Université de Montréal

Méthodes de résolution exactes et heuristiques pour un problème de tournées de techniciens

par

Ines Mathlouthi

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en informatique

5 mars 2018

SOMMAIRE

Les problèmes de tournées de techniciens (TRSPs pour *Technician Routing and Scheduling Problems*) consistent à planifier les routes d'un groupe de techniciens afin de servir des requêtes de clients à moindre coût. Ce sont des problème d'optimisation NP-difficiles rencontrés dans plusieurs domaines d'application, tels que les télécommunications, les services publics et les opérations de maintenance. Chaque application a ses propres attributs, contraintes et objectifs. Malgré la place qu'occupent les TRSPs dans l'industrie, ils n'ont pas été beaucoup étudié par les chercheurs. En effet, modéliser et résoudre de tels problèmes pour des entreprises posent d'importants défis.

Cette thèse s'intéresse ainsi à la modélisation et à la résolution d'un TRSP multi-attributs (MATRSP) motivé par une application où différents types d'équipements permettant de réaliser des transactions électroniques doivent être servis par des techniciens. Il s'agit de répartir et ordonnancer les appels de service pour la maintenance de ces systèmes, en tenant compte de différents attributs, soit les compétences des techniciens, les priorités des tâches, les fenêtres de temps multiples pour le service, l'inventaire des pièces de rechange, ainsi que les pauses et les heures supplémentaires des techniciens.

Nous présentons tout d'abord un modèle de programmation linéaire en nombres entiers mixtes pour ce problème, qui est ensuite résolu avec un solveur commercial. Puis, un algorithme exact de *branch-and-price* est proposé pour traiter des problèmes de plus grande taille. Enfin, une méta-heuristique est développée combinant des idées provenant de différentes sources et intégrant en particulier une recherche tabou, une mémoire adaptative et une évaluation des solutions basée sur leur coût et leur contribution à la diversité des solutions dans la mémoire.

Mots clés : Logistique, problème de tournées de techniciens, programmation linéaire en nombres entiers mixtes, *branch-and-price*, recherche tabou.

SUMMARY

Technician Routing and Scheduling problems (TRSPs) involve the routing and scheduling of a group of technicians in order to serve requests at minimum cost. These NP-Hard problems are encountered in different domains, like telecommunications, public services, maintenance operations and many others. Each application brings its own attributes, constraints and objectives. Despite the importance of TRSPs in practice, they have not received much attention from researchers, because modeling and solving these problems pose important challenges.

This thesis is about modeling and solving a multi-attribute TRSP (MATRSP) motivated from an application in which various types of electronic transaction equipments must be serviced by technicians. This problem consists of routing technicians to serve maintenance requests, while taking into account many attributes like technician skills, task priorities, multiple time windows, parts inventory, breaks and overtime.

We first develop a mixed integer linear programming model (MIP) for this problem which is then solved directly with a commercial solver. Then, medium-sized problem instances are tackled through a column generation scheme embedded in a branch-and-price algorithm. For instances of practical size, we finally propose a metaheuristic approach that combines ideas from several sources, including a tabu search, an adaptive memory and a solution evaluation based on its cost and its contribution to the memory diversity.

Mots clés: Logistics, technician routing and scheduling problem, mixed integer linear programming, branch-and-price, tabu search.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

DÉDICACES

À mes chers parents,
C'est à vous que je dois cette thèse
et c'est à vous que je la dédie
Je vous aime

Ines

REMERCIEMENTS

Madeleine Ferron disait «dans la vie, les Hommes sont tributaires les uns des autres. Il y a donc toujours quelqu'un ... à remercier ». Je suis tout à fait d'accord car la présence de certaines personnes a été indispensable pour l'aboutissement de ce travail et j'aimerais bien les mettre en avant dans ces remerciements.
Tout d'abord, je remercie messieurs les membres du jury pour le grand honneur qu'ils me rendent d'évaluer ce travail, qu'ils trouvent tous ici l'expression de mon profond respect et de ma gratitude. Je remercie également mes directeurs de thèse Jean-Yves Potvin et Michel Gendreau, qui m'ont guidé et qui n'ont jamais lésiné afin de me gratifier de leurs précieux conseils. Par leur amabilité et leur modestie de grands hommes, ils me resteront un modèle.

Je tiens à remercier le CIRRELT et le DIRO pour leur accueil et les conditions de travail ainsi que leurs équipes administratives et techniques. En particulier Nath, Lucie, Johanne, Éric, Céline et Véronique pour leur serviabilité et leur écoute. Ce fut un grand plaisir d'échanger avec vous. Mes remerciements vont aussi aux indispensables Serge, Guillaume et Pierre pour leur aide, et surtout leur patience quand certaines de mes tâches causaient des débordements à la grille.

Un grand MERCI à ma famille, mes amis et mes collègues qui, avec leur question, « quand est-ce que tu soutiens cette thèse ? », m'ont permis de ne pas dévier de mon but final. Merci à Ilyes, Nahla, Ghada, Chedlia, Amira, Dorsaf, Fatma, Khadija, Zizou, Khalil et Chahd, qui par leur présence ont égayé mes longues nuits de labeur. Qu'ils trouvent ici l'expression de ma reconnaissance.
À Mohamed pour son soutien et son encouragement. Que ce travail soit un témoignage de ma reconnaissance et de mon grand amour.

INTRODUCTION

Contexte et défis

Plusieurs industries offrent un service de maintenance ou un service après-vente pour leurs clients et c'est souvent ce qui témoigne de la qualité d'une entreprise aux yeux du client. Offrir ces services peut toutefois se révéler une arme à double tranchant, car d'un côté elle séduit les clients et les fidélise et d'un autre côté elle engendre des coûts importants. C'est ainsi que les problèmes de tournées de techniciens (TRSPs pour *Technician Routing and Scheduling Problems*) sont apparus afin de fidéliser les clients en leur offrant un service de maintenance efficace et rapide tout en minimisant les couts engendrés.

Les TRSPs font partie de la classe des problèmes de tournées de véhicules (VRPs pour *Vehicle Routing Problems*) et s'apparentent en particulier aux VRPs avec fenêtres de temps (VRPTWs pour *Vehicle Routing Problems with Time Windows*). Ils ont cependant leurs caractéristiques propres comme les compétences des techniciens pour les différentes tâches, l'inventaire des pièces de rechange et les temps de service qui sont assez longs en comparaison des temps de déplacement.

Les VRPs sont des problèmes NP-difficiles dont l'étude a mené à des milliers de publications dans la littérature scientifique. Certains chercheurs ont opté pour des méthodes exactes tels que le *branch-and-bound* ou le *branch-and-price*. D'autres ont sacrifié l'optimalité pour l'efficacité en proposant des méta-heuristiques. Ces dernières se sont révélées très performantes. Malgré l'intérêt des chercheurs pour les VRPs, et leurs similitudes avec les TRSPs, malgré aussi l'importance des TRSPs en pratique, ils n'ont fait l'objet que de peu de travaux.

Chaque problème de type TRSP rencontré en pratique a ses caractéristiques spécifiques et les algorithmes développés doivent en tenir compte, que ce soit en rapport avec l'objectif (coût, temps, distance, gain total associé aux tâches desservies) ou les attributs présents comme les priorités des clients, les temps de service, l'inventaire des pièces de rechange, les pauses, les fenêtres de temps, les compétences des techniciens, etc. Les problèmes de

tournées de techniciens multi-attributs (MATRSP) posent ainsi des défis majeurs, tant par leur variété que leur difficulté intrinsèque.

Objectif, démarche et contributions

L'objectif principal de cette thèse et de résoudre un TRSP inspiré d'une application réelle. Pour y arriver, plusieurs étapes intermédiaires ont été nécessaires en commençant par une définition et une modélisation appropriées du problème. C'est ainsi que les contributions de cette thèse se divisent en trois axes principaux.

La première contribution présente un modèle de programmation en nombres entiers mixtes (*MIP pour Mixed Integer Program*) intégrant les principales caractéristiques de l'application réelle. Ce modèle est ensuite résolu avec un solveur commercial. Cette contribution met l'accent sur la difficulté du problème et sur l'impact des différents attributs sur sa complexité.

La deuxième contribution présente un algorithme de *branch-and-price* pour le problème. Lors de la génération de colonnes, le problème maître est un problème de *set-packing* (étant donné qu'il n'est pas nécessaire de servir toutes les tâches) où les variables de décision correspondent aux routes des techniciens. Chaque sous-problème est ensuite associé à un technicien et correspond à un problème de plus court chemin élémentaire avec contraintes de ressources (ESPPRC pour *Elementary Shortest Path Problem with Resource Constraints*). Deux approches ont été utilisées afin de résoudre les ESPPRCs, qui sont des problèmes NP-difficiles, soit l'algorithme classique de programmation dynamique de Feillet [**22**] et une relaxation de ce dernier appelée DSSR (pour *Decremental State-Space Relaxation*) [**57**]. Afin d'obtenir l'intégralité des variables, nous proposons deux types de branchements qui exploitent les particularités de notre problème.

La dernière contribution propose une méta-heuristique capable de résoudre des problèmes de taille plus réaliste. Cette méta-heuristique repose sur une recherche tabou et inclut une mémoire adaptative ainsi qu'une évaluation des solutions qui tient compte de leur coût et de leur contribution à la diversité de la mémoire. Cette dernière évaluation a été proposée par Vidal [**70**] dans le cadre d'un algorithme génétique pour résoudre des problèmes de VRPs et elle a donné d'excellents résultats.

Organisation du document

L'organisation de ce document correspond à la démarche présentée plus haut. Le Chapitre 1 présente d'abord une revue des problèmes apparentés aux TRSPs, en commençant par le problème canonique du voyageur de commerce (TSP pour *Traveling Salesman Problem*), suivi de quelques variantes du VRP pour arriver enfin aux problèmes de tournées de techniciens. Le Chapitre 2 correspond au premier article accepté dans la revue *Information Systems and Operational Research*. On y présente la modélisation mathématique de notre problème et sa résolution avec un solveur commercial sur des instances de différents types. Le Chapitre 3 correspond au second article soumis à *Journal on Vehicle Routing Algorithms*. Il présente tous les détails de notre algorithme de branch-and-price et compare les résultats obtenus avec ceux du premier article. Le Chapitre 4 correspond au troisième article soumis à *Computers and Operations Research*. On y présente les différentes composantes de notre approche métaheuristique et on y rapporte des résultats sur des instances de tailles plus réalistes. Enfin, le document se termine par une conclusion qui propose différentes avenues de recherche.

Chapitre 1

ÉTAT DE L'ART

Ce chapitre présente un survol des problèmes de tournées qui intéressent les chercheurs depuis des décennies. Nous commençons par le problème classique du voyageur de commerce et poursuivons avec les problèmes de tournées de véhicules avant d'aborder le problème de tournées de techniciens qui fait l'objet de cette thèse. Le chapitre décrit brièvement ces problèmes tout en mettant l'accent sur quelques méthodes de résolution.

1.1. Le problème du voyageur de commerce

Le problème du voyageur de commerce (TSP pour *Traveling Salesman Problem*) est l'un des problèmes les plus anciens et les plus connus en optimisation combinatoire. Sa définition est la suivante : un commerçant, partant de sa ville de résidence, doit visiter un certain nombre de clients, chacun situé dans une ville différente, avant de revenir à son point de départ. L'objectif du commerçant est de minimiser la distance totale parcourue.

Plus formellement, le TSP est défini sur un graphe non orienté où les nœuds représentent les villes (clients) et les arêtes liant deux villes sont pondérées par une distance ou longueur. L'objectif est de trouver un cycle Hamiltonien (où chaque nœud est visité exactement une fois) de longueur minimale. Une première description théorique du problème est donnée par J.B. Robinson [58] à la fin des années 40. Plus tard, G.B. Dantzig et al. [14] proposent en 1954 une solution pour ce problème sur une instance de 49 villes en faisant appel à la programmation linéaire.

Pour résoudre un TSP, qui est un problème NP-difficile, on peut procéder de trois façons :

— utiliser une approche énumérative qui consiste à considérer tous les cycles Hamiltoniens et à identifier celui qui est de longueur minimale. Cette approche est toutefois de complexité $O(n!)$ (où n est le nombre de nœuds) ce qui devient vite prohibitif ;

— utiliser une méthode exacte, comme la programmation dynamique qui permet de résoudre le problème en $O(n^3 2^n)$ [34], ce qui est encore de complexité exponentielle ;

— sacrifier l'optimalité pour l'efficacité en appliquant des approches heuristiques telles que l'heuristique d'amélioration de Lin et Kernighan [20] ou encore des métaheuristiques, comme la recherche tabou ou les algorithmes génétiques [10].

1.2. Le problème de tournées de véhicules

Un problème de tournées de véhicules VRP (pour *Vehicle Routing Problem*) peut être vu comme plusieurs TSPs où chaque commerçant est représenté par un véhicule. En effet, il s'agit ici d'identifier un ensemble de tournées de coût minimal permettant de desservir l'ensemble des clients tout en satisfaisant un certain nombre de contraintes. Le coût peut correspondre à la distance totale parcourue par les véhicules, au temps de parcours, au nombre de véhicules utilisés, etc. (ou à une combinaison de ces critères). Le VRP est évidemment un problème NP-difficile, puisqu'il s'agit d'une généralisation du TSP.

Formellement, le VRP est défini sur un graphe orienté G=(V,A) avec [30] :

— $V = \{V_0, .., V_n\}$ un ensemble de nœuds ou V_0 représente le dépôt central et $V \backslash \{V_0\}$ représente l'ensemble des clients ;

— $A = \{(v_i,v_j) : v_i,v_j \in V, i \neq j\}$ un ensemble d'arcs.

La flotte de véhicules est homogène et le nombre de véhicules disponibles k est fixé a priori, laissé libre ou majoré par une constante. Une matrice des coûts $C = \{c_{ij}\}$ est également définie sur A. Le VRP consiste alors à identifier un ensemble de tournées de coût minimal, où chaque véhicule exécute au plus une tournée, tel que :

— chaque tournée commence et se termine au dépôt ;

— chaque client est visité une et une seule fois ;

— la demande totale des clients d'une tournée n'excède pas la capacité du véhicule.

La présence d'une contrainte de capacité fait en sorte qu'on appelle aussi ce problème le CVRP (pour *Capacitated VRP*). Parfois, une contrainte portant sur la longueur maximale des tournées s'ajoute ou remplace la contrainte de capacité.

Le VRP, tel que décrit, est à la source de nombreux problèmes réels en transport de marchandises ou de personnes. D'une application à l'autre, de nouvelles contraintes et de nouveaux besoins surgissent. C'est ainsi que différentes variantes du VRP sont apparues. Pour les fins de la discussion, nous les divisons en trois grandes familles, soit les VRPs statiques, les VRPs dynamiques et les VRPs Multi-Attributs (MAVRPs).

1.2.1. VRPs statiques

Les VRPs classiques sont statiques et déterministes [4], étant donné que toutes les données du problème, tels que les clients et leur demande, sont supposées connues à l'avance (statique) et avec certitude (déterministe). Parmi les VRPs statiques et déterministes les plus connus, il vaut la peine de mentionner :

VRP avec fenêtres de temps Il s'agit ici de problèmes de type CVRP où une fenêtre de temps $[a_i, b_i]$ est associée à chaque client $i \in V \backslash \{V_0\}$. On retrouve donc la contrainte de capacité des CVRPs et une contrainte qui impose que le début du service à chaque client i ait lieu entre les instants a_i et b_i. Le VRPTW (pour *Vehicle Routing Problem with Time Windows*) est équivalent à un CVRP lorsque $a_i = 0$ et $b_i = +\infty$ [38]. Une extension du VRPTW consiste à considérer plusieurs fenêtres de temps disjointes à chacun des clients. De telles fenêtres de temps multiples sont couramment rencontrées dans les problèmes réels. Du point de vue du temps de passage du véhicule, cette caractéristique peut être avantageuse, car elle offre davantage de choix. Du point de vue modélisation et résolution, toutefois, les fenêtres de temps multiples compliquent les choses.

VRP avec retours Le VRPB (pour *Vehicle Routing with Backhauls*) est un CVRP ou l'ensemble des clients est divisé en deux sous-ensembles. Un premier sous-ensemble où l'on doit livrer la marchandise et un deuxième sous-ensemble où l'on recueille la marchandise [66]. Dépendant de la variante considérée, différentes contraintes de précédence entre les deux sous-ensembles de clients doivent être respectées.

VRP avec collectes et livraisons Dans le PDVRP (pour *Pickup and Delivery VRP*), chaque client est associé à un point de collecte (*pickup*) et un point de livraison (*delivery*) avec une contrainte de précédence entre les deux points [61, 65].

1.2.2. VRPs dynamiques

De nos jours, l'exploitation de l'information en temps réel est de plus en plus importante dans le domaine des transports. Ainsi, les DVRPs (pour *Dynamic VRPs*) sont les équivalents dynamiques des VRPs statiques. La différence vient de ce qu'une partie des informations, telle que l'arrivée d'un nouveau client, ne devient disponible qu'au cours de la journée d'opérations et doit être intégrée dans les routes courantes des véhicules.

Pour exploiter de façon pertinente l'information disponible en temps réel, il faut recourir à de nouveaux outils technologiques comme les GPSs (pour *Global Positioning Systems*) [35] qui permettent de localiser les véhicules, ainsi que les GPRSs (pour *General Packet Radio Services*)[42] ou téléphones portables $4G$ qui assurent une communication continue à un coût peu élevé. Pour plus de détails, le lecteur est référé à [49].

1.2.3. VRPs multi-attributs

De nombreux problèmes dans le monde réel peuvent être modélisés comme des VRPs. Toutefois, chaque nouvelle application apporte de nombreuses caractéristiques et contraintes additionnelles qui lui sont propres, telles des contraintes de non compatibilité entre les diverses entités du problème (clients, marchandises, chauffeurs, véhicules), la présence d'une

flotte hétérogène, des fenêtres de temps multiples pour le service, etc. Ces problèmes très complexes sont appelés *Multi-Attribute VRPs* (MAVRPs) ou encore *Rich VRPs* (RVRPs). De tels problèmes posent des défis importants pour les chercheurs.

1.2.4. Quelques méthodes de résolution

Le nombre de méthodes de résolution proposées pour résoudre les différentes variantes des VRPs a explosé au fil des ans. Entre méthodes exactes, heuristiques et méta-heuristiques, chaque chercheur essaie de répondre au mieux aux exigences de son problème (par exemple, temps de calcul versus qualité de la solution).

Dans ce qui suit, nous allons nous concentrer sur quelques méthodes exactes ou heuristiques parmi les plus efficaces pour certaines variantes mentionnées précédemment.

VRP avec contraintes de capacité — Baladacci et al. [3] ont présenté une nouvelle formulation pour ce problème de type *two-commodity network flow*. La relaxation linéaire induit une borne inférieure qui est ensuite intégrée dans un algorithme de *branch-and-cut* ;

- Lysgaard et al. [43] présentent un nouvel algorithme de *branch-and-cut* intégrant différents types de coupes : capacité, capacité généralisée, coupes de Gomory classiques, etc.
- Fukasawa et al. [25] combinent l'algorithme de Lysgaard et al. avec un algorithme de génération de colonnes obtenant ainsi une approche hybride de type *branch-and-cut-and-price* ;
- Baladacci et al. [2] proposent une formulation de partitionnement d'ensemble qui est résolue avec une méthode de génération de colonnes intégrant des coupes de type capacité et cliques ;
- Gendreau et al. [29] résolvent un CVRP avec restriction sur la longueur de chaque route à l'aide d'une recherche tabou. L'espace de recherche autorise les solutions non réalisables qui sont toutefois pénalisées de façon dynamique : la pénalité est augmentée si la solution courante est non réalisable pour un certain nombre d'itérations et elle est diminuée dans le cas contraire.

VRP avec fenêtres de temps — Desrochers et al. [17] proposent un algorithme de *branch-and-price* pour résoudre ce problème. Des routes ou colonnes sont générées en résolvant un problème de plus court chemin avec contraintes de capacité et fenêtres de temps à l'aide de la programmation dynamique ;

- Kohl et al. [40] proposent un algorithme de branch-and-bound où la borne inférieure est calculée par une méthode de sous-gradient ;
- Irnich et Villeneuve [36] présentent un algorithme de *branch-and-price* où le sous-problème est résolu par une méthode d'élimination de k-cycles ;

— Feillet et al. [22] résolvent de manière exacte le problème de plus court chemin élémentaire avec contraintes de ressources (ESPPRC pour *Elementary Shortest Path Problem with Resource Constraints*) à l'aide d'une extension de l'algorithme d'étiquetage précédemment développé pour une version relaxée du problème avec chemins non élémentaires (SPPRC) [36]. Les algorithmes de résolution pour les ESPPRCs et SPPRCs constituant la colonne vertébrale des méthodes de génération de colonnes, Chabrier [9] a par la suite proposé certains raffinements afin d'accélérer la résolution des ESPPRCs, comme une réduction de la taille du graphe et des techniques de sauvegarde des étiquettes. Ces raffinements ont permis de résoudre certaines instances classiques rapportées dans la littérature.
— Jepsen et al. [37] présentent un algorithme de *branch-and-cut-and-price*, en ajoutant au problème maître certaines inégalités valides ;
— Desaulniers et al. [16] modifient l'algorithme de Jepsen et al. [37] en utilisant dans un premier temps la recherche tabou plutôt que la programmation dynamique afin de générer plus rapidement des colonnes de coût réduit négatif.

Il faut noter que toute méthode exacte pour le VRPTW basée sur une formulation de partitionnement d'ensemble peut s'adapter facilement au CVRP (en relaxant la contrainte temporelle lors de la phase de génération de colonnes). Malheureusement, cette adaptation n'est pas vraiment efficace.

VRP avec fenêtres de temps multiples — Wouter et al. [72] optimisent des tournées de sites touristiques (musées, théâtres, etc.) en fonction des intérêts des voyageurs, de leur budget, de la distance, du temps disponible, etc. Il faut aussi prendre en compte les temps d'ouverture et de fermeture des sites qui peuvent être multiples au cours d'une même journée. Le problème est modélisé comme un Orienteering Problem (OP) [68] où il faut trouver le meilleur chemin permettant d'atteindre plusieurs balises dispersées sur un territoire. Chaque balise (site touristique) correspond à un nœud du graphe auquel est associé un score (intérêt du voyageur). Le but est de maximiser le score total à l'intérieur de contraintes temporelles.
Les auteurs abordent ce problème à l'aide d'une méthode de recherche locale itérative. Une hybridation avec la méthode GRASP (*Greedy randomized adaptive search*) permet de renforcer la diversification de la recherche dans l'espace des solutions.
— Doerner et al. [19] s'attaquent à un problème de collecte de sang à différents sites (clients) avec fenêtres de temps multiples et interdépendantes. Ce problème est un VRPTW, mais avec deux particularités. D'abord, le sang est produit de façon continue à chacun des sites au cours de la journée. Ensuite une contrainte est introduite afin que le sang produit à chacun des sites soit retourné au dépôt

à l'intérieur d'un certain temps limite, le sang étant un produit périssable. Cette dernière contrainte crée une interdépendance entre les collectes aux différents sites. Pour calculer le nombre minimum de collectes à un site dans une journée, nous avons :

— a_i et b_i le début et la fin de la journée de production au site i ;
— f_i la durée du service au site i ;
— t_{i0} le temps de parcours entre le site i et le dépôt 0 ;
— τ la durée de conservation du sang.

Ainsi, $t_i = \tau - f_i - t_{i0} - f_0$ est le temps maximal entre deux services consécutifs au site i, en supposant que le véhicule se rend directement au dépôt par la suite. Pour construire les tournées, les auteurs utilisent la mesure $r_i = \lceil \frac{b_i - a_i}{t_i} \rceil$ qui correspond au nombre minimum de collectes au site i durant la journée de production. Cette mesure permet de déterminer des fenêtres de temps à chaque site i pour chaque collecte. Après la création des tournées, la concaténation de certaines tournées est ensuite envisagée, menant ainsi à une nouvelle mise à jour des fenêtres de temps.

