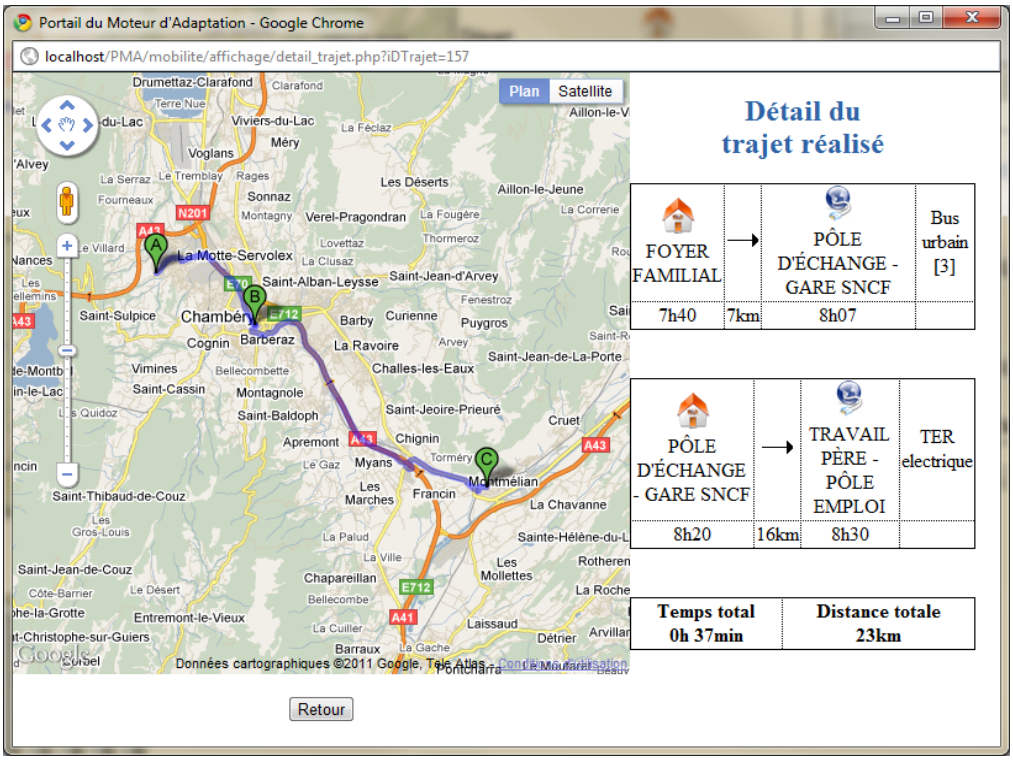


Figure A.5 – Détail d'un trajet.

Un panneau d'identification habitat permet d'amorcer la description de la maison de l'utilisateur en précisant le type du bâtiment, les installations et les équipements existants A.6.

Une fois l'ensemble de ces éléments renseignés, un premier calcul est lancé, conduisant à l'écran de la Figure A.7. L'affichage du résultat est quasi instantané. Cette interface donne des conseils sur les décisions à considérer dans les deux domaines (mobilité, habitat) ainsi que l'ordre dans lequel les travaux doivent être effectués. Un tableau de bord permet de visualiser rapidement l'évolution de l'empreinte dans le temps. Il est possible que l'utilisateur revienne à son profil pour sélectionner ou désactiver chaque offre élémentaire séparément et de visualiser ainsi, de manière immédiate, leur incidence sur l'étiquette énergétique globale.

Figure A.6 – Panneau habitat.

