Contribution aux graphes creux pour le problème de tournées sur arcs déterministe et robustes : théorie et algorithmes
Sara Tfaili

To cite this version:

HAL Id: tel-01698313
https://tel.archives-ouvertes.fr/tel-01698313

Submitted on 1 Feb 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Pour obtenir le diplôme de doctorat

Spécialité : Mathématiques Appliquées

Préparée au sein de « Université Le Havre Normandie - LMAH »

Contribution aux Graphes Creux pour le problème de Tournées sur Arcs Déterministe et Robuste: théorie et algorithmes.

Présentée et soutenue par
Sara TFAILI

<table>
<thead>
<tr>
<th>Nom</th>
<th>Département</th>
<th>Institution</th>
<th>Rôle</th>
</tr>
</thead>
<tbody>
<tr>
<td>M. Marc DEMANGE</td>
<td>Professeur</td>
<td>School of Science, RMIT University, Melbourne - Australia</td>
<td>Rapporteur</td>
</tr>
<tr>
<td>M. Ibrahima DIARASSOUBA</td>
<td>Maître de Conférences</td>
<td>Université Le Havre Normandie</td>
<td>Examinateur</td>
</tr>
<tr>
<td>M. Abdellah SALHI</td>
<td>Professeur</td>
<td>Department of Mathematical Sciences, University of ESSEX, Colchester - UK</td>
<td>Rapporteur</td>
</tr>
<tr>
<td>M. El-Ghazali TALBI</td>
<td>Professeur</td>
<td>Polytech'Lille, Université de Lille 1</td>
<td>Examinateur</td>
</tr>
<tr>
<td>M. Abdelkader SBIHI</td>
<td>Professeur</td>
<td>ESCEM Ecole de Management, Tours</td>
<td>Co-directeur de thèse</td>
</tr>
<tr>
<td>M. Adnan YASSINE</td>
<td>Professeur</td>
<td>Université Le Havre Normandie</td>
<td>Directeur de thèse</td>
</tr>
</tbody>
</table>

Thèse dirigée par Adnan YASSINE et Abdelkader SBIHI, Laboratoire de Mathématiques Appliquées du Havre (LMAH), France
Acknowledgements

In the middle of difficulty lies opportunity.

Sometimes our light goes out, but is blown into flame by Allah and by some human beings.

All Praises to Allah, swt, for blessing my project...

I would like to express my special appreciation and thanks to my two supervisors Professor Adnan YASSINE and Professor Abdelkader SBIHI. I would like to thank you for encouraging my work and for allowing me to grow as a research scientist. Your advice on both research as well as on my career has been priceless, and without your supervision and constant help this dissertation would not have been possible. Your guidance helped me in all the time of research and writing of this thesis.

Professor SBIHI, it was a real privilege and an honor for me to share your exceptional scientific knowledge. Thanks for your availability and constructive suggestions which were determinant for the accomplishment of the work presented in this thesis. I have been lucky to have a supervisor like you who cared so much about my work, and who responded to my questions so promptly.

Besides my supervisors, thanks to every member of my thesis committee: Professor Marc DEMANGE, Professor Abdellah SALHI, Professor El-Ghazali TALBI and Dr. Ibrahima DIARRASSOUBA for participating as committee members of the doctoral examination committee even at hardship. I am pleased for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions.

To my colleagues in the applied mathematics laboratory “LABORATOIRE DE MATHEMATIQUES APPLIQUEES DU HAVRE-LMAH” and to Professor Aziz ALAOUI, thank you for hosting me during my doctoral years.

My sincere gratitude is reserved for my colleagues Dr. Hamdi DKHIL, and Dr. Ibrahima DIARRASSOUBA who provided me an unconditional support to develop my thesis and particularly for the C++ and JAVA coding check
parts. Without your precious help, it would not be possible to conduct this research.

I am also indebted to the “Association de Spécialisation et d’Orientation Scientifique” for the financial support.

My deepest thanks go to the ones who were the first supporters of my dream; Dr. Hussein Katbye and Dr. Mahmoud Koabaz.

To all my friends in Le Havre, especially to those who welcomed and supported me to overcome the obstacles and the difficulties I faced, thank you. Tatiana Fawaz, Joanna Eid, LE CEDRE restaurant staff and to YASSINE’s family, I am deeply grateful for every single encouraging moment you offered me. Tatiana, I would never forget your precious help and continuous motivation.