VRPs multi-attributs — Dayarian et al. [15] résolvent un problème de collecte et de distribution du lait au Québec. Les auteurs proposent une formulation de type partitionnement d'ensemble où chaque variable correspond à une route (colonne) réalisable. Cette formulation est ensuite résolue à l'aide d'un algorithme de *branch-and-price*. Les sous-problèmes de type ESPPRC sont résolus avec un algorithme de programmation dynamique intégrant une méthode de relaxation à l'étape de l'étiquetage (DSSR pour *Discrete State-Space Relaxation*) [57]. Plus précisément, cette méthode permet la génération de chemins avec des cycles lors de la procédure d'étiquetage. Cette relaxation est progressivement renforcée en identifiant de plus en plus de nœuds dits critiques à chaque itération et en interdisant les visites multiples à ces derniers. Essentiellement, si un chemin ne contient pas de cycles à la fin d'une itération, alors il s'agit d'une solution à l'ESPPRC et la procédure se termine, sinon les nœuds qui sont visités plus d'une fois dans le chemin le moins coûteux sont ajoutés à l'ensemble des nœuds critiques et on passe alors à l'itération suivante ;

— Vidal et al. [70] proposent un algorithme génétique hybride unifié pour les MA-VRPs. Une particularité de cet algorithme est une évaluation des solutions qui combine leur coût et leur contribution à la diversité de la mémoire adaptative. L'approche proposée permet aussi d'ajouter facilement de nouvelles contraintes ou de relaxer les contraintes existantes. Avec une seule implémentation et un seul jeu de paramètres, cet algorithme a égalé ou surpassé les meilleures solutions connues pour 26 variantes de problèmes de tournées de véhicules.

DVRPs Gendreau et al. [28] résolvent un VRP dynamique provenant d'une application pour la livraison du courrier qui doit être livré à l'intérieur de fenêtres de temps bien précises. Les auteurs ont proposé une recherche tabou intégrant une nouvelle structure de voisinage appelée *CROSS exchanges*. Au fur et à mesure que la recherche progresse, les routes des meilleures solutions rencontrées sont stockées dans une mémoire adaptative. Cette mémoire est ensuite utilisée pour créer de nouvelles solutions de départ pour la recherche tabou lorsque cette dernière stagne et ne parvient plus à identifier une meilleure solution.

Pour plus de détails le lecteur peut se référer à la revue de littérature de Pillac et al. [51].

1.3. Problème de Tournée de Réparateur

Le Problème de Tournée de Réparateur (ou TRP pour *Traveling Repairman Problem*) est un problème apparenté au TRSP car il s'intéresse à la visite de clients par un réparateur. Le problème de base n'implique qu'un seul réparateur qui doit répondre à des appels de clients avec pour objectif de minimiser la somme de leurs temps d'attente. Le TRP est un problème NP-difficile en général, mais il peut être résolu en un temps polynomial sur des exemples bien particuliers, comme le TRP défini sur un graphe linéaire [1].

Plusieurs variantes du TRP classique ont été décrites dans la littérature :

WTRP : C'est le TRP où un poids w (pour *weight*) est affecté à chaque client afin de refléter son importance. L'objectif est alors de minimiser la somme des temps d'attente pondérés par les poids [26] ;

TRPTW Dans cette variante, un intervalle temps doit être respecté pour le service à chaque client [24] ;

k-TRP : Il s'agit d'un TRP avec plusieurs réparateurs [21] ;

k-TRP à dépôts multiples : c'est un k-TRP où les tournées des réparateurs débutent à des points de départ différents [21] ;

DTRP : C'est un TRP défini dans un environnement dynamique. Ici, les appels des clients sont reçus tout au cours de la journée et doivent être intégrés à la tournée courante du réparateur [18].

1.4. Problèmes de Tournées de Techniciens

Dans plusieurs domaines, les entreprises offrent des services après-vente, comme la maintenance de leurs produits. Or, gérer de tels services de façon efficace n'est pas évident. C'est pourquoi le Problème de Tournées de Techniciens (TRSP pour *Technician Routing and Scheduling Problem*) a séduit certains chercheurs [73].

Le TRSP peut être défini comme suit. Soit un ensemble de techniciens ayant chacun des compétences pour des tâches bien définies auxquelles on attache possiblement des niveaux de priorité différents. Soit aussi un ensemble de clients où chaque client requiert les services d'un technicien ayant la compétence pour effectuer la tâche demandée. L'objectif est alors d'affecter les techniciens aux tâches et de construire la route de chaque technicien en ordonnançant les tâches qui lui sont affectées de façon à minimiser un certain objectif, comme la distance totale parcourue. Il faut aussi que les routes soient réalisables et qu'elles satisfassent les contraintes touchant aux compétences des techniciens, ainsi que des contraintes additionnelles comme le nombre d'heures de travail maximal dans une journée et les fenêtres de temps pour le service aux clients. Parfois, les compétences des techniciens pour les différentes tâches s'expriment par des degrés d'habileté (plutôt que par des contraintes) et sont alors intégrées à l'objectif. La prise en compte des compétences est ce qui distingue principalement les TRSPs des TRPs. Aussi, les articles traitant du TRP s'intéressent la plupart du temps à la tournée d'un seul réparateur (bien qu'on retrouve quelques articles traitant du k-TRP où plusieurs réparateurs sont disponibles).

Les problèmes de tournées de techniciens font donc partie de la classe des problèmes de tournées de véhicules et s'apparentent en particulier aux VRPTWs dû à la présence de fenêtres de temps. Ils ont aussi des caractéristiques distinctes comme les compétences des techniciens pour les différentes tâches à réaliser, la disponibilité ou non de pièces de rechange et les temps de service qui sont souvent assez longs en comparaison des temps de déplacement. Les TRSPs sont des problèmes NP-difficiles pour lesquels la résolution d'instances de grande taille en un temps raisonnable constitue un défi.

Dans la suite, nous divisons les TRSPs en trois grandes variantes, soit les TRSPs classiques, de nature statique, les TRSPs dynamiques et les TRSPs avec temps de service stochastiques.

1.4.1. TRSPs classiques

Nous avons choisi cette appellation pour les problèmes comprenant les caractéristiques spécifiées dans la définition de base du problème, telles que mentionnées plus haut. Nous allons présenter les approches rapportées dans la littérature pour les résoudre.

(a) Méthodes exactes

Cortes et al. [8] présentent un problème de tournées de techniciens pour des imprimantes Xerox. L'entreprise reçoit des appels de service de ses clients et doit leur affecter des réparateurs en tenant compte de la priorité de l'appel. Celle-ci dépend de l'importance du client et de la nature de la fenêtre de temps (e.g., fenêtre de taille réduite). Dû à un trop grand nombre de variables, les auteurs ont mis de côté une formulation basée sur les arcs, optant plutôt pour une approche de partitionnement d'ensemble. Une méthode de

génération de colonnes s'inspirant de [74] a donc été développée où le sous-problème est résolu à l'aide de la programmation par contraintes.

Zamorano et Stolletz [75] ont étudié un problème où des équipes de techniciens doivent assurer une maintenance quotidienne de certains équipements sur une période d'une semaine. Un algorithme de *branch-and-price* est proposé pour résoudre le problème. Ce dernier fait appel à deux approches de décomposition : la première repose sur les journées tandis que la deuxième repose sur les équipes.

(b) Méthodes de recherche locale

Tsang et al. [67] ont proposé une méthode de descente locale rapide ainsi qu'une recherche locale guidée pour résoudre un problème rencontré par la compagnie British Telecom où les compétences des techniciens sont représentées par des facteurs d'habileté qui influencent le temps de service. Weigel et al. [71] ont aussi utilisé une descente locale basée sur un voisinage de type Or-Opt [48] pour un problème d'assistance technique. De leur côté, Blackley et al. [5] ont fait appel à une descente locale rapide pour un problème de maintenance périodique des escaliers mécaniques et des élévateurs.

(c) Méthode hybride

Xu et Chiu [73] présentent un problème de tournées de techniciens rencontré dans le domaine des télécommunications. Il s'agit ici d'affecter un ensemble de techniciens ayant différentes compétences à des tâches, tout en tenant compte des plages horaires disponibles. Dans ce problème, il n'est pas nécessairement possible de servir toutes les tâches. Quatre algorithmes sont proposés, en l'occurrence un algorithme glouton, un algorithme glouton amélioré, une méthode de recherche locale et une méthode GRASP.

— ***Algorithme Glouton***
Cet algorithme traite l'ensemble des tâches une à une et insère chaque tâche à coût minimal dans la route d'un technicien. À la fin, le résultat est un ensemble S de triplets (j,k,t). Chaque triplet indique que la tâche $j \in J$ est affectée au technicien $k \in K$ et commence à l'instant t. Dans cet algorithme, w_j est le poids, p_j la durée et l_j la borne supérieure de la fenêtre de temps pour la tâche j. De plus S_{jk} est la compétence du technicien k pour la tâche j.

Algorithme Glouton pour le TRSP (J,K,S).

1. Trier en ordre croissant l'ensemble des tâches J selon le critère w_j/p_j (les tâches ayant la même valeur w_j/p_j sont triées en ordre décroissant selon $l_j - p_j$) ;
2. Pour chaque tâche $j \in J$ faire :

   2.1 Trier l'ensemble des techniciens K selon la valeur S_{jk} (les techniciens ayant la même valeur S_{jk} sont triés en ordre croissant selon la distance qui les sépare de l'endroit où doit s'effectuer la tâche) ;

   2.2 Tant qu'on n'a pas affecté un technicien à la tâche j et il reste un technicien à considérer faire :

   Si la route du technicien k est réalisable pour la tâche j

   Insérer j dans la route du technicien k à coût minimal ;

   $S \longleftarrow S \cup (j,k,t)$;

   Sinon passer au technicien suivant.

— ***Algorithme Glouton Plus***

Pour améliorer l'algorithme glouton de base, on exploite le fait qu'un technicien ne peut accomplir plus de six tâches par jour. L'algorithme Glouton Plus prend donc la solution initiale produite par l'algorithme Glouton, puis identifie la route optimale pour chaque technicien à l'aide d'un algorithme exact de *Branch-and-Bound*.

— ***Méthode de recherche locale***

La méthode de recherche locale modifie la solution obtenue par l'algorithme Glouton Plus à l'aide de quatre opérateurs de voisinage qui modifient la composition des routes des techniciens. Plus précisément, les opérateurs sont :

Ajouter : ajouter une tâche (non encore affectée) à la route d'un technicien ;

Échanger : étant donné deux tâches i et j affectées respectivement aux techniciens k et k', réaffecter j à k et i à k' ;

Déplacer : étant donné une tâche i affectée à un technicien k, réaffecter i à un technicien $k' \neq k$;

Troquer : remplacer une tâche i affectée à un technicien k par une tâche non encore affectée.

— ***GRASP***

La méthode GRASP a donné de bons résultats pour la résolution de plusieurs variantes du problème de tournées de véhicules [**55**]. C'est pour cette raison que les auteurs l'ont aussi appliqué au TRSP. Cette méthode combine des heuristiques gloutonnes randomisées avec des méthodes de recherche locale (voir [**23**] pour une introduction à la méthode GRASP).

Ainsi, l'algorithme Glouton est randomisée de la façon suivante : à chaque itération, une tâche est sélectionnée aléatoirement dans une liste appelée RCL (pour *Restricted Candidate List*) contenant les α tâches de plus grand poids non encore affectées à un technicien. Ensuite, un technicien est choisi pour réaliser la tâche en sélectionnant aléatoirement dans une autre liste RCL pour les techniciens, construite selon le critère utilisé dans l'algorithme Glouton. Une implantation parallèle de la méthode est également décrite dans le but d'améliorer les temps d'exécution.

(d) Recherche adaptative à grand voisinage

Cordeau et al. [11] présentent une heuristique constructive et une méthode de recherche adaptative à grand voisinage (ALNS pour *Adaptive Large Neighborhood Search*) [53] afin de résoudre un problème réel présenté dans le cadre de la compétition française de recherche opérationnelle ROADEF.

Ici, des tâches sont affectées à des équipes de techniciens ayant les compétences requises pour les réaliser sur un intervalle de plusieurs jours, tout en tenant compte de trois niveaux de priorité différents pour les tâches. L'objectif est de minimiser la somme des temps de finalisation de la dernière tâche pour chaque niveau de priorité et pour l'ensemble des tournées. Une heuristique de construction est d'abord utilisée afin d'identifier une première solution réalisable. Cette heuristique fonctionne en deux phases. La première phase consiste à former chaque équipe à partir d'une tâche germe (tout en se gardant une réserve de techniciens). Pour ce faire, les mesures suivantes sont utilisées :

— Mesure critique : mesure l'importance d'effectuer la tâche en premier, en fonction de sa priorité, de sa durée et de ses successeurs ;
— Mesure de difficulté : mesure la difficulté d'identifier une équipe pour cette tâche, en se basant sur les compétences requises ;
— Mesure de similarité : mesure la similitude des compétences requises entre la nouvelle tâche et les tâches déjà choisies

Ces trois mesures sont combinées afin d'obtenir un *score* et c'est la tâche germe ayant le *score* le plus élevé qui est choisie. Une équipe ayant les compétences requises pour la réaliser est ensuite formée. Dans la deuxième phase, les tâches restantes sont affectées aux équipes existantes, en se basant sur une mesure des compétences perdues. Au cours de ce processus, il est possible d'adjoindre un ou plusieurs techniciens à une équipe existante afin de lui permettre de réaliser une tâche. S'il reste des tâches à la fin et qu'aucune équipe ne peut les réaliser, celles-ci sont alors considérées pour le jour suivant.

La solution générée par l'heuristique de construction est ensuite améliorée par une méthode de destruction et reconstruction semblable à la méthode ALNS rapportée dans [60].

Pour choisir les tâches à retirer de la solution courante, quatre méthodes de destruction sont proposées :

— destruction aléatoire ;
— destruction avec dépendances : ici on désire retirer des tâches présentant des similarités au niveau des compétences requises.
— destruction semi-aléatoire, semi-dépendante ;
— destruction d'une équipe et de toutes les tâches qui lui ont été affectées.

Pour réparer les solutions détruites, l'heuristique de construction présentée plus haut est appliquée aux tâches qui ont été retirées de la solution. A chaque itération, les heuristiques de destruction et de reconstruction sont choisies selon le principe de la roulette et le *score* des heuristiques est modifié en fonction de la qualité de la solution obtenue. Certaines fonctionnalités nouvelles sont également intégrées à la méthode :

Fonction objectif modifiée : la fonction objectif originale est modifiée de façon à prendre en compte non pas seulement la dernière tâche pour chaque niveau de priorité, mais les ε dernières tâches. Cette modification permet d'obtenir une plus grande diversification des valeurs possibles de la fonction objectif ;

Permutation des priorités : l'algorithme considère trois permutations diffentes des priorités (par exemple, affecter les tâches de niveau 1, 2 et 3 dans cet ordre ; ensuite de niveau 3, 2 et 1, etc.). En commençant avec les tâches dont le niveau de priorité apparaît en premier dans la permutation, l'algorithme construit une solution sur laquelle la méthode adaptative est ensuite appliquée. Après un certain nombre d'itérations, les tâches sont fixées dans la solution, et les tâches avec le niveau de priorité suivant sont considérées. La solution obtenue à la fin peut encore être réoptimisée en libérant les tâches considérées en premier.

Kovacs et al. [41] ont aussi utilisé une recherche adaptative à grand voisinage pour résoudre deux versions d'un problème réel de tournées de techniciens. Il s'agit de planifier les routes formant la journée de travail d'un ensemble de techniciens ayant différents niveaux de compétences. Les tâches demandent aussi un certain niveau d'une ou de plusieurs compétences et doivent être réalisées à l'intérieur de fenêtres de temps.
Une première version de l'algorithme est développée pour le cas où chaque tournée est réalisée par un seul technicien, alors que la deuxième version planifie des tournées d'équipes de travail. Les auteurs proposent une heuristique de type ALNS dans les deux cas en faisant appel à différentes méthodes de destruction et de reconstruction de solutions.
Contrairement à Ropke et Pisinger dans [60], un *score* est associé à une paire d'opérateurs (destruction, reconstruction) qui est mis à jour à chaque itération selon la qualité de la solution obtenue.
Dans la première version, les auteurs utilisent les méthodes de destruction suivantes :

— destruction aléatoire ;
— destruction des pires tâches : on enlève ici les tâches qui ont l'impact le moins favorable sur la valeur de l'objectif ;
— destruction avec dépendances : on enlève d'abord une tâche choisie de façon aléatoire et on enlève ensuite celles qui présentent des similarités (du point de vue distance, fenêtres de temps et compétences requises) ;
— destruction par clusters : une route est d'abord choisie aléatoirement, puis on construit un arbre couvrant minimum pour le sous-ensemble des nœuds du graphe faisant partie de cette route. On supprime ensuite l'arc le plus long de façon à obtenir deux clusters. L'un des deux clusters est alors choisi aléatoirement et tous ses nœuds sont retirés de la solution.

Pour les méthodes de reconstruction, on a :

— heuristique d'insertion avec ordre fixé : on considère les tâches une à une dans un ordre donné et on insère chaque tâche à l'endroit qui entraîne l'augmentation la plus faible de la valeur de l'objectif ;
— heuristique d'insertion avec ordre variable : dans cette approche, l'ordre d'insertion des tâches est variable ; plus précisément, on identifie à chaque itération le meilleur endroit pour insérer chaque tâche ; la prochaine tâche choisie est alors celle qui optimise une mesure qui dépend du meilleur endroit où on peut l'insérer.
— heuristique de regret : on calcule la somme des différences au niveau de l'objectif entre l'insertion d'une tâche à sa meilleure position et aux q meilleures positions suivantes. On choisit ensuite la tâche pour laquelle cette somme est la plus grande.

Pour adapter les méthodes de destruction à la deuxième version, une étape est ajoutée à la phase de destruction qui supprime des équipes précédemment formées ainsi que les techniciens dits "redondants" (i.e., qui ne sont pas vraiment impliqués dans la solution). La méthode de destruction aléatoire d'une équipe, telle que décrite dans [11], est également utilisée.

À l'étape de reconstruction, les tâches sont réinsérées de la même façon que dans la première version, mais sans vérifier les compétences des équipes. On procède ensuite à cette vérification. Des techniciens toujours disponibles ayant les compétences requises peuvent alors être ajoutées aux équipes. Enfin, s'il y a toujours des techniciens non affectés, de nouvelles équipes peuvent être formées.

(e) Matheuristique

Pillac et al. [51] présentent une matheuristique pour un TRSP où les pièces de rechange requises pour réaliser une tâche sont prises en compte. L'approche proposée comprend une

heuristique constructive, une recherche parallèle adaptative à grand voisinage (pALNS) et un programme mathématique qui est utilisé à des fins de post-optimisation.

1.4.2. TRSPs dynamiques

Comme leur nom l'indique, les TRSPs dynamiques possèdent un ou plusieurs aspects dynamiques tels : une partie ou toutes les tâches à réaliser se présentent de façon continue au cours de la journée, le temps de déplacement d'un client à un autre varie dynamiquement, etc. Cette spécificité rend ces problèmes beaucoup plus compliqués que les TRSPs statiques.

Pillac et al. [**50**] présentent une méthode pour résoudre un TRSP où une partie des tâches est de nature dynamique. Leur méthode de résolution a été implantée sur une architecture parallèle et peut se résumer ainsi :

1. Un ensemble de solutions initiales avec les tâches connues à l'avance sont créées en utilisant une heuristique de regret [**54**] ;
2. Chaque solution est affectée à un processus esclave qui exécute un algorithme de type ALNS afin de l'améliorer ;
3. Les meilleures solutions sont ensuite communiquées au processus maître, qui met à jour l'ensemble des solutions élites. Cette procédure est répétée pour un certain nombre d'itérations ;
4. La meilleure solution obtenue est retournée ;
5. À l'arrivée d'une nouvelle tâche, la portion des tâches déjà exécutées est fixée et le processus maître relance alors les processus esclaves (Étape 2) afin d'intégrer cette nouvelle tâche.

Bostel et al. [**7**] proposent deux méthodes de résolution pour un problème de tournées de techniciens dynamique. Il s'agit d'un problème rencontré par une compagnie qui gère un réseau d'aqueduc et où des tournées de techniciens doivent être planifiées sur une période d'une semaine pour des réparations ou de la maintenance. Les tâches à ordonnancer sont de deux types, soit celles qui sont connues d'avance (maintenance préventive) et celles qui apparaissent de façon dynamique. Chaque tâche a une fenêtre de temps qui doit être respectée et la compagnie possède un nombre de techniciens et de véhicules limité.

La première méthode proposée est un algorithme mémétique [**47**]. Au début, une population initiale de solutions est créée. Puis, à chaque itération, des solutions enfants sont générées à partir des solutions parents afin de produire une nouvelle population. Essentiellement, deux solutions parents échangent des routes afin de créer deux solutions enfants, qui sont ensuite améliorées avec une méthode de recherche locale. L'algorithme mémétique est d'abord appliqué avec les tâches statiques seulement afin de produire des tournées planifiées pour chacun des jours de la semaine. Les tâches dynamiques sont ensuite intégrées dans

la solution à mesure qu'on avance dans le temps, toujours en faisant appel à l'algorithme mémétique.

La deuxième approche de résolution est basée sur une méthode de génération de colonnes qui n'est applicable qu'à des instances de petite taille.

1.4.3. TRSPs avec temps de service stochastiques

En général, il est difficile d'estimer le temps de service requis par une tâche. Il est donc indiqué de prendre en compte cette incertitude. Ordonez et al. [12], par exemple, modélisent le temps de service comme une variable aléatoire et proposent un modèle d'optimisation robuste (où l'on considère le pire cas pour le temps total requis pour servir les tâches). Le problème est résolu avec une méthode de génération de colonnes où le problème maître correspond à un problème de partitionnement. Le sous-problème est abordé à l'aide de la programmation par contraintes.

Les auteurs dans [6] décrivent un problème réel rencontré par British Telecom qui gère un grand nombre de techniciens avec différentes compétences pour servir des clients qui sont éparpillés géographiquement. Le problème présente des aspects dynamiques et stochastiques. Ainsi, une fraction des tâches est révélée au cours de la journée, tandis que les temps de parcours et les temps de service sont stochastiques. Afin de résoudre le problème, les auteurs font d'abord appel à une méthode de clustering (K-means) pour regrouper les tâches qui sont proches d'un point de vue géographique. Chaque groupe définit alors une zone. Ensuite, une heuristique d'insertion est utilisée pour affecter les techniciens aux zones en commençant avec les techniciens qui n'ont que peu de compétences. L'ordre d'insertion des tâches est défini à l'aide d'un système de règles qui favorise les tâches de courte durée avec une date de fin serrée.

Chapitre 2

PROGRAMMATION LINÉAIRE EN NOMBRES ENTIERS POUR UN PROBLÈME MULTI-ATTRIBUT DE TOURNÉES DE TECHNICIENS

Ce chapitre correspond à un article qui a été publié dans la revue *Information Systems and Operational Research* 56(1), 33-49, 2018. Les co-auteurs sont I. Mathlouthi, M. Gendreau et J.-Y. Potvin.

Dans ce chapitre nous considérons un problème multi-attributs de tournées de techniciens motivé par une application touchant à des équipements de transactions électroniques. Le but est de créer des routes pour les techniciens afin réaliser des tâches chez différents clients, tout en maximisant le gain total associé aux clients desservis moins les coûts d'exploitation (distance totale parcourue, heures de travail supplémentaires). De plus, des contraintes doivent êtres prises en considération comme les compétences des techniciens pour les différentes tâches, leur horaire de travail, une distance maximale autorisée, des fenêtres de temps multiples pour le service et un inventaire de pièces à respecter. Un modèle de programmation linéaire mixte avec variables continues et entières est proposé pour formuler ce problème. Le modèle est ensuite résolu avec le solveur commercial CPLEX. Les résultats des expériences démontrent la difficulté du problème en fonction de ses différentes caractéristiques et soulignent son inhérente complexité.

Mixed Integer Linear Programming for a Multi-Attribute Technician Routing and Scheduling Problem

In this paper, we consider a multi-attribute technician routing and scheduling problem motivated by an application for the maintenance and repair of electronic transaction equipment. This problem is aimed at routing technicians to perform tasks at different customer locations so as to maximize the total gain associated with served customers, minus the operations costs (total traveled distance and overtime of the technicians). At the same time, a number of constraints must be satisfied like technician skills, breaks, maximum distance, multiple time windows and parts inventory. A mixed integer linear programming model is proposed to address this problem, which is then solved with a commercial solver. The computational results explore the difficulty of the problem along various dimensions and underline its inherent complexity.

Keywords : Technician routing and scheduling problem, multiple time windows, inventory, mixed integer linear programming.

2.1. Introduction

In the technician routing and scheduling problem (TRSP), a number of technicians must serve different tasks at minimum cost while satisfying resource constraints. The TRSP belongs to the class of Vehicle Routing Problems (VRP) [32] and is related in particular to the VRP with time windows [38]. There are, however, significant differences such as skill requirements to perform different types of tasks and relatively large service times when compared to travel times. In our case, we also deal with inventory issues, in particular the availability of spare parts and special parts in the technician's vehicle.