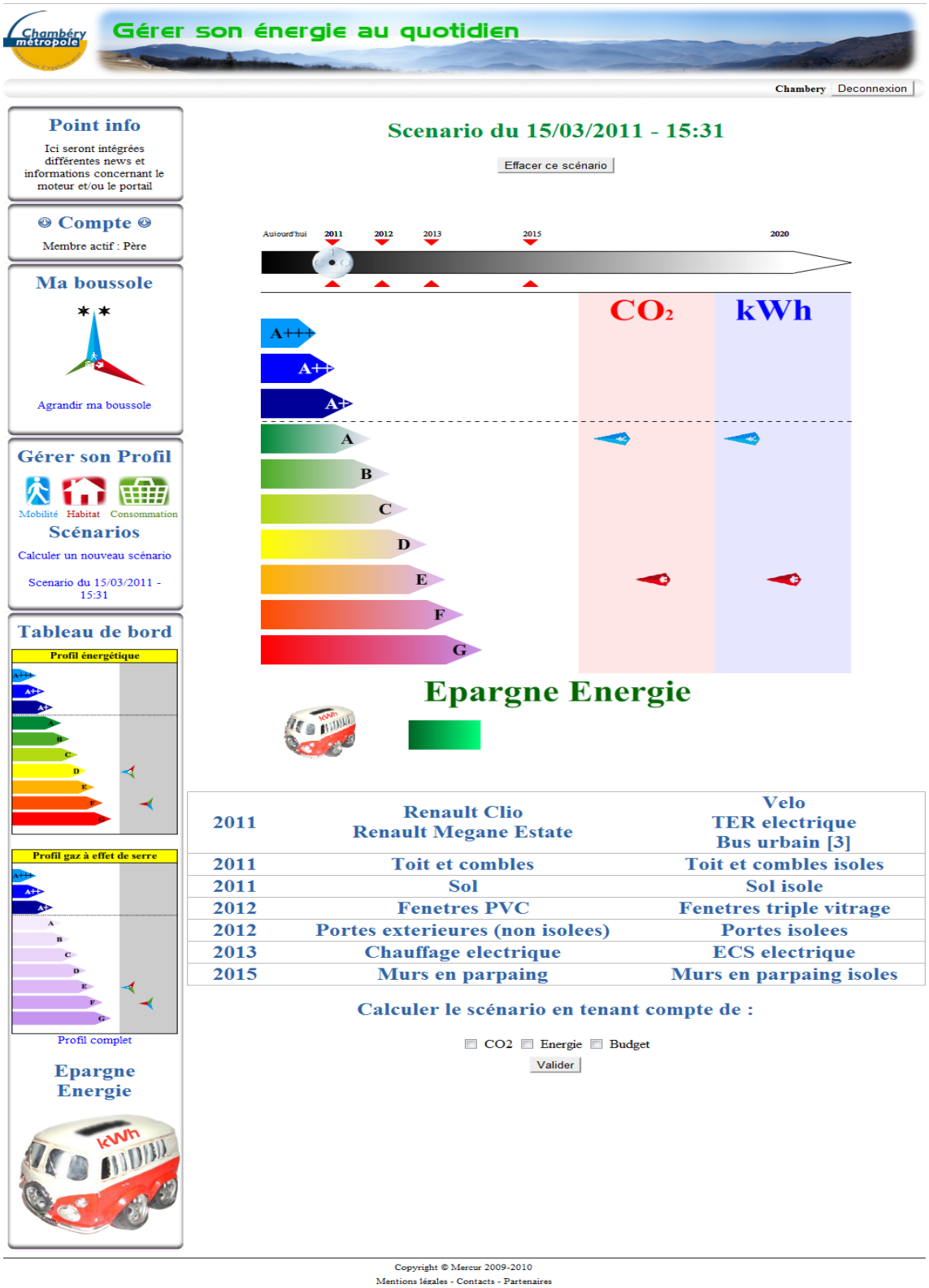


Figure A.7 – Un plan d'amélioration.

Bibliographie

Bibliographie

[1] G. B. Alvarenga, G. R. Mateus, and G. de Tomi. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & amp, Operations Research*, 34(6) :1561 – 1584, 2007.

[2] D. Applegate, R. E. Bixby, V. Chvátal, and J. C. William. Concord TSP solver : http ://www.tsp.gatech.edu/concorde/.

[3] J. Aráoz, E. Fernández, and O. Meza. Solving the prize-collecting rural postman problem. *European Journal of Operational Research*, 196(3) :886–896, 2009.