A special thanks go to my family. Words cannot express how grateful I am to my mother and father for all of the sacrifices that they have made on my behalf. Your prayer for me was what sustained me thus far. I would also like to thank my sisters Fatima and Nour, and my brother Hussein for their infinite inspiration to strive towards my goal. I am also grateful to my uncle Abbas for his support. At the end, I would like to express appreciation to my beloved fiancé Salman, who was always beside me in the moments when there was no one to answer my queries.

To all those who have rekindled this light...

I owe the deepest thanks.

Le Havre, December 1st, 2017
1 Preliminaries, Basic Definitions and Notations .......................................................... 9
   1.1 Introduction ............................................................................................................ 9
   1.2 Generalities on Graphs ....................................................................................... 9
       1.2.1 Directed and undirected graphs ................................................................. 10
       1.2.2 Particular graphs and their properties ....................................................... 12
   1.3 Combinatorial Optimization .................................................................................. 16
   1.4 Complexity of an Optimization Problem .............................................................. 17
   1.5 Research Motivations ........................................................................................... 18
   1.6 Coming Chapters .................................................................................................. 19

2 Node and Arc Routing Problems ............................................................................. 21
   2.1 Node routing problem .......................................................................................... 21
       2.1.1 Mathematical formulation of the capacitated vehicle routing problem .... 23
       2.1.2 Sparse CVRP .............................................................................................. 24
   2.2 Arc routing problem ............................................................................................ 28
       2.2.1 The Chinese Postman Problem (CPP) ....................................................... 29
       2.2.2 The Rural Postman Problem (RPP) ............................................................ 29
       2.2.3 Multi-vehicle Arc Routing Problems ......................................................... 29
       2.2.4 Capacitated Chinese Postman Problem (CCPP) ...................................... 30
       2.2.5 Capacitated Arc Routing Problem (CARP) .............................................. 30
       2.2.6 CARP Applications .................................................................................... 36
       2.2.7 Different Variants of CARP ...................................................................... 40
       2.2.8 Different Formulations of the CARP ......................................................... 45
   2.3 Robust Optimization ............................................................................................. 52
3 The Sparse CARP: Mathematical Formulation and Transformations

3.1 Introduction .................................................. 55
3.2 Mathematical formulations of the sparse CARP .......... 56
  3.2.1 Mathematical formulations of the sparse capacitated arc routing problems .................. 57
3.3 From ARP to VRP: a general review ..................... 62
  3.3.1 A first transformation technique ..................... 62
  3.3.2 A second transformation technique ................. 65
  3.3.3 A compact transformation technique of arc routing problem into node routing problem .... 66
3.4 A transformation technique of sparse CARP into CVRP with sparse feasible graph ............ 69
  3.4.1 The steps of the transformation technique .......... 69
  3.4.2 Costs Division ........................................... 71
3.5 Graph density conservation ............................... 74
  3.5.1 Validation of the transformation ..................... 77
3.6 Conclusion .................................................. 81

4 Efficient Algorithms for Solving the Sparse CARP 83

4.1 Solution approaches for the sparse CARP ............... 84
  4.1.1 An initial solution for the transformed CARP .... 84
  4.1.2 Dynamic graphs ........................................ 84
  4.1.3 The constructive heuristic CH ....................... 84
  4.1.4 An improving randomized procedure based heuristic: IRP ................................ 89
  4.1.5 A tabu search algorithm for the sparse CARP .... 91
4.2 Numerical Results ........................................ 94
4.3 Conclusion ................................................ 103

5 Robust Sparse CARP with Uncertain Travel Costs 105

5.1 Review about Robust Optimization ...................... 106
5.2 Uncertain Vehicle Routing Problems ..................... 108
  5.2.1 Stochastic Vehicle Routing Problem ............... 108
  5.2.2 Robust Vehicle Routing Problem ................. 108
5.3 Capacitated Arc Routing Problem under Uncertainty 109
  5.3.1 Robust Capacitated Arc Routing Problem under sparse graph and under Travel Costs Uncertainty 110
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>Efficient algorithms for solving robust sparse CARP</td>
<td>113</td>
</tr>
<tr>
<td>5.4.1</td>
<td>A heuristic algorithm for solving the robust sparse CARP under travel costs uncertainty</td>
<td>113</td>
</tr>
<tr>
<td>5.4.2</td>
<td>An adapted tabu search algorithm for solving the robust sparse CARP under travel costs uncertainty</td>
<td>118</td>
</tr>
<tr>
<td>5.5</td>
<td>Computational Experiments</td>
<td>120</td>
</tr>
</tbody>
</table>
General Introduction

During the last decades, an important progress in the real life applications that concern scheduling and transportation has occurred. Such a development that lies in the fields of operations research and applied mathematics spur the enterprises to ameliorate their strategies especially that the number of people involved in the flow of goods and information has increased, and this leads to strong implications for product traceability, improved information systems, cost minimization and trade facilitation.