This TRSP, which was proposed to us by a company involved in the maintenance and repair of electronic transactions equipment, is defined as follows. There is a set of technicians, each with different skills for different types of tasks. These tasks may also have different priority levels. There is a set of customers, where every customer requires the service of a technician to perform a particular task. The goal is then to assign tasks to technicians and to build a route for each technician, starting and ending at his home base location, so as to optimize an objective that involves the maximization of the total gain obtained by serving tasks minus the operations costs (total traveled distance and overtime of the technicians). The solution must also satisfy various constraints related to the required skills of the technicians to perform the assigned tasks, breaks, maximum distance, multiple time windows, availability of spare parts and special parts. It should be noted that technician skills are sometimes accounted for in the literature as degrees of ability (instead of constraints) and are integrated into the objective.

A particular feature of our problem comes from an inventory of spare parts. More precisely, a technician leaves his home base position to start his route with an initial inventory. However, if there is not enough parts to serve all tasks, he has the opportunity to replenish once along the route by going through a particular depot, which has previously been assigned to him. There are also special parts that are not carried in the vehicle unless one or more tasks along the route require them. In such a case, the technician must also take them from his depot before performing these tasks.

Overall, the main contributions of this work come from the development of a mixed integer linear programming (MILP) model for a complex problem motivated by a real-world application. Furthermore, an analysis of the problem difficulty along various dimensions is provided by generating instances with different characteristics and by solving them with CPLEX.

The remainder of this paper is organized as follows. In Section 2, we review some related work. In Section 3, the problem is introduced. Then, the mathematical formulation is presented in Section 4. The generation of the test instances is explained in Section 5, followed by the results obtained in Section 6. Finally, Section 7 concludes the paper.

2.2. Related Work

TRSPs have received limited attention compared to VRPs, despite their numerous practical applications. The first work is reported in 1997 by Tsang and Voudouris [**67**] where the authors introduce the technician workforce scheduling problem faced by British Telecom. The particularity of this problem is that there are no skill constraints. They are replaced by a proficiency factor that reduces the service time depending on the technician experience. A Guided Local Search (GLS) and a so-called Fast Local Search are used to solve this problem. Later, Weigel and Coo [**71**] introduce the problem faced by a well-known retailer when providing on-site technical assistance. The proposed solution consists of assigning requests to technicians and then optimizing each route individually through Or-opt exchanges [**48**].

In [**73**], Xu and Chiu propose a MILP for a TRSP where the objective is to maximize the number of served requests, while taking into account request priorities, skills and overtime. Four heuristics based on local search and GRASP are reported. In [**5**], Blakeley et al. solve a periodic maintenance problem faced by the Schindler Elevator Corporation for their elevators and escalators. In this application, the technician routes must account for technician skills, travel times and working regulations. A little bit later, a similar application was addressed by Tang et al. [**64**] with a tabu search heuristic.

In 2007, the French Operations Research Society (ROADEF) initiated a challenge based on a problem encountered by France Telecom. The participants had to schedule technician tours on a multiple-day horizon. The particularity of this problem is that each task needs one or more skills with different proficiency levels, while technicians can have multiple skills.

To solve this problem, teams of technicians working together must be created. However, the routing aspect of the problem is ignored. Based on this challenge, Hashimioto et al. [33], developed a GRASP for this problem while Cordeau et al. [11] proposed a mathematical model and a problem-solving methodology based on an adaptive large neighborhood search (ALNS).

A dynamic variant of the TRSP is addressed in Bostel et al. [7]. The authors introduce a problem faced by Veolia, a water treatment and distribution company. In this problem, technician routes must be planned over a period of one week for repair or maintenance. The tasks to be scheduled can either be known in advance (preventive maintenance) or can occur dynamically. Each task has a time window for service. The first proposed method is a memetic algorithm, which is first applied on static tasks to produce tours for every day of the week. Dynamic tasks are then integrated into the solution as they occur, still using the memetic algorithm. The second approach is based on a column generation algorithm which can only be applied to problem instances of small size.

Finally, Pillac et al. [50] also address a TRSP in which a fraction of the tasks occur dynamically. A parallel architecture is proposed to speed up the calculations. An initial solution is first created with known tasks using a regret heuristic [54]. This solution is then improved with ALNS [53]. The latter works by successively destroying (removing tasks) and repairing (reinserting tasks) to produce a new solution from the current one. When a new task is received, the part of the current solution already executed is fixed and the new task is incorporated into the solution by running the ALNS for a limited number of iterations.

To the best of our knowledge, no work considers concurrently technician skills, task priorities, multiple time windows, breaks, overtime and parts inventory. This complex problem will now be introduced more precisely in the next section.

2.3. Problem Statement

Our problem is a technician routing and scheduling problem encountered in maintenance and repair services for electronic transactions equipment. Although dynamic customer requests can sometimes be accommodated in the real problem (same-day service), it is assumed here that all tasks to be performed at customer locations are known beforehand. Also, there is no need to serve all of them, that is, tasks can be postponed. Each task is characterized by the following attributes :

— Gain (based on the customer's service priority) ;
— Subset of technicians with the required skills to perform the task ;
— Types and number of required spare parts of each type ;
— Special part, if any ;
— Service time ;
— Multiple time windows.

There are one or more depots, where each depot contains a (virtually infinite) number of parts. Each technician is assigned to a particular depot for the replenishment of his spare parts or for the acquisition of special parts. Each technician can work from 9H00 AM to 5H00 PM (if no overtime) and is allowed three breaks during the day : one break of 15 minutes in the morning and afternoon, respectively, and a mid-day break of 30 minutes. The morning, mid-day and afternoon periods are defined through time window constraints. Finally, a technician cannot travel more than a given maximum distance during his workday.

With this information, the problem is to design technician routes for one day, where each route starts and ends at the technician's home base and serves a number of tasks, including one possible stop at the preassigned depot, while satisfying the required skills for each task, the multiple time windows at each location, the time window for each break, the required number of spare parts for each task and the requirement (or not) of a special part for each task. The goal is to optimize an objective involving overtime, total traveled distance and total gain over the performed tasks.

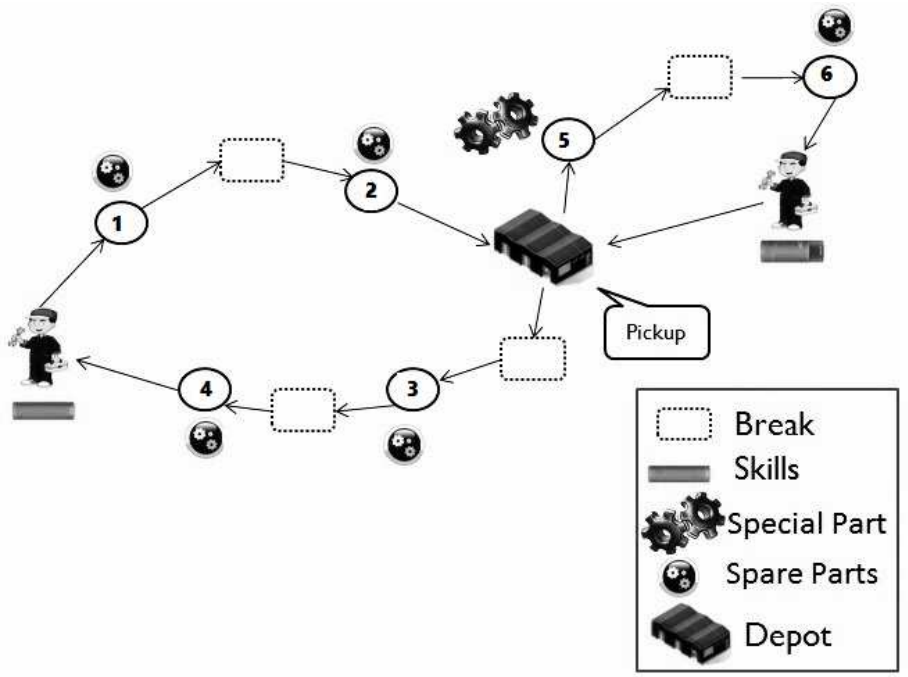

FIGURE 2.1. Example with two technician routes

Figure 2.1 shows an example with one depot, a single customer requiring a special part, and two technician routes. The first technician has all the required skills to perform all tasks. His route serves tasks 1, 2, 3, 4 in this order. After task 2, he goes to the depot to replenish his inventory of spare parts, takes a break and then goes to task 3. Then, he takes another break before going to task 4. The second technician does not have all the required skills to perform all tasks. His route serves tasks 5 and 6 and he must first go to the depot to pick up the special part needed by task 5. He then takes a break before going to task 6.

2.4. Mathematical Model

In this section a mixed integer linear programming model for our problem is provided. The parameters are first defined, followed by the decision variables. It should be noted that we are not aware of any model that accounts for all these characteristics.

2.4.1. Parameters

— $D = \{1, \ldots, g\}$: Set of depots ;
— $K = \{1, \ldots, m\}$: Set of technicians ;
— $K^d \subseteq K$: Set of technicians assigned to depot $d \in D$;
— $I = \{1, \ldots, n\}$: Set of tasks ;
— $\Theta = \{\theta^1, \ldots, \theta^m\}$: Set of starting positions of the technicians (home bases) ;
— $I' = I \cup \Theta$
— $I'' = I' \cup D$
— $P = \{1, \ldots, o\}$: Set of types of spare parts ;
— $I_t \subseteq I$: Set of tasks that require a special part ;
— $P_i \subseteq P$: Set of types of spare parts required to perform task $i \in I$;
— F_i : Set of time windows of task $i \in I$;
— α_i : Gain of task $i \in I$;
— a_i^r : Lower bound of time window $r \in F_i$ of task $i \in I$;
— b_i^r : Upper bound of time window $r \in F_i$ of task $i \in I$;
— a_l : Lower bound of time window of break $l = 1, 2, 3$;
— b_l : Upper bound of time window of break $l = 1, 2, 3$;
— a^k : Earliest workday start time of technician $k \in K$;
— b^k : Latest (regular) workday end time of technician $k \in K$;
— σ_i : Duration of service of task $i \in I$;
— σ_l : Duration of break $l = 1, 2, 3$;
— d_{ij} : Distance between i et $j \in I''$, $i \neq j$;
— t_{ij} : Travel time between i et $j \in I''$, $i \neq j$;
— $d_{\max}$: Maximum traveled distance of each technician ;
— d^k : Depot of technician $k \in K$;
— z_{pi} : Number of spare parts of type $p \in P$ needed to perform task $i \in I$;
— v_p^k : Number of new spare parts of type $p \in P$ in the vehicle of technician $k \in K$ after replenishment ;
— $v_{p\theta^k}^k$: Number of spare parts of type $p \in P$ in the vehicle of technician $k \in K$ at his starting position (home base) ;
— $s_i^k = \begin{cases} 1, & \text{if technician } k \in K \text{ has the required skills to perform task } i \in I \text{ ;} \\ 0, & \text{otherwise.} \end{cases}$

— M : Arbitrarily large constant (although a single symbol is used here, different arbitrary large constants can be associated with different constraints) ;
— β : Replenishment time at the depot for each technician ;
— $\psi, \vartheta, \varphi, \upsilon \in [0,1]$.

2.4.2. Variables

— δ^k : Overtime of technician $k \in K$;
— U_{pi}^k : Number of spare parts of type $p \in P$ in the vehicle of technician $k \in K$ after performing task $i \in I$;
— τ_i : Start time of task $i \in I$;
— τ_l^k : Start time of break $l = 1, 2, 3$ of technician $k \in K$;
— $x_{ij}^k = \begin{cases} 1, & \text{if technician } k \text{ has not visited his depot before moving directly from } i \text{ to } j \text{ ;} \\ 0, & \text{otherwise.} \end{cases}$
— $\bar{x}_{ij}^k = \begin{cases} 1, & \text{if technician } k \text{ has visited his depot before moving directly from } i \text{ to } j \text{ ;} \\ 0, & \text{otherwise.} \end{cases}$
— $\tilde{x}_{ij}^k = \begin{cases} 1, & \text{if technician } k \text{ moves from } i \text{ to } j \text{ through his depot ;} \\ 0, & \text{otherwise.} \end{cases}$
— $y_i^k = \begin{cases} 1, & \text{if task } i \text{ is assigned to technician } k \text{ ;} \\ 0, & \text{otherwise.} \end{cases}$
— $w_{il}^k = \begin{cases} 1, & \text{if technician } k \text{ takes his break } l = 1, 2, 3 \text{ after task } i \text{ ;} \\ 0, & \text{otherwise.} \end{cases}$
— $\bar{w}_l^k = \begin{cases} 1, & \text{if technician } k \text{ is active and does not take his break } l = 1, 2, 3 \text{ ;} \\ 0, & \text{otherwise.} \end{cases}$
— $f_i^r = \begin{cases} 1, & \text{if task } i \text{ is performed in time window } r,\ r \in F_i \text{ ;} \\ 0, & \text{otherwise.} \end{cases}$
— $\bar{u}^k = \begin{cases} 1, & \text{if technician } k \text{ is inactive (not used)} \\ 0, & \text{otherwise.} \end{cases}$

2.4.3. Model

$$\max \varphi \sum_{k \in K} \sum_{i \in I} \alpha_i y_i^k - \vartheta \sum_{k \in K} \sum_{i,j \in I\prime} (x_{ij}^k + \bar{x}_{ij}^k) d_{ij} + \tilde{x}_{ij}^k (d_{id^k} + d_{d^k j}) - \psi \sum_{k \in K} \delta^k - \upsilon \sum_{k \in K} \sum_{l=1,2,3} \bar{w}_l^k \quad (2.4.1)$$

Subject to

Task/Technician constraints

$$\sum_{k\in K} y_i^k \leq 1 \quad i \in I \tag{2.4.2}$$

$$y_i^k \leq s_i^k \qquad \begin{matrix} i \in I \\ k \in K \end{matrix} \tag{2.4.3}$$

Coherence between variables of type x and u

$$\bar{u}^k = 1 - \sum_{i\in I}(x_{\theta^k i}^k + \tilde{x}_{\theta^k i}^k) \qquad k \in K \tag{2.4.4}$$

Coherence between variables of type x and y

$$\sum_{j\in I}(x_{ji}^k + \tilde{x}_{ji}^k + \bar{x}_{ji}^k) + x_{\theta^k i}^k + \tilde{x}_{\theta^k i}^k = y_i^k \qquad \begin{matrix} i \in I \\ k \in K \end{matrix} \tag{2.4.5}$$

$$\bar{x}_{i\theta^k}^k + \sum_{j\in I} \bar{x}_{ij}^k = \tilde{x}_{\theta^k i}^k + \sum_{j\in I}(\bar{x}_{ji}^k + \tilde{x}_{ji}^k) \qquad \begin{matrix} i \in I \\ k \in K \end{matrix} \tag{2.4.6}$$

Flow constraints

$$\sum_{i\in I}(x_{\theta^k i}^k + \tilde{x}_{\theta^k i}^k) \leq 1 \qquad k \in K \tag{2.4.7}$$

$$\sum_{i\in I}(x_{\theta^k i}^k + \tilde{x}_{\theta^k i}^k) - \sum_{i\in I}(x_{i\theta^k}^k + \bar{x}_{i\theta^k}^k) = 0 \qquad k \in K \tag{2.4.8}$$

$$[x_{\theta^k i}^k + \tilde{x}_{\theta^k i}^k + \sum_{j\in I}(x_{ji}^k + \tilde{x}_{ji}^k + \bar{x}_{ji}^k)] - [x_{i\theta^k}^k + \bar{x}_{i\theta^k}^k + \sum_{j\in I}(x_{ij}^k + \tilde{x}_{ij}^k + \bar{x}_{ij}^k)] = 0 \tag{2.4.9}$$

$$\begin{matrix} i \in I \\ k \in K \end{matrix}$$

Special parts

$$y_i^k \leq \tilde{x}_{\theta^k i}^k + \sum_{j\in I}(\tilde{x}_{ji}^k + \bar{x}_{ji}^k) \qquad \begin{matrix} i \in I_t \\ k \in K \end{matrix} \tag{2.4.10}$$

Inventory constraints

$$U_{pj}^k + (1 - (x_{ji}^k + \tilde{x}_{ji}^k + \bar{x}_{ji}^k))M + v_p^k \tilde{x}_{ji}^k \geq z_{pi} \qquad \begin{matrix} (i \neq j) \in I \\ p \in P_i \\ k \in K \end{matrix} \tag{2.4.11}$$

$$v_{p\theta^k}^k + (1 - (x_{\theta^k i}^k + \tilde{x}_{\theta^k i}^k))M + v_p^k \tilde{x}_{\theta^k i}^k \geq z_{pi} \qquad \begin{matrix} i \in I \\ p \in P_i \\ k \in K \end{matrix} \tag{2.4.12}$$

$$U_{pi}^k \leq U_{pj}^k - z_{pi} + (1 - (x_{ji}^k + \bar{x}_{ji}^k + \tilde{x}_{ji}^k))M + v_p^k \tilde{x}_{ji}^k \qquad \begin{matrix}(i \neq j) \in I \\ p \in P_i \\ k \in K\end{matrix} \tag{2.4.13}$$

$$U_{pi}^k \leq v_{p\theta^k}^k - z_{pi} + (1 - (x_{\theta^k i}^k + \tilde{x}_{\theta^k i}^k))M + v_p^k \tilde{x}_{\theta^k i}^k \qquad \begin{matrix}i \in I \\ p \in P_i \\ k \in K\end{matrix} \tag{2.4.14}$$

Maximal distance constraint

$$\sum_{i,j \in I} ((x_{ij}^k + \bar{x}_{ij}^k)d_{ij} + \tilde{x}_{ij}^k(d_{id^k} + d_{d^k j})) + \sum_{i \in I} d_{\theta^k i} x_{\theta^k i}^k + \sum_{i \in I} (d_{\theta^k d^k} + d_{d^k i})\tilde{x}_{\theta^k i}^k \tag{2.4.15}$$
$$+ \sum_{i \in I} d_{i\theta^k}(x_{i\theta^k}^k + \bar{x}_{i\theta^k}^k) \leq d_{\max}$$
$$k \in K$$

Time constraints

$$\delta^k \geq (\tau_i + \sigma_i + t_{i\theta^k} - b^k) - (1 - (x_{i\theta^k}^k + \bar{x}_{i\theta^k}^k))M \qquad \begin{matrix}i \in I \\ k \in K\end{matrix} \tag{2.4.16}$$

$$\tau_j + \sigma_j + (x_{ji}^k + \bar{x}_{ji}^k)t_{ji} + \tilde{x}_{ji}^k(t_{jd^k} + t_{d^k i} + \beta) + \sum_{l=1,2,3} \sigma_l w_{jl}^k \tag{2.4.17}$$
$$\leq \tau_i + (1 - (x_{ji}^k + \bar{x}_{ji}^k + \tilde{x}_{ji}^k))M$$
$$\begin{matrix}(i \neq j) \in I \\ k \in K\end{matrix}$$

$$a^k + x_{\theta^k i}^k t_{\theta^k i} + \tilde{x}_{\theta^k i}^k(t_{\theta^k d^k} + t_{d^k i} + \beta) \leq \tau_i + (1 - (x_{\theta^k i}^k + \tilde{x}_{\theta^k i}^k))M \qquad \begin{matrix}i \in I \\ k \in K\end{matrix} \tag{2.4.18}$$

$$\sum_{r \in F_i} f_i^r a_i^r \leq \tau_i \leq \sum_{r \in F_i} f_i^r b_i^r \qquad i \in I \tag{2.4.19}$$

$$\sum_{r \in F_i} f_i^r = \sum_{k \in K} y_i^k \qquad i \in I \tag{2.4.20}$$

$$\sum_{i \in I} w_{il}^k + \bar{w}_l^k + \bar{u}^k = 1 \qquad \begin{matrix}k \in K \\ l = 1,2,3\end{matrix} \tag{2.4.21}$$

$$w_{il}^k \leq y_i^k \qquad \begin{matrix}i \in I \\ k \in K \\ l = 1,2,3\end{matrix} \tag{2.4.22}$$

$$\tau_i + \sigma_i \leq \tau_l^k + (1 - w_{il}^k)M \qquad \begin{matrix}i \in I \\ k \in K \\ l = 1,2,3\end{matrix} \tag{2.4.23}$$

$$a_l(1-\bar{w}_l^k-\bar{u}_k) \leq \tau_l^k \leq b_l(1-\bar{w}_l^k-\bar{u}_k) \qquad \begin{matrix} k \in K \\ l = 1,2,3 \end{matrix} \tag{2.4.24}$$

Domain restrictions

$$\tau_i, \tau_l^k, \delta^k \geq 0 \tag{2.4.25}$$

$$x_{ij}^k, \bar{x}_{ij}^k, \tilde{x}_{ij}^k, y_i^k, f_j^r, w_i^l, \bar{w}_l^k, \bar{u}^k \in \{0,1\} \tag{2.4.26}$$

This model contains three different types of x variables to account for the three possible cases along a technician route : technician k travels from i to j and has not previously visited his depot (x_{ij}^k) ; technician k travels from i to j and has previously visited his depot ($\bar{x}_{ij}^k$) ; technician k visits his depot while traveling from i to j ($\tilde{x}_{ij}^k$). Clearly, coherence among these three types of variables must be maintained along each route.

The main components of our model can now be described.

— The objective function (2.4.1) maximizes the total gain minus the total distance and total overtime. The fourth component is used to force the technicians to take their breaks. Each component is weighted by a different parameter.

— Constraints (2.4.2) and (2.4.3) ensure that each task is visited by at most one technician with the required skills.

— Constraint (2.4.4) establishes a relationship between variables x, $\tilde{x}$ and $\bar{u}^k$. It states that a technician is inactive if he does not depart from his home base.

— Constraints (2.4.5) and (2.4.6) ensure the coherence between the values of variables x, $\tilde{x}$, $\bar{x}$ and y. Constraint (2.4.5) states that task i must be assigned to technician k if he visits this task either from his home base or from another task. Constraint (2.4.6) establishes a relationship among the three types of x variables. That is, if a technician travels from task i to return to his home base or to visit another task and if he has already visited the depot, then this visit to the depot must have taken place just before reaching i or anywhere else along the route before reaching i.

— Constraints (2.4.7) to (2.4.9) are the flow conservation constraints. Constraints (2.4.7) and (2.4.8) state that a technician can depart from the depot at most once and if he departs then he must return to the depot. Constraint (2.4.9) corresponds to the flow conservation constraint of each task.

— Constraint (2.4.10) states that a technician must visit his depot before serving a task that requires a special part.

— Constraints (2.4.11) and (2.4.12) ensure that a technician has the required spare parts to perform a task. Constraint (2.4.11) states the following : assuming that task j is visited just before task i, the number of available spare parts after the service of j, plus any additional parts obtained by visiting the depot while traveling from j to i

should cover the requirements of task i. Constraint (2.4.12) covers the case when task i is visited directly from the depot.

— Constraints (2.4.13) and (2.4.14) are aimed at updating the inventory after performing a task. Again, constraint (2.4.14) covers the case when task i is visited directly from the depot.

— Constraint (2.4.15) forces the maximum travel distance of each technician to be satisfied. This constraint takes into account the three different types of x variables introduced above while considering, at the same time, if i is the first, the last or an intermediary task along the route.

— Constraint (2.4.16) defines the overtime of each technician. If i is the last task in a technician route, then the return time at the depot is derived from the service start time of task i (i.e., service start time plus service time plus travel time to the home base). Then, the return time at the depot minus the end time of a regular workday provides the overtime (if any).

— Constraints (2.4.17) and (2.4.18) ensure the time continuity of each route. Basically, if task i is visited after j in the route of technician k, then the service start time at j plus the service time plus any break time of technician k plus the time to travel to i (either directly or through the depot) should not exceed the service start time of task i. These constraints also eliminate subtours.

— Constraints (2.4.19) and (2.4.20) are related to the time windows. Constraint (2.4.19) forces the service start time of each task to take place within one of the multiple time windows. Constraint (2.4.20) establishes a relationship between variables of type f and y. Basically, a task can be visited within one of its time windows if and only if it is assigned to a technician.

— Constraints (2.4.21) to (2.4.24) force each technician to take his breaks at an appropriate time. Constraint (2.4.21) states that a technician is either active and takes a break, active and does not take a break or inactive (for each one of the morning, mid-day and afternoon breaks). Constraint (2.4.22) establishes a relationship between variables of type w and y. Basically, a technician can only take a break after some task i if he has been assigned to this task. Constraint (2.4.23) ensures the time continuity of a technician route. If a break is taken by a technician after task i, then the service start time at i plus the service time should not exceed the start time of the break. Finally, constraint (2.4.24) forces the start time of the break to take place between the appropriate time bounds.