[4] C. Archetti, A. Hertz, and M. G. Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1), February 2007.

[5] E. Balas and C. Yu. Finding a maximum clique in an arbitary graph. *SIAM Journal of Computing*, 15 :1054–1068, 1986.

[6] R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Oper. Res.*, 59(5) :1269–1283, 2011.

[7] R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1) :1–6, 2012.

[8] A. Banks, J. Vincent, and C. Anyakoha. A review of particle swarm optimization. part I : background and development. *Natural Computing*, 6 (4) :467–484, 2007.

[9] A. Banks, J. Vincent, and C. Anyakoha. A review of particle swarm optimization. part II : hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, 7(1) :109–124, 2008.

[10] P. Baptiste and S. Demassey. Tight lp bounds for resource constrained project scheduling. *OR Spectrum*, 26 :251–262, 2004.

[11] P. Baptiste and C. Le Pape. Constraint propagation and decomposition techniques forhighly disjunctive and highly cumulative project scheduling problems. *Constraints*, 5(1/2) :119–139, 2000.

[12] P. Baptiste, C. Le Pape, and W. Nuijten. Satisfiability tests and time-bound adjustments for cumulative scheduling problems. *Annals of Operations Research*, 92 :305–333, 1999.

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

[14] J. E. Beasley. Route-first cluster-second methods for vehicle routing. *Omega*, 11 :403–408, 1983.

[15] T. Bektas. The multiple traveling salesman problem : an overview of formulations and solution procedures. *Omega*, (34) :209–319, 2006.

[16] J. Berger and M. A. Ayechew. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 31 :2037–2053, 2004.

[17] J. Berger and M. Barkaoui. A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *INFOR*, 41 :179–194, 2003.

[18] J. Berger, M. Salois, and R. Begin. A hybrid genetic algorithm for the vehicle routing problem with time windows. In Robert Mercer and Eric Neufeld, editors, *Advances in Artificial Intelligence*, volume 1418 of *Lecture Notes in Computer Science*, pages 114–127. Springer Berlin / Heidelberg, 1998.

[19] J-F. Bérubé, M. Gendreau, and J-Y. Potvin. An exact epsilon-constraint method for bi-objective combinatorial optimization problems : Application to the traveling salesman problem with profits. *European Journal of Operational Research*, 194(1) :39–50, 2009.

[20] J. Blanton, L. Joe, and R. L. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 452–459, 1993.

[21] C. Blum and A. Roli. Metaheuristics in combinatorial optimization : Overview and conceptual comparison. *ACM Computing Survey*, 35(3) :268–308, 2003.

[22] L. Bonnefoy. L'optimisation par essaims particulaires appliquée au team orienteering problem. Preprint available at : http ://ludovicbonnefoy.files.wordpress.com/2010/10/majecstic2010.pdf, 2010.

[23] P. Botte and H. Bouly. Le moteur de "décarbonation" ou moteur d'adaptation, outil d'aide à la décision pour atteindre le "facteur 4". *Association pour le développement des techniques de transport, d'environnement et de circulation, Paris, FRANCE*, 204 :65–69, 2009.

[24] H. Bouly, A. Moukrim, D. Chanteur, and L. Simon. Un algorithme de destruction/construction itératif pour la résolution d'un problème de tournées de véhicules spécifique. In *MOSIM'08*, 2008.

[25] H. Bouly, D-C. Dang, and A. Moukrim. A memetic algorithm for the team orienteering problem. *4OR*, 8(1) :49–70, 2010.

[26] S. Boussier, D. Feillet, and M. Gendreau. An exact algorithm for team orienteering problems. *4OR*, 5(3) :211–230, 2007.

[27] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part i : Route construction and local search algorithms. *Transportation Science*, 39 (1) :104–118, 2005.

[28] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii : Metaheuristics. *Transportation Science*, 39(1) :119–139, February 2005.

[29] O. Bräysy, W. Dullaert, and M. Gendreau. Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics*, 10 :587–611, 2004.