On one hand, all the enterprises aim at satisfying the demands of their customers in the different activities of production, transport, buying and selling. On the other hand, they wish to achieve this satisfaction in the best possible conditions including the terms of optimization and profit as well. As a result, it is important to develop efficient algorithms that allow the achievement of all these goals. Considering the domain of transportation, the most well known problems are the node routing problems (VRP) and the arc routing problem (ARP) and their variants. The Arc Routing Problem (ARP) consists in determining a least cost traversal of some arcs of the network subject to side constraints. Such problems are encountered in a variety of practical situations such as road maintenance, street sweepers, snow-plowing, salt gritting, garbage collection, mail delivery, school bus transportation, meter reading, network maintenance (electrical lines or gas mains inspection), etc. In all these applications, each street segment must be covered in its entirety. In this work, we start with a brief description of the VRPs and ARPs that have been treated so far in the literature, and we focus on the capacitated arc routing problem (CARP) over a special class of underlying graphs under a deterministic case and under a set of scenarios.

Throughout the following dissertation, we deal with CARP under sparse
underlying graphs. For simplification, we call this problem the sparse CARP. More precisely and to avoid any misunderstanding that may concern the definition of our studied problem, we treat the CARP with underlying graphs of densities between 0.5 and 0.001. Such CARP problem instances have not been treated before, and to the most of our knowledge, we are the first who study such cases. The importance of this study lies in its applications especially in the cases of crisis such as wars and distribution of dangerous and sensitive goods as chemical or nuclear substances or medicines. For instance, if a natural disaster took place such as earthquake or hurricane, then any transportation will be limited on the resulting sparse underlying graph. This is due to the fact that many routes will be cut and barred. In this thesis, we study (1) the sparse CARP which is defined over a sparse underlying graph and (2) the robust sparse CARP under travel costs uncertainty. New mathematical formulations, new resolution methods and new algorithms are introduced. We develop greedy heuristics and new adapted metaheuristics based on adapted tabu search algorithm.

This thesis is organized as following:

In the first chapter, we present the basic notions and definitions of graph theory and combinatorial optimization. A survey about the Vehicle Routing Problem (VRP) and the Arc Routing Problem (ARP) and their variants with emphasis on the capacitated arc routing problem (CARP), their formulations and applications are all presented in the second chapter. The different transformation techniques of ARP into VRP which have been presented in the literature are reviewed in chapter 3. Moreover, a new transformation technique of ARP into VRP over sparse underlying graphs is introduced in this chapter, and new mathematical formulations of the sparse CARP are presented too. The fourth chapter is dedicated to the new heuristic and metaheuristic algorithms which are developed to solve the studied problem. The chapter is ended by a set of computational experiments that show the effectiveness of the developed approaches and methods. In the last chapter, we treat the robust sparse capacitated arc routing problem under travel costs uncertainty where we introduce this uncertainty by a set of scenarios. A mathematical modeling of the robust problem is given in addition to a greedy heuristic algorithm and an adapted tabu search algorithm to solve the problem. Promising numerical results showing the robustness of the developed methods are presented in the end of the chapter.
We conclude the dissertation of this thesis by a general conclusion that summarizes our developed work and by a set of perspectives in addition to detailed bibliographies.
Introduction Générale

Face à une concurrence de plus en plus sévère, les entreprises sont contraintes à améliorer leur performance et leur moyen de production ainsi que l'utilisation rationnelle des ressources à la fois humaines, financières et en matières premières nécessaires à la bonne marche de leur modèle économique. Cette course à la performance incite les entreprises à identifier les problèmes auxquels ils font face, de les résoudre et d’implanter la (les) solution(s) obtenues.