— Constraints (2.4.25) and (2.4.26) define the domain of the variables.

2.5. Test instances

An instance generator was developed for testing purposes. In the following, we explain how the various characteristics of each instance were generated.

1. *Service area*. The service area corresponds to a 40 km × 40 km or 50 km × 50 km squared area.
2. *Depot Location*. There are 3 depots randomly located within the service area.
3. *Task location*. Each task is randomly located within the service area.
4. *Task service time*. The service time of each task is randomly chosen between 30 and 45 minutes.
5. *Task gain*. The gain associated with a task in randomly chosen between 1 and 10.
6. *Technician home base*. To get a good coverage of the tasks to be performed, the first two technicians are located at the opposite ends of the service area (along the diagonal). The other technicians are randomly located within the service area.
7. *Technician skills*. With each technician is associated the percentage of tasks that he can perform, which is chosen from 100%, 50% and 25%.
8. *Parts*. The number of spare parts needed to perform a task is randomly chosen between 0 and 3. Then, each spare part is assigned a type, among 4 different types. Also, a special part is needed with a probability of 0.125.
9. *Time windows*. Both narrow and wide time windows are considered. The latter are twice as wide as the former on average. The length of a narrow time window is randomly generated between 60 and 90 minutes. The lower bound of the first time window is chosen randomly between 9H00 AM and noon. The lower bounds of the remaining time windows are set between 2 and 3 hours after the upper bound of the previous time window until a maximum of 3 time windows are obtained.

It should be noted that many of the above values come from the real-world application. However, no information was available to us about the location of tasks, depots and home bases. Figure 2.2 shows an example with 20 tasks, 2 depots and 3 technicians within a 40 km × 40 km service area. The crow fly distance is assumed between each pair of locations and the speed of the vehicles is set at 50 km/h.

A total of $2 \times 2 \times 4 = 16$ subsets of instances, with 5 instances in each subset, are generated by considering every possible combination of the parameter values shown in Table 2.1. After reporting the results obtained with CPLEX on these instances, we evaluate the individual impact of parameters Skills, Service time, Special parts and # Technicians on the performance of the solver, by considering different values for each one of them, while keeping

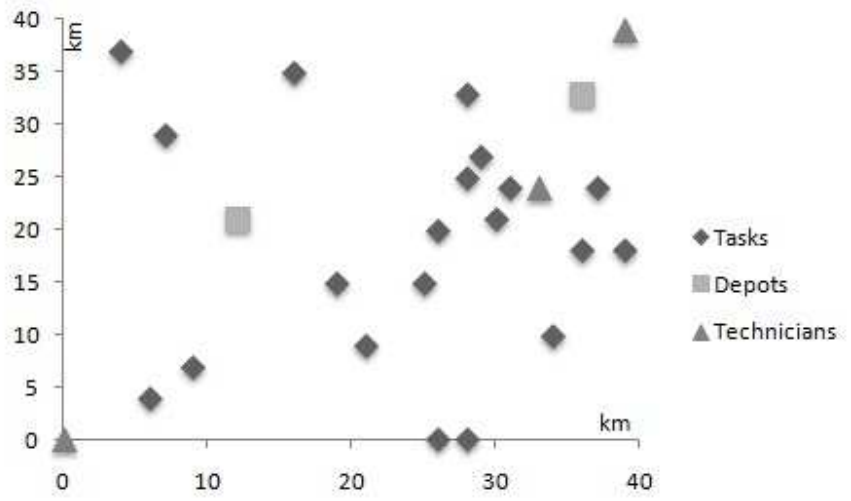

FIGURE 2.2. An example with 20 tasks, 3 technicians and 2 depots

the other parameter values fixed to those shown in Table 2.1. Table 2.2 presents the new value(s) considered for each parameter.

Time windows	Narrow, Wide
Service Area	40kms × 40kms, 50kms × 50kms
# Tasks	10, 15, 20, 25
# Technicians	3
Skills (% Tasks)	100% (1 technician), 50% (1 technician), 25% (1 technician)
Service time	30-45 minutes
Special part (prob.)	0.125

TABLE 2.1. Basic parameter values

Skills 1	50% (2 technicians), 25% (1 technician)
Skills 2	100% (3 technicians)
Service time 1	15-30 minutes
Service time 2	10-20 minutes
Special part 1 (prob.)	0
Special part 2 (prob.)	0.25
# Technicians	4

TABLE 2.2. New parameter values

With regard to the skill configuration 50% (2 technicians) and 25% (1 technician), it should be noted that the second technician with 50% of all tasks automatically takes the tasks that cannot be performed by the first technician. In this way, every task is covered. Also, when the number of technicians is 4, the fourth technician can perform 25% of all tasks. That is, the basic skill configuration shown in Table 1 becomes 100% (1 technician), 50% (1 technician) and 25% (2 technicians).

Finally, the maximum distance traveled by each technician during his workday was set to 125 km, while the weight parameters in the objective function were set to $\psi = 5$ (overtime), $\vartheta = 5$ (total distance), $\varphi = 550$ (total gain), $\upsilon = 50$ (breaks) and then normalized between 0 and 1. Thus, more emphasis is given to the total gain over all performed tasks. The value of parameter υ is also relatively high to force the technicians to take their breaks.

2.6. Computational results

This section reports the results obtained with CPLEX 12.6 on the subsets of instances introduced in the previous section. The solver was given a maximum of 24 hours of computation time on a 3.07GHz Intel Xeon X5675 processor. We recall that each subset is made of 5 different instances. The tables of results provide the following information (in this order) :

- *Name*. The name of each subset of instances using the format X-Y-Z, where X is the width of the time windows (either N for narrow or W for wide), Y the size of the service area (either 40 for 40kms × 40kms or 50 for 50kms × 50 kms) and Z the number of tasks (either 10, 15, 20 or 25).
- *# Opt* : number of instances solved to optimality (the number of instances for which an integer solution was obtained, without any proof of optimality, is put between parentheses) ;
- *CPU* : average computation time in hours :minutes :seconds (only for instances solved to optimality) ;
- *Gap* : average gap in percentage (only for instances not solved to optimality)
- *# Tasks* : average number of tasks performed by the technicians ;
- *# Idle* : average number of idle (unassigned) technicians.

Table 2.3 reports the results obtained with the basic parameter values shown in Table 2.1. We observe that CPLEX can solve all instances of size 10 within minutes. However, CPLEX begins to strive when the number of tasks increase to 15. Only 9 instances out of 20 are solved to optimality and the computation times now vary between 7 and 16 hours. Only two instances of size 20 are solved to optimality and none of size 25. It should be noted that two technicians are often enough to cover all tasks on the smallest instances of size 10 (as indicated by the average number of idle technicians). Also, increasing the size of the service area leads to a fewer number of served tasks, which is quite understandable due to (1) the increased travel times between locations and (2) the maximum travel distance constraint of each technician which is now binding in some cases.

Considering now Table 2.4, we observe that the instances are generally easier to solve when the percentage of special parts increases. In particular, the number of instances that are solved to optimality increases from 26 (no special parts) to 31 (25% of special parts). In

fact, the need for special parts constrains the solution space, thus reducing the combinatorial complexity.

Table 2.5 reports the results when the skills are modified. When each technician has all the required skills (which, in fact, eliminates this distinctive characteristic from the problem), the instances become more difficult to solve due to the increased flexibility in the assignment of technicians to tasks. Conversely, when the skills of the technicians are more restricted, a larger number of optimal solutions emerges, even for instances of size 25 (at the expense of a substantial increase in computation times).

The impact of service times is reported in Table 2.6. We observe a slight increase in the number of idle technicians and number of tasks performed by the technicians when the service time is reduced (except for W-50-25). This result makes sense because each technician can now serve more tasks during his workday. Finally, and as expected, Table 2.7 shows that the number of idle technicians and number of tasks performed by the technicians increase when an additional technician is available.

Overall, these results show that the technicians' skills have a significant impact on the difficulty of the problem. The percentage of special parts and the service times also have an impact, although to a lesser extent.

As a last attempt to improve our results, we quickly generated a relatively good initial solution to provide an upper bound to CPLEX (warm start). For this purpose, we developed a greedy insertion heuristic followed by a steepest descent based on two types of modifications : moving a task at another position and exchanging the position of two tasks. The results obtained on the basic configurations with and without the warm start on the largest instances with 20 and 25 tasks are reported in Table 2.8. Although some improvement was obtained in most cases with regard to the gap or computation time (when CPLEX terminates before the time limit of 24 hours), this approach did not allow us to solve instances of larger size. Clearly, the problem considered is very complex and deserves further methodological developments.

2.7. Concluding remarks

We have introduced a technician routing and scheduling problem motivated by an application in the domain of electronic transactions equipment. A mixed integer linear programming formulation is proposed and solved with CPLEX. The results show that the problem is very difficult, given that only instances of small size can be solved to optimality. The modeling effort reported in this work will now pave the way for further methodological developments. Our next step will be to consider a branch-and-price framework where branching applies to variables of the MILP. Eventually, constraints could be introduced to obtain a

branch-and-price-and-cut framework. These developments will help to solve instances of larger size. Matheuristics will also be experimented with, thus sacrificing optimality in the hope of solving even larger instances in reasonable computation times.

Mastering the deterministic variant of the problem, however, is just a part of the story (or an approximation of the real story) because the real-word is stochastic. In our case, parts may be defective, service can take longer than expected, etc. Thus, it would be desirable to integrate sources of uncertainty into the problem-solving approach, which would lead to stochastic programming approaches. Clearly, the development of a solution methodology for a real-world problem is a long process and this paper is just a first milestone along the way. But, once completed, the developed software could be run during the night to produce good and robust technician routes for the next day. This achievement would obviously be of great value for the involved company.

If we take a broader look at TRSPs, it is clear that this field of research has received only limited attention up to now. In the few works reported in the literature, the problem definition is tightly linked to the underlying application. It remains that many extensions with a rather general scope have not yet been thoroughly addressed, or not addressed at all. Without being exhaustive, we can mention synchronization issues among technicians to perform particular tasks, workload balance among technicians, management of the technicians' tools, occurrence of tasks during the day that must be handled in real-time (as observed in corrective maintenance), multiperiod schedules, etc. On the methodological side, there is also plenty of space for new developments, particularly with regard to exact methods. For example, stochastic programming seems to be a totally unexplored avenue among exact methods. Finally, recent metaheuristic and matheuristic developments based on large neighborhood search offer promising avenues for further research.

Name	# Opt	CPU	Gap	# Tasks	# Idle
N-40-10	5	00 :13 :59	0%	9.6	0.8
N-40-15	2(3)	16 :37 :55	5%	14	0.2
N-40-20	0(5)	-	10%	17	0
N-40-25	0(5)	-	12%	20.6	0
N-50-10	5	00 :09 :02	0%	8.4	0.8
N-50-15	3(2)	08 :10 :29	10%	11.8	0.2
N-50-20	1(4)	15 :17 :09	17%	15.6	0.2
N-50-25	0(5)	-	18%	18.6	0
W-40-10	5	00 :12 :25	0%	10	0.6
W-40-15	1(4)	12 :49 :02	3%	14	0.4
W-40-20	0(5)	-	9%	17	0.2
W-40-25	0(5)	-	9%	21	0
W-50-10	5	00 :10 :33	0%	8.4	0.8
W-50-15	3(2)	07 :05 :31	10%	12.2	0.2
W-50-20	1(4)	07 :02 :50	17%	15.6	0.2
W-50-25	0(5)	-	12%	19.33	0

TABLE 2.3. Basic configurations

Name	No special parts (0%)					Basic configurations (12.5%)					More special parts (25%)				
	# Opt	CPU	Gap	# Tasks	# Idle	# Opt	CPU	Gap	# Tasks	# Idle	# Opt	CPU	Gap	# Tasks	# Idle
N-40-10	5	00 :26 :23	0%	9.8	0.8	5	00 :13 :59	0%	9.6	0.8	5	00 :08 :36	0%	9.6	1
N-40-15	1(4)	10 :25 :01	4%	13.8	0.2	2(3)	16 :37 :55	5%	14.0	0.2	1(4)	08 :28 :50	4%	14.0	0.2
N-40-20	0(5)	-	10%	16.4	0.2	0(5)	-	10%	17.0	0	0(5)	-	9%	17.0	0
N-40-25	0(5)	-	12%	20.6	0	0(5)	-	12%	20.6	0	0(5)	-	12%	19.8	0
N-50-10	5	00 :16 :15	0%	8.4	0.8	5	00 :09 :02	0%	8.4	0.8	5	00 :05 :45	0%	8.4	0.8
N-50-15	3(2)	07 :28 :21	11%	12.4	0.2	3(2)	08 :10 :29	10%	11.8	0.2	3(2)	03 :58 :37	8%	11.8	0.2
N-50-20	0(5)	-	16%	15.8	0.2	1(4)	15 :17 :09	17%	15.6	0.2	1(3)	13 :08 :14	15%	15.75	0
N-50-25	0(5)	-	20%	18.2	0	0(5)	-	18%	18.6	0	0(4)	-	14%	19.0	0
W-40-10	5	00 :21 :04	0%	10.0	0.8	5	00 :12 :25	0%	10.0	0.6	5	00 :11 :04	0%	10.0	0.6
W-40-15	0(5)	-	3%	14.2	0.6	1(4)	12 :49 :02	3%	14.0	0.4	2(3)	06 :27 :19	4%	14.0	0.4
W-40-20	0(5)	-	7%	17.6	0.2	0(5)	-	9%	17.0	0.2	0(5)	-	7%	17.6	0
W-40-25	0(5)	-	8%	21.6	0	0(5)	-	9%	21.0	0	0(5)	-	10%	21.0	0
W-50-10	5	00 :21 :27	0%	8.4	0.8	5	00 :10 :33	0%	8.4	0.8	5	00 :08 :07	0%	8.4	0.6
W-50-15	2(3)	10 :18 :42	9%	12.2	0.2	3(2)	07 :05 :31	10%	12.2	0.2	3(2)	07 :22 :17	9%	12.0	0.2
W-50-20	0(5)	-	14%	16.0	0.2	1(4)	07 :02 :50	17%	15.6	0.2	1(4)	06 :57 :52	17%	15.6	0.2
W-50-25	0(5)	-	21%	18.8	0	0(5)	-	12%	19.33	0	0(5)	-	19%	18.8	0

TABLE 2.4. Special parts

Name	Reduced skills					Basic configurations					All skills				
	# Opt	CPU	Gap	# Tasks	# Idle	# Opt	CPU	Gap	# Tasks	# Idle	# Opt	CPU	Gap	# Tasks	# Idle
N-40-10	5	00 :00 :13	0%	9.8	0.6	5	00 :13 :59	0%	9.6	0.8	5	04 :32 :46	0%	10	0.8
N-40-15	5	00 :20 :42	0%	14	0.2	2(3)	16 :37 :55	5%	14	0.2	0(5)	-	1%	15	0
N-40-20	3(2)	02 :32 :16	4%	16.6	0	0(5)	-	10%	17	0	0(5)	-	2%	19	0
N-40-25	1(4)	06 :51 :25	9%	20.6	0	0(5)	-	12%	20.6	0	0(5)	-	4%	24	0
N-50-10	5	00 :00 :15	0%	8.4	0.2	5	00 :09 :02	0%	8.4	0.8	5	02 :04 :45	0%	10	0.8
N-50-15	5	00 :14 :10	0%	11.6	0	3(2)	08 :10 :29	10%	11.8	0.2	0(5)	-	1%	15	0
N-50-20	3(2)	01 :18 :27	7%	15	0	1(4)	15 :17 :09	17%	15.6	0.2	0(5)	-	2%	19.6	0
N-50-25	3(2)	16 :50 :36	7%	17.8	0	0(5)	-	18%	18.6	0	0(5)	-	5%	24	0.2
W-40-10	5	00 :00 :04	0%	10	0.2	5	00 :12 :25	0%	10	0.6	5	02 :44 :09	0%	10	0.8
W-40-15	5	00 :20 :30	0%	13.6	0.2	1(4)	12 :49 :02	3%	14	0.4	0(5)	-	1%	15	0
W-40-20	3(2)	04 :00 :09	4%	17.2	0	0(5)	-	9%	17	0.2	0(5)	-	1%	20	0
W-40-25	1(4)	21 :50 :40	10%	21	0	0(5)	-	9%	21	0	0(5)	-	2%	24.6	0
W-50-10	5	00 :00 :07	0%	8.4	0.2	5	00 :10 :33	0%	8.4	0.8	5	01 :49 :19	0%	10	0.8
W-50-15	5	00 :17 :07	0%	12	0	3(2)	07 :05 :31	10%	12.2	0.2	0(5)	-	1%	15	0
W-50-20	3(2)	02 :13 :02	13%	15.2	0	1(4)	07 :02 :50	17%	15.6	0.2	0(5)	-	2%	20	0
W-50-25	3(2)	06 :35 :34	7%	17.6	0	0(5)	-	12%	19.33	0	0(5)	-	6%	22.6	0

TABLE 2.5. Skills

Name	Basic configurations (30-45 min)					Reduced service times (15-30 min)					Reduced service times (10-20 min)				
	# Opt	CPU	Gap	# Tasks	# Idle	# Opt	CPU	Gap	# Tasks	# Idle	# Opt	CPU	Gap	# Tasks	# Idle
N-40-10	5	00 :13 :59	0%	9.6	0.8	5	00 :06 :53	0%	10	1	5	00 :08 :48	0%	10	1
N-40-15	2(3)	16 :37 :55	5%	14	0.2	2(3)	12 :22 :50	3%	14	0.6	2(3)	10 :52 :24	3%	14	0.6
N-40-20	0(5)	-	10%	17	0	1(4)	11 :54 :04	6%	17.4	0.2	1(4)	10 :51 :57	8%	17.4	0
N-40-25	0(5)	-	12%	20.6	0	0(5)	-	10%	21	0	0(4)	-	7%	21.75	0
N-50-10	5	00 :09 :02	0%	8.4	0.8	5	00 :11 :09	0%	8.4	0.8	5	00 :10 :47	0%	8.4	0.8
N-50-15	3(2)	08 :10 :29	10%	11.8	0.2	3(2)	05 :58 :33	9%	11.8	0.6	3(2)	05 :50 :03	9%	11.8	0.6
N-50-20	1(4)	15 :17 :09	17%	15.6	0.2	1(4)	09 :24 :39	16%	15.8	0.6	1(4)	09 :38 :45	16%	15.8	0.6
N-50-25	0(5)	-	18%	18.6	0	0(5)	-	15%	19.4	0	0(5)	-	16%	19.4	0
W-40-10	5	00 :12 :25	0%	10	0.6	5	00 :10 :07	0%	10	1	5	00 :11 :34	0%	10	1
W-40-15	1(4)	12 :49 :02	3%	14	0.4	3(2)	10 :33 :01	4%	14.4	0.8	3(2)	11 :35 :29	4%	14.4	0.8
W-40-20	0(5)	-	9%	17	0.2	0(5)	-	5%	17.8	0.4	0(5)	-	4%	18.2	0.4
W-40-25	0(5)	-	9%	21	0	0(5)	-	7%	21.6	0	0(5)	-	6%	22	0
W-50-10	5	00 :10 :33	0%	8.4	0.8	5	00 :10 :53	0%	8.4	1	5	00 :12 :42	0%	8.4	1
W-50-15	3(2)	07 :05 :31	10%	12.2	0.2	3(2)	05 :55 :33	9%	12.2	0.4	3(2)	06 :24 :20	10%	12.2	0.4
W-50-20	1(4)	07 :02 :50	17%	15.6	0.2	1(4)	07 :12 :34	18%	15.8	0.6	1(4)	07 :20 :44	18%	15.8	0.6
W-50-25	0(5)	-	12%	19.33	0	0(5)	-	18%	18.4	0	0(5)	-	16%	19.2	0

TABLE 2.6. Repair times

Name	Basic configurations (3 technicians)					More technicians (4 technicians)				
	# Opt	CPU	Gap	# Tasks	# Idle	# Opt	CPU	Gap	# Tasks	# Idle
N-40-10	5	00 :13 :59	0%	9.6	0.8	5	00 :28 :40	0%	10	1.6
N-40-15	2(3)	16 :37 :55	5%	14	0.2	1(4)	23 :13 :39	2%	14.8	0.6
N-40-20	0(5)	-	10%	17	0	0(5)	-	2%	20	0
N-40-25	0(5)	-	12%	20.6	0	0(5)	-	3%	24	0
N-50-10	5	00 :09 :02	0%	8.4	0.8	5	00 :09 :08	0%	10	1
N-50-15	3(2)	08 :10 :29	10%	11.8	0.2	4(1)	10 :45 :13	4%	14.6	0
N-50-20	1(4)	15 :17 :09	17%	15.6	0.2	0(5)	-	5%	18	0
N-50-25	0(5)	-	18%	18.6	0	0(5)	-	8%	22	0
W-40-10	5	00 :12 :25	0%	10	0.6	5	00 :25 :42	0%	10	1.6
W-40-15	1(4)	12 :49 :02	3%	14	0.4	1(4)	16 :52 :17	1%	15	0.6
W-40-20	0(5)	-	9%	17	0.2	0(5)	-	2%	20	0.2
W-40-25	0(5)	-	9%	21	0	0(5)	-	4%	24.2	0
W-50-10	5	00 :10 :33	0%	8.4	0.8	5	00 :08 :10	0%	10	1
W-50-15	3(2)	07 :05 :31	10%	12.2	0.2	4(1)	08 :14 :21	5%	14.6	0
W-50-20	1(4)	07 :02 :50	17%	15.6	0.2	0(5)	-	4%	18.4	0
W-50-25	0(5)	-	12%	19.33	0	0(5)	-	11%	22.2	0

TABLE 2.7. Number of technicians

	Without warm start			**With warm start**		
Name	# Opt	CPU	Gap	# Opt	CPU	Gap
N-40-20	0(5)	-	10%	0(5)	-	8.1%
N-40-25	0(5)	-	12%	0(5)	-	9.4%
N-50-20	1(4)	15 :17 :09	17%	1(4)	12 :43 :31	14.9%
N-50-25	0(5)	-	18%	0(5)	-	14.7%
W-40-20	0(5)	-	9%	0(5)	-	7.1%
W-40-25	0(5)	-	9%	0(5)	-	7.4%
W-50-20	1(4)	07 :02 :50	17%	1(4)	06 :31 :39	14.7%
W-50-25	0(5)	-	12%	0(5)	-	13%

TABLE 2.8. Warm start results

Chapitre 3

UN ALGORITHME DE BRANCH-AND-PRICE POUR UN PROBLÈME MULTI-ATTRIBUTS DE TOURNÉES DE TECHNICIENS

Ce chapitre correspond à un article qui a été soumis pour publication dans la revue *Journal on Vehicle Routing Algorithms*. Les co-auteurs sont I. Mathlouthi, M. Gendreau et J.-Y. Potvin.

Dans ce chapitre, nous introduisons une méthode exacte de branch-and-price pour un problème multi-attributs de tournées de techniciens, qui correspond à celui étudié au chapitre 2. Nous introdusions en particulier un nouveau schéma de branchement ternaire qui exploite le fait que les routes des techniciens ne sont pas obligées de couvrir toutes les tâches. Les résultats des expériences montrent que notre algorithme peut résoudre des instances allant jusqu'à 45 tâches, ce qui améliore grandement les résultats rapportés dans le chapitre précédent.

Branch-and-Price for a Multi-Attributes Technician Routing and Scheduling Problem

In this paper, we present an exact branch-and-price algorithm for a multi-attribute technician routing and scheduling problem. This problem integrates a number of distinctive features from a real-world application, like the management of an inventory of parts and multiple time windows for service. A new ternary branching structure is introduced within the branch-and-price algorithm, based on the fact that not all tasks need to be performed. The computational results show that our algorithm can solve instances with up to 45 tasks and greatly improves upon a commercial solver applied to an arc-based mixed integer program.

Keywords : Routing, technicians, multiple time windows, inventory of parts, branch-and-price, column generation, elementary shortest path with resource constraints, ternary branching.

3.1. Introduction

The Technician Routing and Scheduling Problem (TRSP), which is a special occurrence of the workforce scheduling and routing problem, is found in many real-world applications, although it has received only limited attention in the scientific literature. In this paper, we tackle a TRSP previously introduced in [45]. This problem is motivated by an application for the maintenance and repair of electronic transactions equipment. It can be defined as follows. There is a set of technicians with different skills who are available to carry out a number of different tasks. Each task has a priority level, depending on its urgency or the customer's importance. One must then assign tasks to the technicians and build a route for each technician, starting and ending at his home base location, so as to minimize an objective that accounts for overtime, total traveled distance, and total gain over the performed tasks. The solution must also satisfy constraints related to the technicians' skills required to perform their assigned tasks, working hours and multiple service time windows associated with each task.