[30] B. Bullnheimer, R. F. Hartl, and C. Strauss. An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89 :319–328, 1997.

[31] S. Butt and T. Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers and Operations Research*, (21) :101–111, 1994.

[32] S. E. Butt and D. M. Ryan. An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, 26 :427–441, 1999.

[33] I-M. Chao, B. Golden, and E. A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88 :464–474, 1996.

[34] I-M. Chao, B. Golden, and E. A. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88 :475–489, 1996.

[35] N. Christofides, A. Mingozzi, and P. Toth. *Combinatorial Optimization*, chapter The Vehicle Routing Problem, pages 315–338. Wiley, 1979.

[36] J. Clausen. Branch and bound algorithms : principles and examples. *Parallel Computing in Optimization*, pages 239–267, 1997.

[37] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158. ACM, 1971.

[38] J-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2) :105–119, 1997. ISSN 1097-0037.

[39] J-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon F., and Soumis. The vehicle routing problem. chapter VRP with Time Windows, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[40] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[41] J. Z. Czech. Parallel simulated annealing for the vehicle routing problem with time windows. In *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 376–383, 2002.

[42] D-C. Dang and A. Moukrim. Subgraph extraction and metaheuristics for the maximum clique problem. *J. Heuristics*, 18(5) :767–794, 2012.

[43] D-C. Dang, R. N. Guibadj, and A. Moukrim. A pso-based memetic algorithm for the team orienteering problem. In *EvoApplications*, pages 471–480, 2011.

[44] D-C. Dang, R. N. Guibadj, and A. Moukrim. An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 2013.

[45] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1) :80–91, 1959.

[46] G. B. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2 :393–410, 1954.

[47] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3) :387–404, 2008.

[48] C. Duhamel, P. Lacomme, and C. Prodhon. Efficient frameworks for greedy split and new depth first search split procedures for routing problems. *Computers & Operations Research*, 38(4) :723–739, 2011.

[49] J. Edmonds. Covers and packings in a family of sets. *Bulletin of The American Mathematical Society*, 68 :494–499, 1962.

[50] J. Erschler, P. Lopez, and C. Thuriot. Raisonnement temporel sous contraintes de ressources et problï¿½mes d'ordonnancement. *Revue d'Intelligence Artificielle*, 5(3) :7–36, 1991.

[51] D. Feillet. *Routing problems with profits : Study and application to a freight transportation problem*. PhD thesis, Ecole centrale des arts et manufactures, Chatenay-Malabry, France, 2001.

[52] 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) :216–229, 2004.

[53] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6 :109–133, 1995.

[54] M. Fischetti, J. J. S. González, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2) :133–148, 1998.

[55] M. L. Fisherand, K. O. Jörnsten, and O. B. G. Madsen. Vehicle routing with time windows : Two optimization algorithms. *Operations Research*, 45(3) : 488–492, 1997.

[56] A. S. Fraser. Simulation of genetic systems by automatic digital computers. II. Effects of linkage on rates of advance under selection. *Australian Journal of Biological Science*, 10 :492–499, 1957.

[57] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[58] H. Gehring and J. Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN 99, University of Jyväskylä, Finland*, pages 57–64, 1999.

[59] H. Gehring and J. Homberger. Parallelization of a two-phase metaheuristic for routing problems with time windows. *Journal of Heuristics*, 8(3) :251–276, 2002.

[60] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10) :1276–1290, 1995.

[61] B. Gendron, A. Hertz, and P. St-Louis. On edge orienting methods for graph coloring. *Journal of Combinatorial Optimization*, 13 :163–178, 2007.

[62] F. Glover and G. Kochenberger, editors. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.

[63] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[64] B. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34 :307–318, 1987.

[65] B. L. Golden, A. A. Assad, and E. A. Wasil. The vehicle routing problem. chapter Routing vehicles in the real world : applications in the solid waste, beverage, food, dairy, and newspaper industries, pages 245–286. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[66] P. Hansen and N. Mladenovi. Variable neighborhood search : Principles and applications. *European Journal of Operational Research*, 130(3) :449–467, 2001.