Two particular features distinguish our problem from other TRSPs. First, the route schedules must account for technicians' breaks. Second, a task may consume one or more parts. Thus, an inventory of spare parts is carried by each vehicle and the technician must replenish (at most once) at a pre-assigned depot if the number of parts in the inventory does not allow all his assigned tasks to be done. The visit at the pre-assigned depot does not need to take place at the start of the route, as long as no stock out occurs. This visit can also be used to get expensive special parts if they are needed for some tasks (at most one special

part per task). As opposed to spare parts, special parts are not carried by technicians unless they are needed.

In [45], a mixed integer programming (MIP) model was developed for this problem and solved with a commercial solver. But this approach only allowed very small instances to be solved. This work is now aimed at developing a branch-and-price algorithm to allow larger instances to be solved exactly in reasonable computation times.

The rest of this paper is organized as follows. In Section 2, we review some related work. In Section 3, the problem is defined. Then, the solution method is described in Section 4. Results on test instances of different sizes are reported and compared with those obtained with a commercial solver in Section 5. Finally, Section 6 concludes the paper.

3.2. Related Work

Research on the TRSP began with the work of Tsang and Voudouris [67] in 1997. The authors address a problem faced by the technician work force of British Telecom, using guided local search and a fast local search. In this problem, the service time is related to the technician's experience.

Weigel and Coo [71] report a problem faced by an American retailer. In this paper, tasks are first assigned to technicians and routes are then optimized individually using Or-Opt exchanges [48].

Xu and Chiu [73] consider a problem faced by service providers in the telecommunications industry. The objective in this work is to maximize the number of served tasks while taking into account task priority, technician skills and overtime. Four heuristics based on local search and GRASP are reported. An extension of this problem is considered in [13] by taking into account new features, like time windows. A customized biased random key genetic algorithm meta-heuristic is proposed to solve the problem

Blakeley et al. tackle a problem faced by the Schindler Elevator Corporation [5]. Their aim is to schedule periodic maintenance operations while taking into account technician skills and work regulations. Their sophisticated system employs various operations research techniques to assign maintenance work to technicians and construct their routes. Tang et al. [64] address a similar application and solve it with a tabu search heuristic. Another multi-period TRSP for a maintenance provider specialized in electric forklifts is solved with a branch-and-price algorithm in Zamorano and Stolletz [75]. In this application, the technicians are paired into teams to perform tasks and the branching strategy exploits this particularity.

The French Operations Research Society (ROADEF) proposed a challenge based on a problem encountered by France Telecom where routes for technicians must be scheduled on a multiple day horizon and where each task requires one or more skills. In this problem, teams of technicians are created to serve the maximum number of tasks. Based on this challenge,

Hashimoto et al. [33], developed a greedy randomized adaptive search procedure (GRASP), while Cordeau et al. [11] used an adaptive large neighborhood search (ALNS).

In 2008, Bostel et al. [7] address a dynamic TRSP faced by Veolia, a water treatment and distribution company. In this problem, technician routes must be planned over a period of one week for repair or maintenance. The tasks to be scheduled can either be known in advance (preventive maintenance) or can occur dynamically. Each task has a time window for service. The first proposed method is a memetic algorithm, which is first applied to static tasks to produce tours for every day of the week. Dynamic tasks are then integrated into the solution as they occur, using the memetic algorithm. The second approach is based on a column generation algorithm which can only be applied to problem instances of small size.

Pillac et al. [50] also address a TRSP in which a fraction of the tasks occurs dynamically. A parallel architecture is proposed to speed up the calculations. An initial solution is first created for known tasks using a regret heuristic [54]. This solution, is improved by ALNS. The latter successively destroys (removes tasks) and repairs the current solution (reinserts tasks). When a new task occurs, the part of the solution that is already executed is fixed and the new task is incorporated into the solution by running again the ALNS for a limited number of iterations.

The next section will describe the static multi-attribute TRSP studied in this paper.

3.3. Problem Definition

Here, we introduce the new TRSP variant proposed by a company that provides maintenance and repair of electronic transactions equipment. The three main entities involved in this problem are the following :

— Technicians : Each technician has skills which allow him to perform certain tasks and not others. His normal workday extends from 9 :00 AM to 5 :00 PM, if there is no overtime, and his route starts and ends at his home base. Furthermore, his route should not exceed a maximum distance. Each technician can take three breaks during the day : one break of 15 minutes in the morning and afternoon, respectively, and a mid-day break of 30 minutes (where morning, afternoon and mid-day are defined through time intervals). Also, it is not possible for a technician to take a break just before returning to his home base. Each technician starts with an initial inventory of four different types of spare parts. Along the route, a pre-assigned depot can be used to replenish the inventory of spare parts and also get any needed special parts. A fixed capacity is associated with each type of spare parts stored in the vehicle.

— Tasks : It is assumed that tasks can be postponed, so there is no need to serve all tasks. Each task has a gain which stands for its priority. A task also has a service time and multiple possible time windows for service. Finally, it is characterized by the

types and number of spare parts of each type, as well as (possibly) one special part, needed by the technician to perform it ;

— Depots : there are many depots with a (virtually infinite) number of parts.

The problem is to design technician routes for one day, where each route starts and ends at the technician's home location and serves a subset of tasks, including one possible stop at a preassigned depot, while satisfying the required skills for each task, the multiple time windows at each location, the time window for each break, the maximum traveled distance of each route, the required number of spare parts for each task with, possibly, the requirement of a special part. The objective is to minimize a weighted sum of the total traveled distance, total overtime (minus) total gain over all performed tasks. An arc-based MIP for this problem can be found in [45]. In the latter work, a penalty term is also added in the objective to favor the insertion of breaks into the route schedules. Here, there is no such term, because a technician is forced to take a break if his schedule covers the time window associated with that break (which corresponds to the policy used in practice).

The number of variables in the MIP quickly increases with problem size. To alleviate this scalability issue, we propose here an alternative path-based formulation which is addressed through a branch-and-price algorithm, as explained in the next section.

3.4. Branch-and-Price

The branch-and-price algorithm for solving our TRSP is described below, by first introducing the column generation scheme and then the branching scheme(s).

3.4.1. Column generation

Column generation first requires the definition of a master problem. This is the topic of the first subsection. Then, we explain how column generation is realized through the definition of an appropriate pricing problem.

3.4.1.1. *Master problem*

The master is formulated as a set packing problem given that not all tasks need to be served. The decision variables (columns) in this model correspond to possible feasible routes for each technician. Note that there is a separate subset of columns for each technician because the home location, the skills and the preassigned depot are not necessarily the same. With each route is associated a cost. If δ_r, d_r, g_r denote the overtime, distance and gain over served tasks, respectively, for route r then the cost c_r of this route can be written as :

$$c_r = \psi\delta_r + \vartheta d_r - \varphi g_r \tag{3.4.1}$$

where ψ, ϑ and φ are weighting parameters.

To define the master problem, let us denote I the set of tasks, K the set of technicians, R the set of all feasible routes and $R_k \subseteq R$ the subset of feasible routes for technician k. Also, parameter a_{ir} is 1 if task i is part of route r, 0 otherwise. The decision variables are x_r, $r \in R$, where x_r is 1 if route r is in the solution, and 0 otherwise. The model is then the following :

$$(MP) \quad \text{Min} \sum_{r \in R} c_r x_r \tag{3.4.2}$$

subject to

$$\sum_{r \in R} a_{ir} x_r \leq 1 \qquad i \in I \tag{3.4.3}$$

$$\sum_{r \in R_k} x_r \leq 1 \qquad k \in K \tag{3.4.4}$$

$$x_r \in \{0,1\} \qquad r \in R \tag{3.4.5}$$

In this model, the objective is to minimize the total route cost. Constraints (3.4.3) ensure that each task is served by at most one route and constraints (3.4.4) ensure that each technician performs at most one route.

3.4.1.2. *Pricing problem*

In practice, the number of feasible routes R is so large that it is not possible to directly solve the master problem presented above. Thus, we have to resort to column generation, where only a subset of feasible routes (columns) is considered in a so-called restricted master problem (RMP). Starting with an initial set of columns, the linear relaxation of the corresponding RMP is first solved to obtain values for the dual variables. With these dual values, new columns of negative reduced costs are identified by solving a pricing problem and are added to the RMP. The linear relaxation of the augmented RMP is then solved again to obtain new dual values. This is repeated until no column with negative reduced cost can be found. At this point, we have the optimal solution of the linear relaxation of the master problem, which is not necessarily integer.

The initial set of columns is obtained by solving the TRSP with a simple greedy insertion heuristic. The routes are constructed for each technician in turn. At each iteration, the route of the current technician is augmented by adding the unserved task that leads to the smallest increase to the objective value. This is repeated until no more feasible insertion in the route is possible.

Now, let us denote λ_i and μ_k the optimal values of the dual variables for constraints (3.4.3) and (3.4.4), respectively, and $I_r = \{i \in I : a_{ir} = 1\}$. Then, the reduced cost c'_r of route (column) r is

$$c'_r = c_r - \mu_k - \sum_{i \in I_r} \lambda_i \tag{3.4.6}$$

In our column generation algorithm we have one pricing problem per technician to identify a route of negative reduced cost for this technician. The pricing problem corresponds to an elementary shortest path problem with resource constraints (ESPPRC) starting from the home location of the technician and back to it. Apart from the starting and ending home location of the technician, the network also comprises nodes for the pre-assigned depot, the subset of tasks that the technician can perform and the three breaks. The shortest path problem is solved using the well-known dynamic programming approach reported in [22]. In this algorithm, the nodes are assigned labels that represent partial paths used to reach them from the starting node. A label indicates the cost of the path as well as the consumption of resources (e.g., distance, time). More precisely, each label stores the following information on a partial path :

— C : cost ;
— T : time (duration) ;
— D : distance ;
— SP^t : number of remaining spare parts of type t = 1, 2, 3, 4 ;
— F : indicator set to 0 if the depot has already been visited, 1 otherwise.
— V : set of unreachable nodes ;

It should be noted that a node is said to be unreachable if it has already been visited along the partial path, or if there is a resource such that its consumption prevents the node to be reached. In our case, a node (task) may be unreachable due to (1) the time windows (2) the maximum traveled distance or (3) some required part(s) are unavailable and the depot has already been visited.

The labels are used to reduce the number of partial paths that need to be considered during the execution of the dynamic programming algorithm. Basically, a dominance relation is established among the partial paths at a given node through their corresponding labels. If a path p is dominated at some node i, then any extension to an immediate successor of i will produce a path that will also be dominated. Thus, only non-dominated paths need to be considered. For two partial paths p_1 and p_2 from the home location of a technician to a given node i with different labels (C_1,T_1,D_1,SP_1,F_1,V_1) and (C_2,T_2,D_2,SP_2,F_2, V_2), respectively, p_1 dominates p_2 if :

— $C_1 \leq C_2$;
— $T_1 \leq T_2$;

— $D_1 \leq D_2$;
— $SP_1^t \geq SP_2^t$ for spare parts of type $t = 1, 2, 3, 4$;
— $F_1 \geq F_2$
— $V_1 \subseteq V_2$

From an implementation point of view, we also store in the label the number of unreachable nodes s. It is used as a quick filter, since $s_1 \leq s_2$ is required for the condition $V_1 \subseteq V_2$ to be satisfied. Also, the subset of unreachable nodes is represented by a binary vector whose size corresponds to the number of nodes and each entry in the vector is 1 if the associated node is in the subset, 0 otherwise. Then, we have $V_1 \subseteq V_2$ if each entry in the vector for path p_1 is less than or equal to the corresponding entry in the vector for path p_2.

With regard to the technicians' breaks, the path extension works as follows when going from i to j. If we are in the time window of a given break at j and this break has not been taken yet then (1) if it is not possible to insert the break after j without exceeding the break's upper bound, then the break is inserted between i and j and (2) if the break can be inserted after j, the two following cases are accounted for : the technician goes directly from i to j or the break is inserted between i and j.

3.4.1.3. *Decremental State-Space Relaxation*

The column generation algorithm presented above can be improved by relaxing the elementary requirement in the ESPPRC pricing problem. This is known as Decremental State-Space Relaxation (DSSR) [56]. At each execution of the dynamic programming algorithm, the relaxation is tightened by considering nodes involved in a cycle as critical and by forbidding them to be visited more than once. The set of critical nodes thus grows from one execution to the next until an elementary path is obtained. With regard to the implementation, we have a vector φ in the label for the set of unreachable and critical nodes. Then, the last condition $V_1 \subseteq V_2$ for two paths p_1 and p_2 in the previous dominance rule becomes :

— $\varphi_1 \subseteq \varphi_2$

As before, we also maintain the number of unreachable and critical nodes for quick filtering purposes.

3.4.2. Branching

As mentioned above, the column generation algorithm does not necessarily lead to an integer solution, given that a linear relaxation of the current RMP is solved at each iteration. Thus, branching may be required to identify an optimal integer solution. Column generation is first applied at the root node of the search tree. If the solution obtained is integer, then we have the optimal solution of our TRSP and we stop here. Otherwise, we must branch on a fractional variable to create child nodes [69]. The search is then executed by selecting one open node (i.e., a node already generated but not solved yet) and by applying column

generation to the corresponding problem. The algorithm stops when all open nodes are done. The best integer solution found during the search is then the optimal solution of our TRSP.

In the literature, a popular approach is to branch on an arc variable whose (fractional) value corresponds to the flow on that arc. By setting the variable to either 0 or 1, two new child nodes are generated. This additional constraint can be easily integrated into the master and pricing problems at each child node [27]. Unfortunately, the branch where the variable is set to 0 is very weak since it forbids only one arc. In this paper, we exploit the fact that not all tasks need to be performed.

Let us define $y_{ik} = 1$ if task i is assigned to technician k, 0 otherwise. It should be noted that the value of y_{ik} can be obtained by summing the variables x_r in the RMP for which task i is served in route $r \in R_k$. Two different branching strategies based on these variables are tested in the computational results :

Binary branching. Here, two child nodes are created when there is one or more fractional variables y_{ik} in the solution. We first select the variable closest to 0.5 and set it to 1 on one branch and to 0 on the other branch. In the first case, when $y_{ik} = 1$, task i is performed by technician k. Thus, after solving the pricing problem of technician k, we consider only non dominated routes of negative reduced cost that cover task i. In the second case, when $y_{ik} = 0$, task i is not performed by technician k (i.e., the task is either performed by another technician or not performed at all). Accordingly, there is no node for task i in the pricing problem of technician k, which is much stronger than forbidding a single arc.

Ternary branching. Here, three child nodes are created when there is one or more fractional variables y_{ik} in the solution. One branch stands for $y_{ik} = 1$, as in the binary branching. The second branch stands for $\sum_{k' \in K} y_{ik'} = 0$, that is, neither technician k nor any other technician performs task i. Accordingly, there is no node for task i in the pricing problem of each technician. The third branch stands for $y_{ik} = 0$ and $\sum_{k' \in K \setminus \{k\}} y_{ik'} = 1$. Here, although technician k does not perform task i, some other technician does. In this case, there is no node for task i in the pricing problem of technician k and the constraint $\sum_{r \in R \setminus R_k} a_{ir} x_r = 1$ replaces the constraint $\sum_{r \in R} a_{ir} x_r \leq 1$ in the master problem to force some other technician to perform the task.

One could wonder if an integer solution of our TRSP is guaranteed when all variables y_{ik} are integer. This is indeed the case as demonstrated below.

Proposition. Let X be a solution at a node of the branching tree. If there is no task shared by two or more technicians (i.e., y_{ik} is either 0 or 1, $i \in I$, $k \in K$), then X is integer.

Proof. Let us define I_k the set of tasks served by technician $k \in K$ in solution X and $R_k^* = \{r \in R_k : x_r > 0\}$ the set of routes of technician k used in solution X. Since, by hypothesis, no task is shared by two or more technicians for each task $i \in I_k$, we have :

$$\sum_{r \in R_k^*} a_{ir} x_r = 1 \quad i \in I_k,\, k \in K\,; \tag{3.4.7}$$

$$\sum_{r \in R_k^*} a_{ir} x_r = 0 \quad \text{if } i \notin I_k,\, k \in K\,; \tag{3.4.8}$$

To prove that X is integer, we show that all routes (columns) in R_k^* are the same for any given technician $k \in K$, that is :

$$a_{ir} = 1 \quad i \in I_k,\, r \in R_k^*\,; \tag{3.4.9}$$

$$a_{ir} = 0 \quad i \notin I_k,\, r \in R_k^*\,; \tag{3.4.10}$$

If this is true, R_k^* contains a single route r for technician k and the corresponding variable $x_r = 1$ (otherwise, we would have two identical columns in the basis). Hence, X is integer.

By contradiction, let us suppose that equations (3.4.9) and (3.4.10) do not hold for some technician k. Thus, there is at least two routes, r', $r'' \in R_k^*$, $r' \neq r''$, such that $a_{i^*r'} \neq a_{i^*r''}$ for at least one task i^*. Two cases are possible :

If $i^* \notin I_k$ then $a_{i^*r'} = 1$ or (exclusive) $a_{i^*r''} = 1$. Since r', $r'' \in R_k^*$, we have $x_{r'} > 0$, $x_{r''} > 0$ and equation (3.4.8) cannot hold.

If $i^* \in I_k$ then, without loss of generality, let us suppose that $a_{i^*r'} = 1$ and $a_{i^*r''} = 0$. We define $R_k^{i^*} = \{r \in R_k^* | a_{i^*r} = 1\}$. By equation (3.4.7) :

$$\sum_{r \in R_k^{i^*}} x_r = 1$$

and

$$x_{r''} + \sum_{r \in R_k^{i^*}} x_r > 1$$

because $r'' \in R_k^*$ and, consequently, $x_{r''} > 0$. But, it contradicts the fact that :

$$\sum_{r \in R_k^*} x_r \leq 1 \quad k \in K$$

since each technician can serve at most one route, QED.

3.5. Computational results

In the following, we report computational results obtained with our branch-and-price algorithm. First, the test instances are described. Then, the computational behavior of our algorithm on different types of test instances is reported. Finally, a comparison with the mathematical model in [45], when solved with CPLEX, is presented. It should be noted

that the latter model was slightly modified to comply with the inventory constraints stated in Section 3.3, where a fixed vehicle capacity for each type of special parts should not be exceeded (in [45], the parts collected at the depot were simply added to those already present in a vehicle).

3.5.1. Test instances

The test instances were generated as in [45]. Their characteristics, with values that are closely related to the real-world application, are presented in Table 3.1. At the end, we have considered every possible combination of the basic values shown in Table 3.2, for a total of $2 \times 2 \times 8 = 32$ subsets of instances, with 5 instances in each subset. To evaluate the performance of the tested problem-solving methods along various dimensions, other values were also considered for Skills, Service time, Special part and # Technicians, as indicated in Table 3.3. When a new value is tested for one of them, the other characteristics are fixed at their basic value. With regard to the skill configuration 50% (2 technicians) and 25% (1 technician), it should be noted that the second technician with 50% of all tasks automatically takes the tasks that cannot be performed by the first technician. In this way, every task can be served by at least one technician. Also, when the number of technicians is 4, the fourth technician can perform 25% of all tasks. Thus, the skill configuration is 100% (1 technician), 50% (1 technician) and 25% (2 technicians).

In all test instances, the crow fly distance is assumed between each pair of locations, the speed of the vehicles is set at 50km/h and the maximum distance of each technician route is set at 125km. With regard to the objective function (3.4.1), the weights are set to $\varphi = 500$ (gain), $\vartheta = 5$ (distance in kilometers) and $\psi = 1$ (overtime in seconds). In particular, this setting allows a technician to do some overtime to perform a high gain task and leads to more challenging instances than the weighting scheme proposed in [45].

3.5.2. Experiments

This section reports the results obtained on the instances introduced in the previous section. Our branch-and-price algorithm was given a maximum of 24 hours of computation time on a 3.07GHz Intel Xeon X5675 processor. The linear relaxation of the restricted master problems were solved with CPLEX 12.6. The tables of results provide the following information :

— *Name* : Name of each subset of 5 instances using the format X-Y-Z, where X is the size of the time windows (either N for narrow or W for wide), Y the size of the service area (either 40 for 40km × 40km or 50 for 50km × 50 km) and Z the number of tasks (either 10, 15, 20, 25, 30, 35, 40 or 45).

— *Opt* : Number of instances solved to optimality at the root (number of instances solved to optimality after branching) ;

Service area	The service area corresponds to a 40km × 40km or 50km × 50km squared area.
Depot location	There are 3 depots randomly located within the service area.
Task location	Each task is randomly located in the service area.
Task service time	The service time of each task is randomly chosen between 30 and 45 minutes.
Task gain	The gain of each task is randomly chosen between 1 and 10.
Technician home base	The first two technicians are located at the opposite ends of the service area (along the diagonal). The other technicians are randomly located within the service area.
Technician skills	With each technician is associated the percentage of tasks that he can perform, which is chosen from 100%, 50% and 25%.
Parts	The number of spare parts needed to perform a task is randomly chosen between 0 and 3. Then, each spare part is assigned a type, among 4 different types. A special part is needed with a probability of 0.125
Time windows	Both narrow and wide time windows are considered. The latter are twice as wide as the former on average. The length of a narrow time window is randomly generated between 60 and 90 minutes. The lower bound of the first time window is chosen randomly between 9 :00 AM and noon. The lower bounds of the remaining time windows are randomly chosen from 2 to 3 hours after the upper bound of the previous time window until a maximum of 3 time windows are obtained.

TABLE 3.1. Characteristics of the test instances

— *CPU* : Average computation time in seconds (over instances solved to optimality) ; when the time limit of 24 hours is reached on every instance in a subset, the entry is marked with a "-" sign.

— *Nodes* : Average number of nodes in the branching tree (over instances solved to optimality), excluding the root ; when the time limit of 24 hours is reached on every instance in a subset, the entry is marked with a "-" sign.

Table 3.4 reports the results obtained on the test instances generated with the basic parameter values shown in Table 3.2. We tested the branch-and-price algorithm with the

Time windows	Narrow, Wide
Service Area	40km × 40km, 50km × 50km
# Tasks	10, 15, 20, 25, 30, 35, 40, 45
# Technicians	3
Skills (% Tasks)	100% (1 technician), 50% (1 technician), 25 % (1 technician)
Service time	30-45 minutes
Special part (prob.)	0.125

TABLE 3.2. Basic parameter values

Skills 1	100% (3 technicians)
Skills 2	50% (2 technicians), 25% (1 technician)
Service time 1	15-30 minutes
Service time 2	10-20 minutes
Special part 1 (prob.)	0
Special part 2 (prob.)	0.25
# Technicians	4

TABLE 3.3. Other parameter values

standard pricing problem (ESPPRC) and the relaxed one (DSSR), using the binary and ternary branching schemes.

We first note that instances with up to 45 tasks can be solved to optimality with our algorithms. Also, the instances with narrow time windows and a service area of 50 km × 50 km are easier to solve because they are more constrained. In the latter case, it should be noted that the maximum distance constraint on each route plays a more significant role when the service area is larger. In general, DSSR provides an improvement over ESPPRC by allowing more instances to be solved to optimality and by requiring less computation time. Although we do not show the details here, the improvement provided by DSSR with ternary branching over the other tested variants increases on more difficult instances obtained by using the parameter values shown in Table 3.3, for example when all technicians have the required skills to perform all tasks (see below).

Based on these results, we will focus in the following on the branch-and-price algorithm with DSSR and ternary branching. We will first examine the results obtained when a parameter, either Skills, Service time, Special part or # Technicians is set to a new value in Table 3.3, while the other parameters are fixed to their basic value. In the corresponding tables of results, the following additional information is provided :

— *Tasks* : Average number of tasks performed by the technicians ;

— *Idle* : Average number of idle (unassigned) technicians.

Impact of special parts

Table 3.5 reports the results obtained when the probability that a special part is required to perform a task is reduced from the basic value .125 to 0 or increased from .125 to .25. When no special part is required, a visit to the depot is less likely and the problem becomes less constrained. Hence, the problem becomes more difficult. Accordingly, we observe an increase in computation times in this case, although the number of instances solved to optimality does not change much.

Impact of skills

Table 3.6 reports the results obtained when we restrict or enlarge the subset of tasks that each technician can perform. We observed that this characteristic has the largest impact on the performance of the branch-and-price algorithm. For example, if we assume that every technician has all the required skills to perform all tasks, which corresponds to *Skills 1* in Table 3.3, the algorithm cannot solve instances with more than 30 tasks for the subsets with wide time windows and 35 tasks for the subsets with narrow time windows. Conversely, when the skills are reduced according to *Skills 2* in Table 3.3, all instances are solved to optimality in much less computation time. It is even possible to solve instances with 50 tasks, although these results are not reported here for brevity purposes.

Impact of service times

Table 3.7 reports the results when the service time is decreased with regard to the basic configurations. Instead of randomly choosing the time required to perform a task in the interval 30-45 minutes, the configurations *Service Time 1* and *Service Time 2* in Table 3.3 choose a time in the intervals 15-30 minutes and 10-20 minutes, respectively. It should be noted that reducing the service time increases the combinatorial complexity because each technician has now more flexibility for serving tasks. As observed in Table 3.7, the computation times tend to increase with reduced service times, although there is no significant impact on the number of instances solved to optimality.

Impact of number of technicians

Table 3.8 shows the results obtained when the number of technicians is increased from 3 to 4. The problem becomes more difficult, as shown by the computation times and the number of instances solved to optimality (19 fewer instances solved to optimality with 4 technicians). As expected, both the number of tasks performed by the technicians and the number of idle technicians increase.

Overall, the four parameters considered above have an impact on the performance of our branch-and-price algorithm, with the technicians' skills being the most sensitive.

Name	ESPPRC						DSSR					
	Binary Branching			Ternary Branching			Binary Branching			Ternary Branching		
	Opt	CPU	Nodes	Opt	CPU	Nodes	Opt	CPU	Nodes	Opt	CPU	Nodes
N-40-10	5	0.3	0	5	0.3	0	5	0.3	0	5	0.4	0
N-40-15	2(3)	0.9	4.7	2(3)	0.9	9	2(3)	1	3.3	2(3)	1.1	93
N-40-20	3(2)	5.8	14	3(2)	5.1	12	3(2)	4.8	10	3(2)	3.4	120
N-40-25	2(3)	1312.5	104.7	2(3)	1289.6	12	2(3)	53.6	24.7	2(3)	44.8	282
N-40-30	1(1)	1223.5	1022	1(1)	1189.2	282	2(1)	85.1	30	2(1)	69.4	363
N-40-35	0(2)	1518.3	1534	0(2)	1498.2	363	0(2)	334.2	126	0(2)	241.4	2185.5
N-40-40	1(1)	1813.2	2046	1(1)	1066.6	120	1(1)	428	254	1(1)	280.1	3279
N-40-45	0(0)	-	-	0(0)	-	-	0(0)	-	-	0(0)	-	-
N-50-10	4(1)	0.5	2	4(1)	0.6	3	4(1)	0.5	2	4(1)	0.7	3
N-50-15	4(1)	0.6	6	4(1)	0.6	3	4(1)	0.4	6	4(1)	0.7	12
N-50-20	5	1	0	5	1.1	0	5	0.5	0	5	0.3	0
N-50-25	4(1)	1.2	6	4(1)	0.7	3	4(1)	0.6	6	4(1)	0.4	39
N-50-30	4(0)	20.3	0	4(1)	5.2	12	4(1)	6.7	14	4(1)	4.6	39
N-50-35	2(2)	35.1	30	2(2)	29.2	25.5	2(2)	26.1	22	2(2)	22.9	241.5
N-50-40	3(0)	35.1	0	3(0)	34.8	0	3(1)	33.6	30	3(1)	30.4	363
N-50-45	1(2)	110.6	46	1(3)	105.2	39	1(3)	73.5	40.7	1(3)	67.5	606
W-40-10	4(1)	0.5	2	4(1)	0.5	3	4(1)	0.5	2	4(1)	0.3	12
W-40-15	2(3)	2.6	8.7	2(3)	2.2	6	2(3)	1	4.7	2(3)	1	12
W-40-20	4(1)	14.9	14	4(1)	14.4	12	4(1)	14.5	14	4(1)	13.8	39
W-40-25	1(4)	1557.6	1790	1(4)	1408.7	302.3	1(4)	95.2	38	1(4)	91.8	909.8
W-40-30	1(1)	57.2	30	1(1)	50.4	39	1(1)	27.9	30	1(1)	25.2	120
W-40-35	1(0)	3221.9	0	1(1)	3422.4	3279	1(1)	197	62	1(1)	177.7	1092
W-40-40	0(0)	-	-	0(1)	3554.7	3279	0(1)	242.8	126	0(1)	239.9	3279
W-40-45	0(0)	-	-	0(0)	-	-	0(0)	-	-	0(0)	-	-
W-50-10	4(1)	0.5	2	4(1)	0.6	3	4(1)	0.6	2	4(1)	0.6	3
W-50-15	3(2)	0.5	2	3(2)	0.7	3	3(2)	0.4	6	3(2)	0.5	12
W-50-20	4(1)	1.7	6	4(1)	1.9	12	4(1)	1.7	14	4(1)	1.7	12
W-50-25	5	7.6	0	5	6.6	0	5	5.2	0	5	5.8	0
W-50-30	4(0)	18.8	0	4(0)	17.7	0	4(0)	17.6	0	5	15.1	0
W-50-35	3(0)	146.7	0	3(0)	148.5	0	3(0)	27.3	0	3(0)	28.3	0
W-50-40	0(2)	293.6	94	0(3)	280.8	66	1(2)	180.6	94	1(3)	178.7	2550
W-50-45	2(0)	440	0	2(1)	354.8	120	2(0)	237.9	0	2(1)	233.8	3279

TABLE 3.4. Basic Configurations

Name	No special parts (0)					Basic configurations (.125)					More special parts (.25)				
	Opt	CPU	Nodes	Tasks	Idle	Opt	CPU	Nodes	Tasks	Idle	Opt	CPU	Nodes	Tasks	Idle
N-40-10	4(1)	0.5	2	9.8	1	5	0.4	0	9.8	1	5	0.3	0	9.8	1
N-40-15	2(3)	2.1	9	13.4	0.4	2(3)	1.1	93	14	0.2	2(3)	0.6	9	14.4	0.4
N-40-20	2(3)	11	21	16.4	0	3(2)	3.4	120	17	0	4(1)	3.8	12	17.6	0
N-40-25	1(3)	174.6	30	19.8	0	2(3)	44.8	282	19.6	0	3(2)	30.3	25.5	19	0
N-40-30	2(3)	386.1	174	20.8	0	2(1)	69.4	363	20.7	0	2(3)	50.7	66	20.8	0
N-40-35	3(2)	1163.6	241.5	22	0	0(2)	241.4	2185.5	23	0	3(2)	151.8	241.5	22	0
N-40-40	1(1)	1168.1	363	23.5	0	1(1)	280.1	3279	23.3	0	1(1)	246.6	363	23.5	0
N-40-45	1(1)	1223.8	363	25.5	0	0(0)	-	-	-	3	1(1)	945.6	1092	25.5	0
N-50-10	4(1)	0.8	3	8.2	1	4(1)	0.7	3	8.4	1	5	0.4	0	8.4	1
N-50-15	4(1)	0.4	3	12	0.4	4(1)	0.7	12	12	0.2	3(2)	0.4	7.5	12.2	0.2
N-50-20	5	1.8	0	15	0.2	5	0.3	0	15.6	0.2	3(2)	0.3	12	15	0.2
N-50-25	3(2)	14.7	25.5	18.4	0	4(1)	0.4	39	18.6	0	2(3)	0.4	9	17.2	0
N-50-30	4(1)	15	39	20	0	4(1)	4.6	39	19.7	0	4(1)	4.1	12	19.6	0
N-50-35	2(3)	96.7	57	21.2	0	2(2)	22.9	241.5	20.5	0	2(3)	11.4	21	20.8	0
N-50-40	1(3)	139	174	22.5	0	3(1)	30.4	363	22.5	0	3(2)	22.1	39	21.4	0
N-50-45	2(2)	146.1	241.5	24.5	0	1(3)	67.5	606	23.5	0	3(1)	52.8	120	23.5	0
W-40-10	4(1)	0.5	3	9.8	1	4(1)	0.3	12	10	1	4(1)	0.4	3	9.8	1
W-40-15	1(4)	1.5	9.8	13.4	0.6	2(3)	1	12	14	0.4	2(3)	0.8	12	13.6	0.4
W-40-20	3(2)	60.2	79.5	17	0	4(1)	13.8	39	17	0	4(1)	8.1	12	16.8	0
W-40-25	1(2)	109.2	120	20.7	0	1(4)	91.8	909.8	21	0	3(0)	25.6	-	19.3	0
W-40-30	3(1)	127	120	21.3	0	1(1)	25.2	120	22	0	3(1)	36.7	120	21.3	0
W-40-35	2(1)	782.4	1092	23	0	1(1)	177.7	1092	23	0	2(1)	105.9	1092	23	0
W-40-40	1(1)	1010.1	3279	24	0	0(1)	239.9	3279	25	0	1(1)	138	1092	24	0
W-40-45	1(0)	1192.7	0	25	0	0(0)	-	-	-	0	1(0)	236.5	0	25	0
W-50-10	5	0.7	0	8.4	1	4(1)	0.6	3	8.4	1	4(1)	0.3	3	8.4	1
W-50-15	2(3)	1.2	39	12	0.4	3(2)	0.5	12	12.2	0.2	2(3)	0.6	12	11.8	0.2
W-50-20	4(1)	1.1	12	14.8	0.2	4(1)	1.7	12	15.6	0.2	3(2)	1.1	7.5	15	0.2
W-50-25	4(1)	52.5	120	18.2	0	5	5.8	0	19.3	0	3(2)	1.9	12	17.8	0
W-50-30	4(1)	149.6	363	20.4	0	5	15.1	0	19.8	0	4(1)	8.4	39	19.8	0
W-50-35	2(2)	162.4	727.5	22.8	0	3(0)	28.3	0	22	0	3(2)	17.7	25.5	21.6	0
W-50-40	1(3)	239.6	849	23.8	0	1(3)	178.7	2550	25	0	2(2)	128.5	79.5	22.8	0
W-50-45	2(2)	262.1	1092	24.8	0	2(1)	233.8	3279	23.5	0	2(2)	143.5	120	24.5	0

TABLE 3.5. Special parts

Name	Reduced skills					Basic configurations					All skills				
	Opt	CPU	Nodes	Tasks	Idle	Opt	CPU	Nodes	Tasks	Idle	Opt	CPU	Nodes	Tasks	Idle
N-40-10	3(2)	0.5	7.5	10	0.8	5	0.4	0	9.8	1	0(5)	0.8	13.8	10	0.8
N-40-15	5	0.8	0	14.8	0.2	2(3)	1.1	93	14	0.2	0(5)	27.7	42.6	15	0
N-40-20	4(1)	0.6	3	17	0	3(2)	3.4	120	17	0	0(5)	247.5	325.2	19	0
N-40-25	4(1)	0.6	3	20	0	2(3)	44.8	282	19.6	0	0(4)	2379.4	403.5	24.6	0
N-40-30	4(1)	1.3	3	21	0	2(1)	69.4	363	20.7	0	0(3)	2916.1	606	29.3	0
N-40-35	4(1)	4.4	12	23	0	0(2)	241.4	2185.5	23	0	0(0)	-	-	-	-
N-40-40	4(1)	12.9	39	23	0	1(1)	280.1	3279	23.3	0	0(0)	-	-	-	-
N-40-45	3(2)	50.2	79.5	23.4	0	0(0)	-	-	-	3	0(0)	-	-	-	-
N-50-10	3(2)	0.3	12	9.2	0.6	4(1)	0.7	3	8.4	1	2(3)	0.6	9	10	0.8
N-50-15	5	0.4	0	13.2	0.2	4(1)	0.7	12	12	0.2	0(5)	3.9	10.2	15	0
N-50-20	5	0.6	0	15.4	0	5	0.3	0	15.6	0.2	1(4)	35.8	43.5	19.2	0
N-50-25	5	0.5	0	18.2	0	4(1)	0.4	39	18.6	0	1(4)	132.8	133.5	25	0
N-50-30	4(1)	1	3	19	0	4(1)	4.6	39	19.7	0	0(1)	322.1	363	26	0
N-50-35	4(1)	4.2	12	21	0	2(2)	22.9	241.5	20.5	0	0(1)	941.3	1092	27	0
N-50-40	2(3)	2	21	23	0	3(1)	30.4	363	22.5	0	0(0)	-	-	-	-
N-50-45	2(3)	2.8	30	23	0	1(3)	67.5	606	23.5	0	0(0)	-	-	-	-
W-40-10	5	0.5	0	10	0.8	4(1)	0.3	12	10	1	1(4)	1.5	3	10	0.8
W-40-15	3(2)	0.5	12	14.4	0	2(3)	1	12	14	0.4	0(5)	13.1	17.4	15	0
W-40-20	5	0.5	0	18.2	0	4(1)	13.8	39	17	0	0(3)	2170.3	525	20	0
W-40-25	2(3)	0.6	18	21	0	1(4)	91.8	909.8	21	0	0(1)	6485.7	1092	25	0
W-40-30	4(1)	2.1	39	22.2	0	1(1)	25.2	120	22	0	0(1)	5700.3	1092	27	0
W-40-35	4(1)	5.1	39	23.4	0	1(1)	177.7	1092	23	0	0(0)	-	-	-	-
W-40-40	5	26.7	0	23.6	0	0(1)	239.9	3279	25	0	0(0)	-	-	-	-
W-40-45	0(5)	70.5	109.2	23.8	0	0(0)	-	-	-	0	0(0)	-	-	-	-
W-50-10	4(1)	0.3	3	9.5	0.8	4(1)	0.6	3	8.4	1	2(3)	0.5	6	10	0.8
W-50-15	4(1)	0.4	3	14	0	3(2)	0.5	12	12.2	0.2	0(5)	23.8	21	15	0
W-50-20	5	0.5	0	15.2	0	4(1)	1.7	12	15.6	0.2	0(5)	120.8	33.6	20	0
W-50-25	5	0.6	0	17.8	0	5	5.8	0	19.3	0	0(3)	298.2	66	24.6	0
W-50-30	5	1.4	0	19.2	0	5	15.1	0	19.8	0	0(0)	-	-	-	-
W-50-35	4(1)	3.5	3	20.6	0	3(0)	28.3	0	22	0	0(0)	-	-	-	-
W-50-40	4(1)	4.3	12	21	0	1(3)	178.7	2550	25	0	0(0)	-	-	-	-
W-50-45	5	17.6	0	23.2	0	2(1)	233.8	3279	23.5	0	0(0)	-	-	-	-

TABLE 3.6. Skills

Name	Basic configurations (30-45 min)					Reduced service times (15-30 min)					Reduced service times (10-20 min)				
	Opt	CPU	Nodes	Tasks	Idle	Opt	CPU	Nodes	Tasks	Idle	Opt	CPU	Nodes	Tasks	Idle
N-40-10	5	0.4	0	9.8	1	4(1)	0.4	3	9.8	1	4(1)	0.5	3	9.8	1
N-40-15	2(3)	1.1	93	14	0.2	1(4)	2.1	9.8	13.8	0.2	1(4)	2.8	9.8	13.8	0.2
N-40-20	3(2)	3.4	120	17	0	3(2)	17.7	12	17	0	3(2)	22.4	25.5	17	0
N-40-25	2(3)	44.8	282	19.6	0	2(3)	298.5	174	20.2	0	2(2)	354.8	241.5	20.3	0
N-40-30	2(1)	69.4	363	20.7	0	3(1)	343.8	363	21.8	0	2(2)	393.8	363	21.5	0
N-40-35	0(2)	241.4	2185.5	23	0	1(2)	2356.6	3279	22.3	0	0(2)	2430.5	2185.5	23	0
N-40-40	1(1)	280.1	3279	23.3	0	2(0)	3309.2	0	25.5	0	0(1)	3707.3	3279	24	0
N-40-45	0(0)	-	-	-	3	0(1)	2751.2	3279	25	0	0(1)	4285.9	3279	24	0
N-50-10	4(1)	0.7	3	8.4	1	5	0.5	0	8.4	1	5	0.6	0	8.4	1
N-50-15	4(1)	0.7	12	12	0.2	3(2)	0.9	3	11.8	0.4	3(2)	0.4	7.5	11.8	0.4
N-50-20	5	0.3	0	15.6	0.2	4(1)	1.6	12	14.8	0.2	4(1)	1.1	12	14.8	0.2
N-50-25	4(1)	0.4	39	18.6	0	3(2)	5.2	7.5	17.8	0	3(2)	22.8	25.5	17.8	0
N-50-30	4(1)	4.6	39	19.7	0	4(1)	5.6	12	19.8	0	3(2)	34.4	39	20.2	0
N-50-35	2(2)	22.9	241.5	20.5	0	3(2)	162.5	79.5	20.8	0	3(2)	250.5	241.5	23.2	0
N-50-40	3(1)	30.4	363	22.5	0	3(2)	430.6	120	23	0	0(4)	487	363	27.5	0
N-50-45	1(3)	67.5	606	23.5	0	2(2)	453	241.5	23.8	0	0(4)	516.2	1213.5	25	0
W-40-10	4(1)	0.3	12	10	1	5	0.4	0	9.8	1	5	0.5	0	9.8	1
W-40-15	2(3)	1	12	14	0.4	1(4)	1.8	12	13.8	0.4	1(4)	1.7	9.8	13.8	0.4
W-40-20	4(1)	13.8	39	17	0	3(2)	35.2	39	17.4	0	3(2)	39.6	39	17.4	0
W-40-25	1(4)	91.8	909.8	21	0	2(1)	138.6	120	21	0	3(1)	153.7	120	21	0
W-40-30	1(1)	25.2	120	22	0	2(2)	381.8	241.5	22	0	3(2)	845.5	363	21.8	0
W-40-35	1(1)	177.7	1092	23	0	1(1)	501.5	363	24.5	0	0(2)	902.3	727.5	23	0
W-40-40	0(1)	239.9	3279	25	0	0(1)	482.6	363	25	0	1(0)	925.9	0	25	0
W-40-45	0(0)	-	-	-	-	0(0)	-	-	-	-	0(0)	-	-		-
W-50-10	4(1)	0.6	3	8.4	1	4(1)	0.4	3	8.4	1	4(1)	0.5	3	8.4	1
W-50-15	3(2)	0.5	12	12.2	0.2	3(2)	0.7	7.5	12.2	0.4	3(2)	0.6	12	12.2	0.4
W-50-20	4(1)	1.7	12	15.6	0.2	5	2	0	14.6	0.2	5	1.7	0	14.6	0.2
W-50-25	5	5.8	0	19.3	0	5	15.6	0	18	0	5	17.8	0	18	0
W-50-30	5	15.1	0	19.8	0	5	17.6	0	20.2	0	4(1)	22.9	39	20.2	0
W-50-35	3(0)	28.3	0	22	0	3(1)	28.6	39	22.8	0	3(1)	36	39	23	0
W-50-40	1(3)	178.7	2550	25	0	0(4)	188.6	99.8	23.8	0	1(3)	415.7	282	24	0
W-50-45	2(1)	233.8	3279	23.5	0	3(1)	246.5	120	25	0	0(4)	477.3	302.3	25.8	0

TABLE 3.7. Service times

Name	Basic configurations (3 technicians)					More technicians (4 technicians)				
	Opt	CPU	Nodes	Tasks	Idle	Opt	CPU	Nodes	Tasks	Idle
N-40-10	5	0.4	0	9.8	1	3(2)	0.7	3	10	1.8
N-40-15	2(3)	1.1	93	14	0.2	0(5)	2.3	10.2	15	0.4
N-40-20	3(2)	3.4	120	17	0	1(4)	53.2	16.5	20	0
N-40-25	2(3)	44.8	282	19.6	0	2(2)	36.5	39	24.3	0
N-40-30	2(1)	69.4	363	20.7	0	1(3)	257.5	93	27.3	0
N-40-35	0(2)	241.4	2185.5	23	0	2(1)	1213.1	1092	31.3	0
N-40-40	1(1)	280.1	3279	23.3	0	0(1)	1158.4	1092	35	0
N-40-45	0(0)	-	-	-	-	0(0)	-	-	-	-
N-50-10	4(1)	0.7	3	8.4	1	2(3)	0.7	18	10	1.6
N-50-15	4(1)	0.7	12	12	0.2	4(1)	2.4	12	14.2	0.4
N-50-20	5	0.3	0	15.6	0.2	2(3)	7	21	18	0
N-50-25	4(1)	0.4	39	18.6	0	3(2)	44.8	39	22.8	0
N-50-30	4(1)	4.6	39	19.7	0	1(4)	62.6	79.5	25.2	0
N-50-35	2(2)	22.9	241.5	20.5	0	1(3)	989.1	606	27	0
N-50-40	3(1)	30.4	363	22.5	0	1(2)	802.1	727.5	27.3	0
N-50-45	1(3)	67.5	606	23.5	0	0(1)	891.5	1092	29	0
W-40-10	4(1)	0.3	12	10	1	4(1)	0.4	3	10	2
W-40-15	2(3)	1	12	14	0.4	1(4)	3.8	16.5	15	0.6
W-40-20	4(1)	13.8	39	17	0	0(4)	15.3	39	20	0
W-40-25	1(4)	91.8	909.8	21	0	1(1)	98.8	120	24	0
W-40-30	1(1)	25.2	120	22	0	2(1)	3722.5	1092	27.3	0
W-40-35	1(1)	177.7	1092	23	0	1(1)	5202.1	3279	29	0
W-40-40	0(1)	239.9	3279	25	0	0(0)	-	-	-	-
W-40-45	0(0)	-	-	-	-	0(0)	-	-	-	-
W-50-10	4(1)	0.6	3	8.4	1	0(5)	0.6	4.8	9.6	1.2
W-50-15	3(2)	0.5	12	12.2	0.2	3(2)	1.8	12	14.4	0.6
W-50-20	4(1)	1.7	12	15.6	0.2	1(3)	27.8	30	18.3	0
W-50-25	5	5.8	0	19.3	0	2(2)	317.1	241.5	21.8	0
W-50-30	5	15.1	0	19.8	0	1(2)	508.5	363	24.3	0
W-50-35	3(0)	28.3	0	22	0	1(1)	4663.9	3279	25.5	0
W-50-40	1(3)	178.7	2550	25	0	0(1)	1353.3	3279	27	0
W-50-45	2(1)	233.8	3279	23.5	0	0(0)	-	-		-

TABLE 3.8. Number of technicians

3.5.3. Comparison with CPLEX

Branch-and-price has allowed us to solve larger instances than CPLEX, when applied to the arc-based MIP formulation in [45], as shown in Table 3.9. Since the MIP model penalizes skipped breaks, but does not force technicians to take them, an alternative version of our branch-and-price algorithm was developed where breaks were not enforced when solving the pricing problem (which led to a more difficult pricing problem). The results show a huge difference between the branch-and-price algorithm and CPLEX with regard to the number of instances solved to optimality and computation times. In particular, CPLEX can only solve instances with up to 15 tasks. We recall that our branch-and-price algorithm was able to solve instances with up to 45 tasks (for the variant where technicians are forced to take their breaks).

Name	**CPLEX**		**Branch-and-Price**	
	Opt	CPU	Opt	CPU
N-40-10	5	1073	5	1.2
N-40-15	0	-	5	24.6
N-50-10	5	732	5	0.6
N-50-15	0	-	5	0.7
W-40-10	5	1593	5	2.4
W-40-15	0	-	5	23.2
W-50-10	5	655	5	0.6
W-50-15	2	24752	5	2.0

TABLE 3.9. CPLEX vs Branch-and-Price

3.6. CONCLUSION

In this paper, a technician routing and scheduling problem was tackled with a branch-and-price algorithm, using two different branching strategies. Based on the test instances in [45], we first evaluated the benefits provided by DSSR over the standard pricing problem, using both branching strategies. The DSSR implementation with ternary branching proved to be the best. Then, we considered the impact of different problem characteristics on the performance of our algorithm. Finally, a comparison with an arc-based MIP formulation solved with CPLEX showed that branch-and-price is largely superior and can solve larger instances. Our research will now focus on developing metaheuristics and matheuristics for the problem. Furthermore, we want to consider a variant where new tasks occur dynamically and must be integrated in real-time into the current solution.

Chapitre 4

UNE MÉTAHEURISTIQUE BASÉE SUR LA RECHERCHE TABOU POUR RÉSOUDRE UN PROBLÈME DE TOURNÉES DE TECHNICIENS

Ce chapitre correspond à un article qui a été soumis pour publication dans la revue *Computers and Operations Research*. Les co-auteurs sont I. Mathlouthi, M. Gendreau et J.-Y. Potvin.