[67] J-K. Hao. Memetic algorithms in discrete optimization. In Ferrante Neri, Carlos Cotta, and Pablo Moscato, editors, *Handbook of Memetic Algorithms*, volume 379 of *Studies in Computational Intelligence*, pages 73–94. Springer Berlin / Heidelberg, 2012.

[68] W-K. Ho, J. C. Ang, and A. Lim. A hybrid search algorithm for the vehicle routing problem with time windows. *INTERNATIONAL JOURNAL ON ARTIFICIAL INTELLIGENCE TOOLS*, 10 :431–449, 2001.

[69] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[70] J. Homberger and H. Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR*, 37 :297–318, 1999.

[71] J. Homberger and H. Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1) :220–238, 2005.

[72] H. H. Hoos and T. Stutzle. *Stochastic Local Search : Foundations & Applications (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann, 2004.

[73] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Oper. Res.*, 56(2) :497–511, 2008.

[74] S. Jung and B. R. Moon. A hybrid genetic algorithm for the vehicle routing problem with time windows. In *GECCO 2002 : Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 9-13 July 2002*, pages 1309–1316. Morgan Kaufmann, 2002.

[75] B. Kallehauge. Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & amp ; Operations Research*, 35(7) : 2307 – 2330, 2008.

[76] K. Kameyama. Particle swarm optimization - a survey. *IEICE Transactions*, 92-D(7) :1354–1361, 2009.

[77] M. G. Kantor and M. B. Rosenwein. The orienteering problem with time windows. *The Journal of the Operational Research Society*, 43(6) :629–635, 1992.

[78] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[79] H. Kawamura, M. Yamamot, T. Mitamura, K. Suzuki, and A. Ohuchi. Cooperative search based on pheromone communication for vehicle routing problems. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 81a(6) :1089–1096, 1998.

[80] L. Ke, C. Archetti, and Z. Feng. Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, 54(3) :648–665, 2008.

[81] C. P. Keller. Algorithms to solve the orienteering problem : A comparison. *European Journal of Operational Research*, 41(2) :224–231, 1989.

[82] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1983.

[84] N. Labadie, C. Prins, and M. Reghioui. A memetic algorithm for the vehicle routing problem with time windows. *Rairo-operations Research*, 42 :415–431, 2008.

[85] N. Labadie, J. Melechovský, and R. W. Calvo. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *Journal of Heuristics*, 17(6) :7296–753, 2011.

[86] N. Labadie, R. Mansini, J. Melechovský, and R. W. Calvo. The team orienteering problem with time windows : An lp-based granular variable neighborhood search. *European Journal of Operational Research*, 220(1) :15–27, 2012.

[87] P. Lacomme, C. Prins, and W. Ramdane-Cherif. Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, (131) : 159–185, 2004.

[88] G. Laporte. The traveling salesman problem : An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2) : 231–247, 1992.

[89] G. Laporte, M. Desrochers, and Y. Nobert. Two exact algorithms for the distance-constrained vehicle routing problem. *Networks*, 14(1) :161–172, 1984.

[90] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33 :1050–1073, 1985.

[91] H. C. Lau, M. Sim, and K. M. Teo. Vehicle routing problem with time windows and a limited number of vehicles. *European Journal of Operational Research*, 148(3) :559–569, 2003.

[92] A. C. Leifer and M. B. Rosenwein. Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research*, 73(3) : 517–523, 1994.

[93] J. K. Lenstra, A. H. Kan, and G. Rinnooy. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2) :221–227, 1981.

[94] A. Lim and Z. Xingwen. A two-stage heuristic for the vehicle routing problem with time windows and a limited number of vehicles. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, page 82c, jan. 2005.

[95] S-W. Lin and V. F. Yu. A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 217(1) :94–107, 2012.

[96] J. D. C. Little, K. G Murty, D. W. Sweeney, and C. Karel. An algorithm for the traveling salesman problem. 11(6) :972–989, 1963.