Ce chapitre s'intéresse à un problème de tournées de techniciens motivé par une application de réparation et de maintenance des équipements de paiements électroniques. Ce problème présente différentes particularités telles que les fenêtres de temps multiples pour le service et l'inventaire des pièces de rechange pour chaque technicien. Les tâches ont parfois besoin d'une pièce de rechange spéciale. Une méthode de résolution basée sur la recherche tabou, couplée avec une mémoire adaptative, est présentée. L'inclusion des solutions (minima locaux) dans la mémoire adaptative prend en considération la qualité et la diversité de cette mémoire. On rapporte des résultats sur des instances allant jusqu'à 200 tâches. Une comparaison avec la méthode de branch-and-price du chapitre 3 est également présentée sur des instances de plus petite taille.

A Metaheuristic based on Tabu Search for solving a Technician Routing and Scheduling problem

This paper addresses a technician routing and scheduling problem motivated by an application for the repair and maintenance of electronic transactions equipments. The problem exhibits many special features like multiple time windows for service, an inventory of spare parts carried by each technician and tasks that may require a special part to be performed. A problem-solving methodology based on tabu search, coupled with an adaptive memory, is proposed. The inclusion of solutions (local minima) in the adaptive memory takes into account both solution quality and memory diversity. Results are reported on test instances with up to 200 tasks. A comparison with a previously developed branch-and-price algorithm is also reported on instances of small size.

Keywords : Technician routing and scheduling problem, tabu search, adaptive memory, biased fitness.

4.1. Introduction

This paper considers a Technician Routing and Scheduling Problem (TRSP) motivated by a real application for the repair or maintenance of electronic transactions equipments by a number of technicians. The problem is to assign a subset of tasks to each technician and to construct a route over each subset to optimize some objective, which may involve many criteria (like total traveled distance, overtime, etc.). The routes must also satisfy different types of constraints. First, there are compatibility constraints between tasks and technicians, since different skills are required to perform different tasks and a technician does not necessarily possess all those skills. Second, a task may also need a number of spare parts. If the technician does not have the required parts in his initial inventory when he starts his route, he can acquire them by visiting a depot at some point along the route. Typically, an infinite number of parts is assumed at the depot, although only a finite number can be carried by the technician. Third, a task may require a special part which can only be obtained by visiting the depot (i.e., a technician is not allowed to carry a special part from or to his home base location). Fourth, there are time bounds for the service of each task and for the return time of each technician at his home base location.

It is also assumed that not all tasks can be served by the technicians. Thus, a gain is associated with each task and the maximization of the total gain collected along the routes becomes part of the objective. The remaining tasks can then be considered for another day, with typically an increased gain (to avoid being left apart repeatedly). Hence, the gain is related to characteristics of the task or the customer who requests the service.

The remainder of the paper is organized as follows. In Section 2, we provide a review of various works on problems related to ours. In Section 3, we describe our problem. Then, after an introduction to recent ideas that have been integrated into tabu search to improve its performance, we detail our algorithm in Section 4. The test instances are introduced and are followed by computational results in Section 5. In particular, we provide a comparison with optimal solutions previously obtained with an exact method on instances of small sizes. Finally, a conclusion follows in Section 6.

4.2. Literature review

Most papers on the TRSP are based on real applications that exhibit specific features. To the best of our knowledge, the first work on this type of problem was reported in 1997 by Tsang and Voudouris for a telecommunications application [67]. To solve this problem, the authors used different types of local search heuristics. Weigel and Coo [71] then introduced a problem faced by a large retailer when providing on-site technical assistance. The authors solved the problem by developing a heuristic procedure to construct a route for each technician and by improving each route individually with Or-opt exchanges [48]. In [5], the authors addressed a periodic maintenance problem for elevators and escalators. In this application, technician routes had to account for working regulations. A similar application was later addressed by Tang et al. [64] with a tabu search heuristic. In 2007, the French Operations Research Society (ROADEF) initiated a challenge based on a problem encountered by France Telecom. This problem first involves a multi-period horizon. Also, teams of technicians must be created because each task needs multiple skills with different proficiency levels. For this challenge, Hashimioto et al. [33] developed a greedy randomized adaptive search (GRASP) heuristic while Cordeau et al. [11] proposed and adaptive large neighborhood search (ALNS). Another multi-period TRSP for the maintenance of electric forklifts, where technicians are paired to form teams, is solved with a branch-and-price algorithm in Zamorano and Stolletz [75].

A parallel matheuristic is proposed in [51] for a TRSP where tools and spare parts are taken into account. The matheuristic is made of a constructive heuristic, a parallel ALNS and a mathematical programming-based post-optimization procedure. In Mendoza et al. [46] a technician routing problem with conventional and electric vehicles is introduced. Due to their relatively limited driving range, the electric vehicles need to visit one or more recharging stops along their route. The problem is also addressed with a parallel matheuristic, where a number of subproblems are first created and solved with GRASP. The routes of all local minima produced by GRASP are collected to create a repository of routes. Then, a set covering model is solved over these routes.

In [7], the authors introduced a problem faced by a water treatment and distribution company. Here, the technician routes must be planned over a period of one week for repair

or maintenance. The tasks to be scheduled can either be known in advance or can occur dynamically. A memetic algorithm is used to solve the problem, which is first applied on the static tasks to produce initial tours for every day of the week. Then, the dynamic tasks are integrated into the current routes as they occur, still with the memetic algorithm. An exact approach based on column generation is also proposed in this work, but can only be applied to instances of small size. Pillac et al. [50] also addressed a TRSP in which a fraction of the tasks occur dynamically. A parallel architecture is proposed to speed up the calculations. An initial solution is first created with the static tasks using a construction heuristic based on a regret measure [54]. This solution is then improved with ALNS [53]. The latter algorithm works by successively destroying (removing tasks) and repairing (reinserting tasks) the current solution to produce a new solution. For dynamic tasks, the part of the current solution already executed is fixed and the newly occurring tasks are incorporated into the solution by running the ALNS for a limited number of iterations. The same authors also proposed a fast reoptimization approach based on a parallel ALNS in [52].

In [45], a mixed integer programming (MIP) model was proposed for our TRSP. Although the model is useful by providing a formal description of the problem, solving the model exactly with a commercial solver proved to be impractical, except for very small instances with no more than 15 tasks. In a later work [44], a branch-and-price algorithm was developed and was able to solve exactly instances with up to 45 tasks. Given the limitations of exact approaches, we now propose in this paper a tabu search heuristic to address instances of larger sizes.

4.3. Problem Definition

We start this section by describing the main entities, with their attributes, involved in our problem (for a formal MIP formulation, see [45]). They are :

- Tasks
  - Skills : each task requires one or more skills from a technician to be performed.
  - Parts : each task requires one or more spare parts of different types and, possibly, a special part to be performed.
  - Gain : a gain is associated with each task that represents its importance (based on service priority, revenue, customer status, etc.).
  - Multiple time windows : a technician must begin his service within one of multiple time windows associated with task i, where a time window is defined by a lower bound e_i and an upper bound l_i. If the technician arrives before the lower bound, he can wait and start the service at the lower bound.
- Technicians
  - Home base : starting and ending location of the technician's route.

- Skills : each technician has one or more skills that allow him to serve certain tasks and not others.
- Depot : each technician is a priori assigned to a single depot (among a number of possible depots) where he can replenish his inventory of parts.
- Workday : each technician normally works between 9 :00 AM and 5 :00 PM, although overtime is allowed (at the expense of a penalty in the objective). Three breaks can be taken during the day : two breaks of 15 minutes in the morning and afternoon and a mid-day break of 30 minutes. Each break must be taken within a specific time window and a technician must take the break if his schedule intersects with that time window. Furthermore, the route of a technician cannot exceed a maximum traveled distance.
- Inventory : each technician carries an initial inventory of spare parts when he leaves his home base location. If there are not enough spare parts to serve all tasks along a technician's planned route, or if one or more tasks require a special part, then a single visit to a preassigned depot is allowed along the route.

— Parts
- Spare parts : there are different types of spare parts. Although the depot contains a virtually infinite number of spare parts, the technician can only carry a limited number of parts of each type.
- Special parts : special parts are also available at the depot. A technician must visit the depot when one or more tasks along his route require a special part. No special part can be carried to or from the home base location.

— Depots
- There are a fixed number of depots with a virtually infinite number of spare parts.

The problem is to design a route over a subset of tasks for each of a fixed number of technicians while satisfying the hard constraints mentioned above, namely, the required skills and required parts to perform a task, the multiple time windows for the service of a task and the maximum traveled distance of each route. Given that not all tasks can be served, the objective is to maximize the total gain collected, minus the total traveled distance and total overtime over all routes (or, equivalently, to minimize the sum of total traveled distance and total overtime minus the total gain collected). Each component in the objective has a weighting parameter to adjust its importance when evaluating a solution.

4.4. Problem-solving methodology

Our problem-solving methodology is based on tabu search [31] which has proven successful for a variety of hard combinatorial problems, like vehicle routing [28, 29], job shop scheduling [39], quadratic assignment [62], technician routing and scheduling [64], and many others. Modern implementations involve the integration of adaptive memories [59] made of

elite solutions or components of elite solutions in order to perform search intensification or search diversification. An example for a vehicle routing problem can be found in [63] where the routes of elite solutions are stored in memory and used to construct new starting solutions for the tabu search. Basically, a new solution is obtained by mixing routes from different elite solutions in memory. In our tabu search, we manage the adaptive memory as a true population of solutions with considerations for both quality and diversity. This approach has proven very successful in the hybrid genetic algorithm of Vidal et al. [70] when solving different types of vehicle routing problems.

The proposed metaheuristic is shown in Algorithm 1, where s stands for the current solution and s^* for the best solution. It starts with a number of calls to the function Greedy(). The latter randomly applies one of two greedy construction heuristics to create an initial solution which is then improved with a local search descent. The resulting solutions are then filtered out to remove duplicates and are stored in adaptive memory. The best solution in adaptive memory is then used as the initial starting solution. The outer **while** loop is aimed at stopping the algorithm and returning the best solution s^* when a maximum number of iterations or maximum computation time has been reached. Each iteration of the inner **while** loop involves the application of four consecutive tabu searches, each with a different neighborhood structure. The best solution found by a tabu search with a given neighborhood structure is returned and fed to the next one. Each tabu search also takes care to update s^* during its execution, if required. The stopping criterion for any given tabu search is based on a maximum number of iterations. Note also that every local minimum reached by a tabu search is stored in adaptive memory. The inner **while** loop is repeated until a complete pass through the four neighborhoods does not provide any improvement. At this point, the adaptive memory is filtered out to remove duplicates and is possibly reduced if the maximum memory size n_{max} is exceeded. Then, a new starting solution is created by combining routes of different solutions in adaptive memory (see section 4.4.3.3). In the following, the main components of this search framework are described.

4.4.1. Construction heuristics

The adaptive memory is initialized with n_{min} different solutions created with two randomized construction heuristics, each contributing to half of the solutions. These solutions are then improved with a local descent based on the same neighborhood structures than the ones used in the tabu search (see section 4.4.2). In our experiments, n_{min} was set to ten times the number of technicians. The two construction heuristics are the following.

Sequential construction heuristic. In this heuristic, the routes of the technicians are constructed one by one. That is, as long as technicians are available, one of them is randomly selected and his route is initially created with a starting and ending location (home base location of the technician) and the three breaks. Then, a task is randomly selected among those that

Algorithm 1

```
1: procedure SEARCH
2:     for nInit iterations do
3:         s_init ← Greedy ()
4:         s_init ← Local Search (s_init)
5:         Store s_init in adaptive memory
6:     s* ← select best solution in adaptive memory
7:     s ← s*
8:     while number of iterations < nIter_max  or  time < T_max do
9:         while s is improved do
10:            s ← Tabu (s, Inter-Route Move)
11:            s ← Tabu (s, Intra-Route Move)
12:            s ← Tabu (s, Swap)
13:            s ← Tabu (s, Swap-With-New)
14:        if number of solutions in adaptive memory > n_max then
15:            Update memory
16:        s ← Create starting solution from adaptive memory
       return s*
```

can be performed by the technician. The three best feasible insertions in the current route of the technician are considered and one of them is randomly selected. This is repeated until no new task can be added to the route. In case the selected insertion requires a visit to the depot, the three best feasible insertions for the depot are kept by creating three different routes. The insertion procedure then follows with the three routes and the best route is selected at the end. By keeping a number of different routes, each associated with a different insertion place for the depot, the myopic behavior of the construction heuristic is alleviated.

Parallel construction heuristic. This heuristic constructs routes for all technicians in parallel. The route of each technician is first initialized with his starting and ending locations and the three breaks. Then, a task is randomly selected and the best feasible insertion in the route of each technician who can perform it is considered. The task is assigned to the best technician and inserted in his route. This is repeated until no more tasks can be feasibly inserted in the routes. During this insertion procedure, the depot is handled as in the sequential heuristic.

4.4.2. Tabu search

The tabu search improves the starting solutions obtained from the adaptive memory. Its solution space and neighborhood structures are now described.

4.4.2.1. *Solution space*

In the following, a solution is considered feasible (and is part of the solution space) even if the maximum traveled distance constraint is not satisfied. In such a case, the solution is immediately repaired. The repair procedure is applied in turn to each route that exceeds the

maximum distance constraint. This procedure is very simple and removes, at each iteration, the task with the least impact on the objective value until the maximum distance constraint is satisfied again.

4.4.2.2. *Neighborhoods*

The tabu search exploits a sequence of four different neighborhood structures. More precisely :

- *Inter-Route Move*. Each task is moved from one route to another and inserted at its best feasible insertion place.
- *Intra-Route Move*. Each task is removed from its current position and inserted to its best feasible insertion place (other than the current one) in the same route.
- *Swap*.Every pair of tasks from two different routes are swapped. Each task is inserted at the best feasible insertion place in its new route.
- *Swap-With-New*. Each task not currently in the solution is swapped with a task in the solution and is inserted at the best feasible insertion place.

After a move, each task not currently in the solution is considered for insertion in the routes that have been modified (in non increasing order of gain). During the exploration of each neighborhood, a first improvement strategy is implemented. Thus, the first feasible neighbor that provides an improvement over the current solution is chosen. Tabu restrictions are also imposed on each neighborhood. That is, tasks involved in recent moves are declared tabu for θ iterations (unless they can improve the best known solution), where θ is randomly selected in the interval $[\theta_{min}, \theta_{max}]$, with $\theta_{min} = 5$ and $\theta_{max}] = 10$ in our experiments.

It should be noted that the algorithm loops through the four neighborhoods (in the order indicated above) as long as the solution improves. If a pass through the four neighborhoods does not provide any improvement, then a new initial solution is obtained from the adaptive memory to restart the search.

4.4.3. Adaptive Memory

Here, we describe the procedures used to store and fetch solutions from the adaptive memory. Since every solution decomposes into a route per technician, is should be noted that the memory is implicitly divided into a number of smaller memories, where each memory contains the routes of a given technician (see Figure 4.1). This partition is required during the fetching procedure because each technician has specific routes that start from and end at a different home base location and visit a particular preassigned depot. In the following, we first explain how the biased fitness of a solution in adaptive memory is computed, before describing the storing and fetching mechanisms.

4.4.3.1. *Biased fitness*

The quality of a solution in memory corresponds, on the one hand, to its objective value and, on the other hand, to its contribution to the diversity of solutions in that memory [**70**]. The diversity computation is based on the Hamming distance between two solutions s_1 and s_2, see Equation (4.4.1). In this equation, $\Gamma(x)$ is an indicator function : it returns 1 if x is true, 0 otherwise ; $T(i,s)$ returns the technician who performs task i in solution s, while $W(i,s)$ returns the index of the time window for the service of task i in solution s (since there are multiple time windows). When task i is not served by any technician, a dummy value is returned.

$$\delta^H(s_1,s_2) = \sum_{i \in s_1 \cup s_2} \Gamma(T(i,s_1) \neq T(i,s_2)) + \Gamma(W(i,s_1) \neq W(i,s_2)) \tag{4.4.1}$$

The diversity contribution of solution s is then computed as follows :

$$\Delta(s) = \frac{1}{n_c} \sum_{s' \in N_c} \delta^H(s,s') \tag{4.4.2}$$

Thus, it is the average of the Hamming distances between s and the $n_c = |N_c|$ closest solutions in adaptive memory, where n_c is a parameter (set to 20% of the solutions in adaptive memory, as in [**70**]). Then, the ranks $r_f(s)$ and $r_d(s)$ of each solution s in memory with regard to the objective value and diversity contribution, respectively, are computed (where rank 1 is best). With these ranks, the biased fitness BF of solution s is computed as follows, where n_m is the current number of solutions in memory :

$$BF(s) = (n_m - r_f(s) + 1) + \eta(n_m - r_d(s) + 1) \tag{4.4.3}$$

.

In this formula, the weight of the objective value component (first term in the sum) is implicitly set to 1 to guarantee a minimum contribution of this component to the biased fitness. Then, a more or less important bias towards diversity is added depending on the value of parameter $\eta \in [0, 1]$.

4.4.3.2. *Storing and updating*

Every local minimum visited by the tabu search is added to the adaptive memory. After one pass through the four neighborhoods (see Algorithm 1), the memory is updated as follows :

— duplicates (i.e., solutions with the same objective value) are removed ;

— the biased fitness of each solution is computed ;

— if the maximum memory size n_{max} is exceeded, the n_{max} solutions with the best biased fitness are kept. In our experiments, this parameter was set to one hundred times the number of technicians.

4.4.3.3. *Fetching*

As previously mentioned, each solution is decomposed into a route for each technician, where each route is stored in the memory of the corresponding technician. In this process, the route also inherits the biased fitness value of the whole solution.

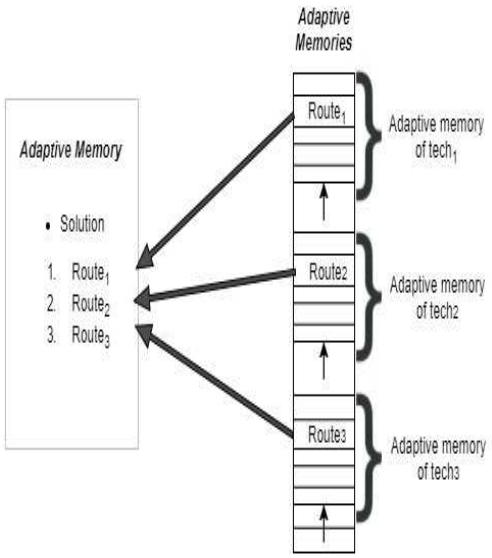

FIGURE 4.1. Adaptive memories

The procedure to create a starting solution for the tabu search is the following (see line 16 of Algorithm 1). We proceed technician by technician. A first technician is considered and each route in the memory of this technician is assigned a probability of selection which is proportional to its biased fitness (roulette wheel). A route r is probabilistically chosen and included in the starting solution. Then, the memory of all other technicians is updated by removing any route with a task in common with route r. This process of (1) considering the next technician, (2) probabilistically selecting a route of that technician, (3) adding this route to the starting solution and (4) updating the memory of all remaining technicians, is repeated until all technicians are done (a complete starting solution is obtained) or until the remaining technicians have no valid routes in memory. In the latter case, the solution is completed by inserting additional tasks with the parallel construction heuristic of section 4.4.1.

4.5. Computational experiments

In this section, the test instances for the experiments are first described. Then, we report the results obtained with three different variants of our metaheuristic. We also compare the solutions produced by our algorithm with the optimum on instances of small size.

4.5.1. Test instances

The test instances come from [45], where the crow fly distance is assumed between each pair of locations and the speed of the vehicles is set at 50km/h. The other characteristics are the following.

- *Service area*. The service area is a 40 km × 40 km or 50 km × 50 km squared area.
- *Depot Location*. There are 3 depots randomly located within the service area.
- *Task location*. Each task is randomly located within the service area.
- *Task service time*. The service time of each task is randomly chosen between 30 and 45 minutes.
- *Task gain*. The gain associated with a task is randomly chosen between 1 and 10.
- *Technician home base*. To get a good coverage of the tasks, the first two technicians are located at the opposite ends of the service area (along the diagonal). The other technicians are randomly located within the service area.
- *Technician skills*. With each technician is associated the percentage of tasks that he can perform : one third of the technicians can perform all tasks, one third of the technicians can perform 50% of the tasks and one third 25% of the tasks.
- *Parts*. The number of spare parts needed to perform a task is randomly chosen between 0 and 3. Then, each spare part is assigned a type, among 4 different types. Also, a special part is needed with a probability of 0.125.
- *Time windows*. Both narrow and wide time windows are considered. The latter are twice as wide as the former on average. The length of a narrow time window is randomly generated between 60 and 90 minutes. The lower bound of the first time window is chosen randomly between 9 :00 AM and noon. The lower bounds of the remaining time windows are set between 2 and 3 hours after the upper bound of the previous time window until a maximum of 3 time windows are obtained.
- *Maximum distance*. The maximum distance traveled by each technician during his workday is set to 125 km.

Different subsets of instances were obtained by varying the values of the following characteristics : size of the service area (either 40km × 40km or 50km × 50km), width of the time windows (either narrow or wide) and number of tasks (50, 100 or 200). Furthermore, tests were performed with 3 and 6 technicians for instances with 50 tasks, 6 and 12 technicians for instances with 100 tasks, and 12 and 24 technicians for instances with 200 tasks. Thus, we

have $2 * 2 * 3 * 2 = 24$ subsets of instances, with 10 instances in each subset. In the tables of results, the subsets are identified with the following fields : width of the time windows (either N for narrow or W for wide), size of the service area (either 40 or 50), number of tasks and number of technicians. For example N-40-50-3 is the subset of instances with narrow time windows, a 40 km $\times$ 40 km service area, 50 tasks and 3 technicians.

Finally, the weights in the objective function were set as follows : 500 for the gain, 5 for the distance (in kilometers) and 1 for overtime (in seconds). Overall, more emphasis is given to the total gain over the total distance and overtime. This setting allows a technician to do some overtime to perform a high gain task and is more challenging than the setting proposed in [45].

4.5.2. Results

Here, we report the results of different experiments on a 3.07GHz Intel Xeon X5675 processor, using the test instances introduced above. Our problem-solving methodology was run for $T_{max} = 1$ hour on the instances of size 50 and 100, which was enough to allow convergence. For the instances of size 200, a computation time of $T_{max} = 3$ hours was required. Based on preliminary experiments, the number of iterations with each neighborhood was set as follows : 75 for Inter-Route Move, 20 for Intra-Route Move, 100 for Swap and 100 for Swap-With-New. These values were determined after some preliminary experiments..

4.5.2.1. *Impact of parameter η*

In this section, we explore the impact of parameter η on the performance of our metaheuristic. This parameter controls the magnitude of the diversity contribution when a solution is evaluated in adaptive memory (biased fitness). When $\eta = 0$, only the objective value of a solution is considered. Conversely, when $\eta = 1$, the objective value and the diversity contribution have the same weight. Six different values for η between 0 and 1 were tested. For each subset of instances and each η value, Table 4.1 reports the average gap in percentage with the best solution produced on each instance over the six η values. The best gap for each subset is shown in bold. We can see that including diversity has a positive impact, since the worst gaps are clearly obtained with $\eta = 0$. When considering all test instances, the best η value is equal to 0.6 with an average gap of 1.87%, as indicated in the line Overall of Table 4.1. Thus, this value will be used in the following experiments.

4.5.2.2. *Comparison with three other variants*

In this section, we compare our algorithm, called *TS-AM-I*, with three other variants :

Name	Gap (%)					
	$\eta = 1$	$\eta = 0.8$	$\eta = 0.6$	$\eta = 0.4$	$\eta = 0.2$	$\eta = 0$
N-40-50-3	1.90	**1.85**	2.25	2.73	1.93	3.79
N-40-50-6	3.48	2.94	**1.91**	3.47	5.41	5.06
N-50-50-3	3.61	2.15	3.41	**2.11**	5.36	6.48
N-50-50-6	2.61	**1.89**	1.92	3.20	2.53	3.15
W-40-50-3	**1.70**	2.88	1.84	2.41	2.41	4.49
W-40-50-6	3.22	3.99	**3.36**	3.99	4.04	4.99
W-50-50-3	1.42	2.77	**1.39**	2.13	2.45	3.52
W-50-50-6	2.96	**1.61**	1.70	2.75	3.50	5.24
Avg.	2.61	2.51	**2.22**	2.85	3.45	4.59
N-40-100-6	**1.03**	1.27	2.17	1.52	1.95	2.46
N-40-100-12	1.54	2.65	**1.29**	1.87	1.64	1.45
N-50-100-6	1.78	2.27	2.23	**0.36**	2.44	4.18
N-50-100-12	1.53	1.97	2.30	**1.16**	2.28	1.93
W-40-100-6	**1.15**	1.36	2.27	2.22	1.75	2.34
W-40-100-12	1.15	1.32	**0.98**	1.61	1.51	1.53
W-50-100-6	4.05	2.06	2.53	2.75	**1.00**	4.27
W-50-100-12	1.94	2.06	**1.77**	3.12	1.96	1.73
Avg.	**1.77**	1.87	1.94	1.82	1.82	2.49
N-40-200-12	**0.58**	2.78	2.09	4.07	5.62	2.51
N-40-200-24	1.25	2.69	**0.16**	2.36	5.49	3.76
N-50-200-12	2.06	**1.81**	4.15	5.25	6.22	2.60
N-50-200-24	0.60	2.76	**0.31**	5.04	1.14	4.55
W-40-200-12	**0.68**	2.84	1.57	4.59	6.21	6.57
W-40-200-24	7.69	**1.36**	1.68	1.38	3.77	4.55
W-50-200-12	5.95	3.56	**1.42**	10.17	8.30	9.98
W-50-200-24	1.74	4.02	**0.25**	5.40	5.18	5.07
Avg.	2.57	2.73	**1.45**	4.78	5.24	4.95
Overall	2.32	2.37	**1.87**	3.15	3.50	4.01

TABLE 4.1. Impact of parameter η

TS-AM. In this variant with adaptive memory, the maximum traveled distance cannot be exceeded when exploring the neighborhood of the current solution. Thus, a neighbor solution is ignored if the maximum distance of one or more technician routes is exceeded.