[97] R. Montemanni and L. Gambardella. Ant colony system for team orienteering problems with time windows. *Foundations of Computing and Decision Sciences*, 34(4) :287–306, 2009.

[98] P. Moscato. New ideas in optimization. chapter Memetic algorithms : a short introduction, pages 219–234. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[99] S. Muthuswamy and S. Lam. Discrete particle swarm optimization for the team orienteering problem. *Memetic Computing*, 3 :287–303, 2011.

[100] Y. Nagata and O. Bräysy. A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters*, 37 (5) :333–338, 2009.

[101] E. Néron, P. Baptiste, and J. N. D. Gupta. Solving hybrid flow shop problem using energetic reasoning and global operations. *Omega*, 29(6) :501–511, 2001.

[102] B. Ombuki, B. J. Ross, and F. Hanshar. Multi-objective genetic algorithms for vehicle routing problem with time windows. *APPLIED INTELLIGENCE*, 24 :17–30, 2006.

[103] I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41 :421–451, 1993.

[104] K-W. Pang. An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints. *Expert Syst. Appl.*, 38(9) : 11939–11946, 2011.

[105] M. Pant, R. Thangaraj, and A. Abraham. A new pso algorithm with crossover operator for global optimization problems. In *Innovations in Hybrid Intelligent Systems*, pages 215–222. Springer Berlin Heidelberg, 2008.

[106] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Dover Publications, 1998.

[107] P. M. Pardalos and M. G. C. Resende, editors. *Handbook of Applied Optimization*. Oxford University Press, USA, 2002.

[108] M. Poggi de Aragão, H. Viana, and E. Uchoa. The team orienteering problem : Formulations and branch-cut and price. In *ATMOS*, pages 142–155, 2010.

[109] J-Y. Potvin and S. Bengio. The vehicle routing problem with time windows - part ii : Genetic search. *INFORMS Journal on Computing 8*, pages 165–172, 1996.

[110] J-Y. Potvin, T. Kervahut, B-L. Garcia, and J-M. Rousseau. The vehicle routing problem with time windows part i : Tabu search. *INFORMS Journal on Computing*, 8(2) :158–164, 1996.

[111] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & amp, Operations Research*, 31(12) :1985–2002, 2004. ISSN 0305-0548.

[112] C. Prins, N. Labadie, and M. Reghioui. Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research*, 47(2) :507–535, 2009.

[113] G. Reinelt. A traveling salesman problem library. *ORSA Journal on Computing*, 1991.

[114] G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & amp ; Operations Research*, 36(4) : 1191–1203, 2009.

[115] Y. Rochat and R. D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1 :147–167, 1995.

[116] R. Sadykov and F. Vanderbeck. Bin packing with conflicts : a generic branch-and-price algorithm. Preprint accepted for publication in INFORMS Journal on Computing, 2012.

[117] D. Y. Sha and C-Y. Hsu. A hybird particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering*, 51(4) :791–808, 2006.

[118] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2) :254–265, 1987.

[119] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden. A path relinking approach for the team orienteering problem. *Computers & Operations Research*, 37(11) :1853–1859, 2010.

[120] E. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23 :661–673, 1993.

[121] E. D. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J-Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, pages 170–186, 1997.

[122] K. C. Tan, Y. H. Chew, and L. H. Lee. A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Comput. Optim. Appl.*, 34(1) :115–151, 2006.

[123] H. Tang and E. Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computer & Operations Research*, 32 :1379–1407, 2005.

[124] R. E. Tarjan. Graph theory and gaussian elimination. Technical report, Stanford University, 1975.

[125] S. R. Thangiah. Vehicle routing with time windows using genetic algorithms. *Application hanbook of genetic algorithms : new frontiers*, II :253–277, 1995.

[126] P. Toth and D. Vigo, editors. *The vehicle routing problem.* Society for Industrial and Applied Mathematics, 2001.

[127] F. Tricoire, M. Romauch, K. F. Doerner, and R. F. Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37 :351–367, 2010.

[128] T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9) :797–809, 1984.

[129] Z. Ursani, D. Essam, D. Cornforth, and R. Stocker. Localized genetic algorithm for vehicle routing problem with time windows. *Applied Soft Computing*, 11 (8) :5375–5390, 2011.

[130] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. Iterated local search for the team orienteering problem with time windows. *Computers & amp ; Operations Research*, 36(12) :3281–3290, 2009.

[131] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, 196(1) :118 – 127, 2009.

[132] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. Metaheuristics for tourist trip planning. In *Metaheuristics in the Service Industry*, volume 624 of *Lecture Notes in Economics and Mathematical Systems*, pages 15–31. Springer Berlin Heidelberg, 2009.

[133] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem : A survey. *European Journal of Operational Research*, 209(1) :1–10, 2011.

[134] V. Černý. Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1) :41–51, 1985.

[135] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. Heuristics for multi-attribute vehicle routing problems : A survey and synthesis. Technical report, CIRRELT, 2012.

[136] X-P. Wang, C l. Xu, and X-P. Hu. Genetic algorithm for vehicle routing problem with time windows and a limited number of vehicles. In *Management Science and Engineering, 2008. ICMSE 2008. 15th Annual Conference Proceedings., International Conference on*, pages 128 –133, sept. 2008.

[137] T. Yamada, S. Kataoka, and K. Watanabe. Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Information Processing Society of Japan Journal*, 43(9) :2864–2870, 2002.

[138] B. Yu, Z-Z. Yang, and B. Yao. An improved ant colony optimization for vehicle routing problem. *European Journal of Operations Research*, 196(1) :171–176, 2009.

[139] X. Zhao. A perturbed particle swarm algorithm for numerical optimization. *Applied Soft Computing*, 10(1) :119–124, 2010.

Titre :
Problèmes de tournées de véhicules et application industrielle pour la réduction de l'empreinte écologique

Résumé :
Dans cette thèse, nous nous sommes intéressés à la résolution approchée de problèmes de tournées de véhicules. Nous avons exploité des travaux menés sur les graphes d'intervalles et des propriétés de dominance relatives aux tournées saturées pour traiter les problèmes de tournées sélectives plus efficacement. Des approches basées sur un algorithme d'optimisation par essaim particulaire et un algorithme mémétique ont été proposées. Les métaheuristiques développées font appel à un ensemble de techniques particulièrement efficaces telles que le découpage optimal, les opérateurs de croisement génétiques ainsi que des méthodes de recherches locales. Nous nous sommes intéressés également aux problèmes de tournées classiques avec fenêtres de temps. Différents prétraitements ont été introduits pour obtenir des bornes inférieures sur le nombre de véhicules. Ces prétraitements s'inspirent de méthodes issues de modèles de graphes, de problème d'ordonnancement et de problèmes de bin packing avec conflits. Nous avons montré également l'utilité des méthodes développées dans un contexte industriel à travers la réalisation d'un portail de services mobilité.

Mots-clés :
Optimisation combinatoire, problèmes de tournées de véhicules, algorithmes évolutionnaires, découpage optimal, raisonnement énergétique, application industrielle.

Title:
Vehicle routing problems and industrial application to reduce the ecological footprint

Abstract:
In this thesis, we focused on the development of heuristic approaches for solving vehicle routing problems. We exploited researches conducted on interval graphs and dominance properties of saturated tours to deal more efficiently with selective vehicle routing problems. An adaptation of a particle swarm optimization algorithm and a memetic algorithm is proposed. The metaheuristics that we developed are based on effective techniques such as optimal split, genetic crossover operators and local searches. We are also interested in classical vehicle problems with time windows. Various pre-processing methods are introduced to obtain lower bounds on the number of vehicles. These methods are based on many approaches using graph models, scheduling problems and bin packing problems with conflicts. We also showed the effectiveness of the developed methods with an industrial application by implementing a portal of mobility services.

Keyword:
Combinatorial optimization, vehicle routing problems, optimal split, evolutionary algorithms, energetic reasoning, industrial application.