TS-I. This variant is obtained by removing the adaptive memory (AM) from *TS-AM-I*. One of the two greedy construction heuristics is applied, through a call to Greedy(), when it is time to get a new starting solution.

TS. This variant is obtained by removing the adaptive memory (AM) from *TS-AM*. One of the two greedy construction heuristics is applied, through a call to Greedy(), when it is time to get a new starting solution.

Table 4.2 reports, for each subset of 10 instances and each variant, the number of times a variant found the best solution (over the four variants) and the average gap in percentage

over the best solutions. Also, for each variant, the average CPU time in seconds to reach its best solutions is indicated.

These results demonstrate the importance of considering solutions that exceed the maximum traveled distance, as well as including the adaptive memory with biased fitness in our problem-solving methodology. In fact, TS-AM-I found the best solution for approximately 85% of the instances, as indicated in the line Overall of Table 4.2 (c.f., 8.46).

4.5.2.3. *Served tasks*

Table 4.3 provides a picture of the average number (#Ta) and percentage (%Ta) of served tasks, as well as the average number of tasks served per technician (#Ta/Te) for configurations with less or more technicians (#Te). For instances with 50 tasks, 3 technicians can serve about 40% of the tasks in the case of the smaller service area, with 6 or 7 tasks in each route. This percentage decreases for the larger service area because more time is needed to get from one task to another. When 6 technicians are available, the number of served tasks obviously increases, although the multiplier tends to be slightly less than 2. It should also be noted that more tasks can be served when the time windows are wide. Similar trends are observed for the instances of size 100 and 200.

4.5.3. Comparison with optimal solutions

Optimal solutions obtained with a branch-and-price algorithm are reported in [**45**]. This algorithm was able to routinely solve instances with up to 25 tasks and TS-AM-I found the optimum in each case. The branch-and-price algorithm began to strive for instances with 30 tasks and was not able to solve any instance with 50 tasks or more (within 24 hours of computation time). Again, TS-AM-I was able to optimally solve all instances for which the optimum was known.

4.6. Conclusion

We proposed a metaheuristic approach to address a TRSP for which exact methods can only solve small instances. Given the practical interest of this type of problem, the proposed methodology opens the way for further progress in this area. The metaheuristic is based on a tabu search enhanced with an adaptive memory, where the evaluation of each solution in memory is driven by both its cost and its contribution to diversity. Our algorithm reached the optimum on each try for instances with less than 50 tasks. It can also solve larger instances, as indicated by the results reported in this paper for instances with up to 200 tasks. Among many possible avenues of research, we now want to explore the application of our metaheuristic to dynamic variants of our TRSP, where new service requests must be integrated in real-time into the current routes.

Name	TS			TS-I			TS-AM			TS-AM-I		
	# Bst	Gap (%)	CPU (s)	# Bst	Gap (%)	CPU (s)	# Bst	Gap (%)	CPU (s)	# Bst	Gap (%)	CPU (s)
N-40-50-3	0	26.08	1 549.55	0	19.65	1 612.35	0	11.38	776.67	10	0.00	598.13
N-40-50-6	0	11.43	1 324.47	0	8.70	1 318.54	3	4.02	1 677.11	7	0.46	1 455.49
N-50-50-3	0	37.29	2 174.15	0	27.11	1 690.32	0	21.01	1 003.99	10	0.00	837.48
N-50-50-6	0	19.58	1 257.15	1	13.20	1 269.87	0	8.66	1 210.93	9	0.00	915.21
W-40-50-3	2	26.81	1 027.83	1	22.08	1 007.78	3	11.52	964.14	4	0.00	815.13
W-40-50-6	0	8.29	1 286.84	0	6.02	1 270.81	2	3.29	1 277.39	8	0.07	1 080.06
W-50-50-3	0	37.78	1 321.57	0	30.33	1 416.81	0	22.03	1 423.29	10	0.00	409.62
W-50-50-6	0	17.74	1 714.26	0	11.72	1 615.73	0	8.07	1 095.96	10	0.15	1 607.55
Avg.	0.25	23.12	1 456.98	0.25	17.35	1 400.28	1	11.25	1 178.69	8.50	0.09	964.83
N-40-100-6	0	10.76	1 664.72	0	9.13	1 661.01	0	7.09	1 217.81	10	0.00	1 166.25
N-40-100-12	0	7.11	1 912.72	0	5.64	1 712.91	0	3.47	1 374.16	10	0.46	1 375.70
N-50-100-6	0	24.07	1 205.76	0	17.13	1 286.56	2	13.13	809.84	8	0.00	791.34
N-50-100-12	0	8.23	1 579.18	5	6.81	1 697.00	0	6.14	1 439.92	5	0.00	1 193.54
W-40-100-6	0	9.24	1 503.54	0	6.85	1 614.83	0	5.02	1 519.72	10	0.25	1 499.16
W-40-100-12	0	5.84	1 812.49	1	2.74	1 620.98	0	2.11	1 219.38	9	1.08	1 084.01
W-50-100-6	0	27.93	1 291.44	1	16.88	1 381.28	1	13.38	1 123.35	8	0.00	1 026.33
W-50-100-12	0	6.74	1 312.40	4	4.02	1 316.81	1	5.51	1 146.69	5	0.22	575.40
Avg.	0	12.49	1 535.28	1.38	8.65	1 536.42	0.5	6.98	1 231.36	8.13	0.25	1 088.97
N-40-200-12	0	3.68	5 633.41	1	3.14	5 686.82	0	2.91	5 449.07	9	1.03	5 237.30
N-40-200-24	0	4.21	7 146.86	0	4.23	7 779.88	0	1.76	7 035.13	10	0.23	6 062.68
N-50-200-12	0	10.71	6 018.59	0	6.03	6 107.62	0	4.86	5 918.12	10	0.17	5 807.03
N-50-200-24	0	5.79	8 301.10	0	2.31	8 618.49	0	5.27	8 605.12	10	2.50	7 308.05
W-40-200-12	3	3.75	5 489.13	0	2.98	5 676.70	2	2.39	5 367.33	5	1.05	4 098.60
W-40-200-24	0	2.38	7 960.04	1	2.43	7 716.51	1	0.06	8 000.05	8	2.40	6 193.89
W-50-200-12	0	13.67	6 171.18	1	8.38	6 292.68	0	10.50	6 226.22	9	0.19	5 100.04
W-50-200-24	0	3.31	8 821.95	1	1.84	8 824.47	0	2.08	8 617.05	9	0.34	7 574.70
Avg.	0.38	5.94	6 942.78	0.5	3.92	7 087.89	0.38	3.73	6 902.26	8.75	0.99	5 922.79
Overall	0.21	13.85	3 311.68	0.71	9.97	3 341.53	0.63	7.32	3 104.10	8.46	0.44	2 658.86

TABLE 4.2. Comparison of three different variants

Name	Less technicians				More technicians			
	#Te	#Ta	%Ta	#Ta/Te	#Te	#Ta	%Ta	#Ta/Te
N-40-50	3	20.2	40.4%	6.7	6	37.5	75.0%	6.3
N-50-50	3	15.2	30.4%	5.1	6	31.8	63.6%	5.3
W-40-50	3	22.9	45.8%	7.6	6	38.1	76.2%	6.4
W-50-50	3	17.6	35.2%	5.9	6	32.1	64.2%	5.4
N-40-100	6	50.5	50.5%	8.4	12	83.6	83.6%	7.0
N-50-100	6	40.3	40.3%	6.7	12	69.3	69.3%	5.8
W-40-100	6	50.7	50.7%	8.5	12	85.2	85.2%	7.1
W-50-100	6	42.0	42.0%	7.0	12	70.3	70.3%	5.9
N-40-200	12	107.5	53.8%	9.0	24	180.5	90.3%	7.5
N-50-200	12	90.3	45.2%	7.5	24	155.8	77.9%	6.5
W-40-200	12	101.0	50.5%	8.4	24	193.1	96.6%	8.0
W-50-200	12	103.0	51.5%	8.6	24	161.0	80.8%	6.7

TABLE 4.3. Served tasks

CONCLUSION

Les contributions de cette thèse touchent à la modélisation et à la résolution d'un problème de tournées de technicien provenant d'une application pour la maintenance et la réparation d'équipements permettant les transactions électroniques.

Le chapitre 1 positionne d'abord notre problème par rapport aux problèmes classiques de tournées de véhicules.

Le chapitre 2 décrit ensuite précisément le problème de type TRSP étudié dans cette thèse. Un modèle de programmation linéraire mixte avec variables entières et continues est proposé et résolu à l'aide de CPLEX. Les résultats montrent que le problème est très difficile et que seules des instances de petite taille peuvent être résolues à l'optimum avec une telle approche. L'effort de modélisation rapporté dans ce chapitre ouvre la voie à des développements pour d'autres variantes du TRSP, en ajoutant ou en retirant des contraintes.

Le chapitre 3 attaque des problèmes de plus grande taille en proposant un algorithme de *branch-and-price*. Pour ce faire, nous considérons un problème maître de type *set packing* avec un sous-problème de génération de routes (colonnes). La formulation du problème maître est très simple, car les variables de décision correspondent à des routes réalisables pour chaque technicien. Pour la génération des routes, nous associons un sous-problème à chaque technicien. Ce dernier est un problème de plus court chemin élémentaire avec contraintes de ressources.Pour résoudre ces derniers, nous avons utilisé deux méthodes différentes, soit l'algorithme de programmation dynamique de Feillet [**22**] et une approche de relaxation décrémentale de l'espace de recherche (DSSR) [**57**]. Pour forcer l'intégralité de la solution, nous avons considéré deux types de branchement. Dans le premier cas, il s'agit d'un branchements binaire où la première branche oblige un technicien donné à effectuer une certaine tâche et où la deuxième branche oblige le technicien à ne pas faire cette tâche. Dans le second cas, il s'agit d'un branchement ternaire où la première branche oblige un technicien donné à effectuer une certaine tâche, la deuxième branche oblige un autre technicien à faire cette tâche et la troisième branche oblige tous les techniciens à ne

pas la faire.
Notre *branch-and-price* a pu résoudre des instances comprenant jusqu'à 45 tâches, apportant ainsi une amélioration par rapport aux résultats rapportés au chapitre précédent. Tel que prévu, la variante DSSR avec branchement ternaire l'a emporté en termes d'efficacité.

Le chapitre 4 décrit une nouvelle méthodologie heuristique combinant des idées provenant de différentes sources. Une recherche tabou avec mémoire adaptative a été proposée. Notre méthode inclut également une méthode avancée d'évaluation des solutions dans la mémoire adaptative qui considère à la fois des critères de qualité et de contribution à la diversité. Nous avons également relaxé la contrainte qui limite la distance maximale autorisée pour chacune des routes, afin d'explorer des solutions non réalisables, mais qui peuvent être facilement réparées.

Les développements possibles dans le futur sont assez nombreux étant donné que les TRSPs ont été relativement peu étudiés. La définition d'un problème de type TRSP est souvent liée à une application sous-jacente et des considérations pratiques additionnelles pourraient donc être abordées. Une piste intéressante, par exemple, serait d'intégrer un certain équilibre au niveau de la charge de travail entre les techniciens. Les études dans la littérature ont généralement pour objectif de minimiser la distance totale parcourue, ce qui peut mener à des solutions où les routes des techniciens sont fortement désiquilibrées. Il serait également intéressant d'intégrer la gestion des outils des techniciens, en particulier lorsque des outils spécialisés et très coûteux sont requis pour réaliser certaines tâches. Enfin, il existe des contextes où les routes doivent être synchronisées pour réaliser des tâches qui demandent l'intervention de plusieurs équipes de techniciens.

Dans une perspective plus large, il faut noter que cette thèse s'est intéressée à la variante déterministe du problème. C'est à dire que l'incertitude souvent inhérente à ces problèmes a été mise de côté. Par exemple, des pièces peuvent se révéler défectueuses ou encore le temps de service peut varier. De plus, certaines tâches peuvent apparaître de façon dynamique au cours de la journée d'opérations (tel qu'observé dans la maintenance corrective) et demander une intégration en temps réel aux routes courantes. Il serait donc souhaitable d'aborder dans le futur les aspects stochastiques et dynamiques de tels problèmes.

Enfin, il y a encore beaucoup de zones inexplorées sur le plan méthodologique, en particulier au niveau des méthodes exactes. Par exemple, la programmation stochastique semble avoir été ignorée jusqu'à présent. Pour ce qui est des méthodes heuristiques, la recherche à grand voisinage a déjà été utilisée avec succès et semble particulièrement prometteuse pour l'avenir.

Bibliographie

[1] F. Afrati, S. Cosmadakis, C.H. Papadimitriou, G. Papageorgiou et N. Papakostantinou : The complexity of the travelling repairman problem. *Informatique Théorique et Applications*, 20:79–87, 1986.

[2] R. Baldacci, N. Christofides et A. Mingozzi : An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

[3] R. Baldacci, E. Hadjiconstantinou et A. Mingozzi : An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52(5):723–738, 2004.

[4] D.J. Bertsimas et G.V. Ryzin : Stochastic and dynamic vehicle routing in the Euclidean plane with multiple capacitated vehicles. *Operations Research*, 41:60–76, 1993.

[5] F. Blakeley, B. Argüello, B. Cao, W. Hall et J. Knolmajer : Optimizing periodic maintenance operations for Schindler Elevator Corporation. *Interfaces*, 33(1):67–79, 2003.

[6] Y. Borenstein, N. Shah, E. Tsang, R. Dorne, A. Alsheddy et C. Voudouris : On the partitioning of dynamic workforce scheduling problems. *Journal of Scheduling*, 13:411–425, 2010.

[7] N. Bostel, P. Dejax, P. Guez et F. Tricoire : Multiperiod planning and routing on a rolling horizon for field force optimization logistics. *In* B. Golden, S. Raghavan et E. Wasil, éditeurs : *The vehicle routing problem : Latest advances and new challenges*, pages 503–525. Springer, 2008.

[8] C.E.Cortes, M.Gendreau, L.M. Rousseau, S.Souyris et A.Weintraub : Branch-and-price and constraint programming for solving a real-life technician dispatching problem. 238(1):300 – 312, 2014.

[9] A. Chabrier : Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33:2972 – 2990, 2006.

[10] S. Chatterjee, C. Carrera et A. Lucy : Genetic algorithms and traveling salesman problems. *European Journal of Operational Research*, 93:490–510, 1996.

[11] J-F. Cordeau, G. Laporte, F. Pasin et S. Ropke : Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409, 2010.

[12] C.E. Cortes, F. Ordonez, S. Sebastian et A. Weintraub : Routing technicians under stochastic service times : A robust optimization approach. *The Sixth Triennial Symposium on Transportation Analysis*, 2002.

[13] R.B. Damm, M.G.C. Resende et D. P. Ronconi : A biased random key genetic algorithm for the field technician scheduling problem. *Computers & Operations Research*, 75:49 – 63, 2016.

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

[15] I. Dayarian, T.G. Crainic, M. Gendreau et W. Rei : A column generation approach for a multi-attribute vehicle routing problem. Rapport technique CIRRELT-2013-57, CIRRELT, 2013.

[16] G. Desaulniers, F. Lessard et 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.

[17] M. Desrochers, J. Desrosiers et M.M. Solomon : A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.

[18] B. Dimitris et V.R. Garrett : *The Dynamic Traveling Repairman Problem*. Sloan School of Management, Massachusetts Institute of Technology, 1989.

[19] K.F. Doerner, M. Gronalt, R.F. Hartl, G. Kiechle et M. Reimann : Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows. *Computers & Operations Research*, 35(9):3034–3048, 2008.

[20] S. Lin et B.W. Kernighan : An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[21] J. Fakcharoenphol, C. Harrelson et S. Rao : The k-traveling repairmen problem. *ACM Transactions on Algorithms*, 3, 2007.

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

[23] T.A. Feo et M.G.C. Resende : A probabilistic heuristic for a computationally difficult set covering problem. *Informatique Théorique et Applications*, 8:67–71, 1989.

[24] G.N. Frederickson et B. Wittman : Speedup in the traveling repairman problem with constrained time windows. *CoRR*, abs/1101.3960, 2011.

[25] R. Fukasawa, H. Longo, J. Lysgaard, M.P. Aragao, M. Reis, E. Uchoa et R.F. Werneck : Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006.

[26] A. Garcia, P. Jodra et J. Tejel : A note on the traveling repairman problem. *Networks*, 40:27–31, 2002.

[27] S. Gélinas, M. Desrochers, J. Desrosiers et M.M. Solomon : A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61:91–109, 1995.

[28] M. Gendreau, F. Guertin, J.-Y. Potvin et É. Taillard : Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33(4):381–390, 1999.

[29] M. Gendreau, A. Hertz et G. Laporte : A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.

[30] M. Gendreau, G. Laporte et J.Y. Potvin : Vehicle routing : Modern heuristics. *In* E. Aarts et J.K. Lenstra, éditeurs : *Local Search and Combinatorial Optimization*, pages 311–337. Wiley, 2003.

[31] F. Glover : Tabu search-Part I. *ORSA Journal on computing*, 1(3):190–206, 1989.

[32] B.L. Golden, A.A. Assad et E.A. Wasil : The vehicle routing problem. chapitre 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.

[33] H. Hashimoto, S. Boussier, M. Vasquez et C. Wilbaut : A GRASP-based approach for technicians and interventions scheduling for telecommunications. *Annals of Operations Research*, 183(1):143–161, 2011.

[34] M. Held et R.M. Karp : A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10:196–210, 1962.

[35] B. Hofmann-Wellenhof, H. Lichtenegger et J. Collins : *Global Positioning Systems : Theory and practice*. Springer, 1993.

[36] S. Irnich et D. Villeneuve : The shortest-path problem with resource constraints and k-cycle elimination for k >= 3. *INFORMS Journal on Computing*, 18(3):391–406, 2006.

[37] M. Jepsen, B.R. Petersen, S. Spoorendonk et D. Pisinger : Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008.

[38] B. Kallehauge, J. Larsen, O.B. Madsen et M.M. Solomon : Vehicle routing problem with time windows. *In* G. Desaulniers, J. Desrosiers et M.M. Solomon, éditeurs : *Column Generation*, pages 67–98. Springer, 2005.

[39] S. Kawaguchi et Y. Fukuyama : Reactive tabu search for job-shop scheduling problems. *In Proceedings of the 11th International Conference on Computer Science Education (ICCSE)*, pages 97–102, 2016.

[40] N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon et F. Soumis : 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999.

[41] A. Kovacs, S.N. Parragh, K.F. Doerner et R.F. Hartl : Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15(5):579–600, 2012.

[42] J. Lempiainen et M. Manninen : *Radio Interface System Planning for GSM/GPRS/UMTS*. Springer, 2001.

[43] J. Lysgaard, A.N. Letchford et R.W. Eglese : A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

[44] I. Mathlouthi, M. Gendreau et J.-Y. Potvin : Branch-and-price for a multi-attribute technician routing and scheduling problem. Rapport technique CIRRELT-2017-56, Interuniversity Research Center on Entreprise Networks, Logistics and Transportation, 2017.

[45] I. Mathlouthi, M. Gendreau et J.-Y. Potvin : Mixed integer linear programming for a multi-attribute technician routing and scheduling problem. *Information Systems and Operational Research*, 56(1):33–49, 2018.

[46] J.-E. Mendoza, A. Montoya, C. Guéret et J.-G. Villegas : A parallel matheuristic for the technician routing problem with conventional and electric vehicles. *In 12th Metaheuristics International Conference (MIC)*, Barcelona, Spain, 2017.

[47] P. Moscato et C. Cotta : Memetic algorithms. *Handbook of Applied Optimization*, pages 157–167, 2002.

[48] I. Or : Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking. Rapport technique, PhD dissertation, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1976.

[49] V. Pillac, M. Gendreau, C. Guéret et A. Medaglia : A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1 – 11, 2013.

[50] V. Pillac, C. Guéret et A. Medaglia : On the dynamic technician routing and scheduling problem. *In 5th International Workshop on Freight Transportation and Logistics (ODYSSEUS)*, Mikonos, Greece, 2012.

[51] V. Pillac, C. Guéret et A. Medaglia : A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7(7):1525–1535, 2013.

[52] V. Pillac, C. Guéret et A. Medaglia : A fast reoptimization approach for the dynamic technician routing and scheduling problem. *In* L. Amadeo, E.-G. Talbi et F. Yalaoui, éditeurs : *Recent Developments in Metaheuristics*, pages 347–367. Springer, 2018.

[53] D. Pisinger et S. Ropke : A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

[54] J.-Y. Potvin et J.-M. Rousseau : A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993.

[55] M.G.C. Resende et C.C. Ribeiro : Greedy randomized adaptive search procedures : Advances, hybridizations, and applications. *In* M. Gendreau et J.-Y. Potvin, éditeurs : *Handbook of Metaheuristics*, pages 283–319. Springer, 2010.

[56] G. Righini et M.Salani : Dynamic programming algorithms for the elementary shortest path problem with resource constraints. *Electronic Notes in Discrete Mathematics*, 17:247 – 249, 2004. Workshop on Graphs and Combinatorial Optimization.

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

[58] J.B. Robinson : On the Hamiltonian game (a traveling-salesman problem). *RAND Memorandum RM-303*, 1949.

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

[60] S. Ropke et D. Pisinger : An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[61] M.W.P. Savelsbergh et M. Sol : The general pickup and delivery problem. *Transportation Science*, 29:17–29, 1995.

[62] J. Tabitha, C. Rego et F. Glover : Multistart tabu search and diversification strategies for the quadratic assignment problem. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 39(3):579–596, 2009.

[63] É. Taillard, P. Badeau, M. Gendreau, F. Guertin et J.-Y. Potvin : A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186, 1997.

[64] H. Tang, E. Miller-Hooks et R. Tomastik : Scheduling technicians for planned maintenance of geographically distributed equipment. *Transportation Research Part E : Logistics and Transportation Review*, 43(5):591 – 609, 2007.

[65] P. Toth et D. Vigo : *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2001.

[66] P. Toth et D. Vigo : Vrp with backhauls. *In* P. Toth et D. Vigo, éditeurs : *The Vehicle Routing Problem*, pages 195–224. Society for Industrial and Applied Mathematics, 2001.

[67] E. Tsang et C. Voudouris : Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. *Operations Research Letters*, 20(3):119 – 127, 1997.

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

[69] F. Vanderbeck : Branching in branch-and-price : a generic scheme. *Mathematical Programming*, 130(2):249–294, 2011.

[70] T. Vidal, T.G. Crainic, M. Gendreau et C. Prins : A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658 – 673, 2014.

[71] D. Weigel et B. Cao : Applying GIS and OR techniques to solve Sears technician-dispatching and home delivery problems. *Interfaces*, 29(1):112–130, 1999.

[72] S. Wouter, V. Pieter, V.B. Greet et V.O. Dirk : The multiconstraint team orienteering problem with multiple time windows. *Transportation Science*, 2011.

[73] J. Xu et S.Y. Chiu : Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7:495–509, 2001.

[74] T. Yunes, A. Moura et C. Souza : A hybrid approach for solving large scale crew scheduling problems. *In* E. Pontelli et C.V. Santos, éditeurs : *Practical Aspects of Declarative Languages*,

volume 1733 de *Lecture Notes in Computer Science*, pages 293–307. Springer, Berlin, Heidelberg, 2000.

[75] E. Zamorano et R. Stolletz : Branch-and-price approaches for the multiperiod technician routing and scheduling problem. *European Journal of Operational Research*, 257(1):55–68, 2017.