En même temps, de réels progrès ont été accomplis dans différents domaines de la vie réelle en se basant sur des méthodes d’optimisation et de simulation. En effet, différents domaines en ingénierie, industrie, finance ou en gestion ont connu un formidable essor grâce à l’utilisation de ces techniques. Ce développement est dû en partie aux progrès réalisés par les mathématiques appliquées, en particulier l’optimisation combinatoire (théorie des graphes, théorie de la complexité,...) et la recherche opérationnelle (modélisation, résolution algorithmique,...) et grâce aussi au développement inédit de l’informatique et de la puissance du calcul dont nous disposons aujourd’hui.

Les différents problèmes pratiques que l’on peut identifier débouchent le plus souvent sur des modèles (d’optimisation et/ou de simulation) que l’on est appelé à résoudre. Or ces problèmes sont le plus souvent (et surtout) difficiles à résoudre à cause de leur complexité inhérente à la fois structurelle et humaine.

Il est donc important de développer des stratégies de résolution capables d’apporter une (des) réponse(s) à (aux) question(s) que l’on se pose et qui permet (ttent) de réaliser le (les) objectif(s) fixé(s). Un des domaines les plus importants en optimisation est celui des transports connu plus généralement sous le nom du problème de tournées de véhicules. Dans la littérature, il peut exister plusieurs déclinaisons ou variantes de ce problème (tournées de
véhicules sur nœuds (VRP), sur arcs (ARP),...). Ces problèmes consistent à déterminer le parcours le moins coûteux sur le réseau routier soumis à un ensemble de contraintes. On rencontre de tels problèmes dans diverses situations pratiques telles que la livraison de marchandises, l'entretien des routes, les balayeuses de rues, le déneigement, le salage, la collecte des déchets ménagers, la livraison du courrier, le transport scolaire, le relevé des compteurs, ...

Dans cette thèse, nous étudions un problème particulier des problèmes de tournées que l'on nomme le problème de tournées sur arcs avec graphe creux. Pour chacune des tournées sur ce type de graphe, chaque arc doit être intégralement couvert. Dans notre travail, nous commençons par une description des problèmes de type VRP et ARP qui ont été traités jusqu'à présent dans la littérature, ensuite nous nous concentrons sur le problème tournées sur arcs avec capacité (capacitated arc routing problem) (CARP) défini sur une classe spéciale de graphes sous-jacents à la fois dans le cas déterministe et avec coûts sous incertitude (ensemble de scénarios).

Dans toute la suite et pour simplifier, nous appelons ce problème le CARP creux (sparse ARP). Plus précisément et pour éviter toute confusion dans la définition de notre problème, nous traitons le CARP avec des graphes sous-jacents de densité comprise entre 0.5 et 0.001.

A notre connaissance, ce type de problèmes dit sparse CARP n'a jusqu'ici pas été étudié et nous sommes les premiers à proposer une telle étude. L'importance de cette étude réside dans ses applications, en particulier dans les cas de crises (guerre, manifestations de rue, ...) ou de catastrophes naturelles (seismes, inondations, ...) ou encore dans la distribution de produits hautement dangereux. Dans le cas d'un tremblement de terre ou d'un ouragan, le transport sera limité et restreint sur le réseau routier. On peut définir un tel réseau par un graphe sous-jacent creux. Cela est dû au fait que de nombreuses routes seront coupées et/ou barrées car seuls certains tronçons du réseau routier seraient requis.

Le travail présenté dans cette thèse est organisé en deux parties essentielles: la première partie est dédiée à l'étude du problème sparse CARP déterministe tandis que la seconde partie est consacrée au problème sparse CARP avec coûts sous incertitude.

Cette thèse est organisée selon le schéma suivant. En effet, dans le premier
chapitre, nous présentons tout d’abord les notions de base et les différentes définitions de la théorie des graphes, de l’optimisation combinatoire. Ensuite, nous donnons une revue de littérature détaillée sur les deux types de tournées de véhicules: sur noeuds (VRP) et sur arcs (ARP) ainsi que certaines de leurs variantes. Par la suite, nous nous focalisons sur le problème de tournées sur arcs avec capacité (CARP) et ses principales variantes. Dans le deuxième chapitre, nous rappelons les formulations de ces problèmes et leurs applications.

Dans le troisième chapitre, nous détaillons l’étude du sparse CARP. Nous rappelons d’abord les différentes techniques de transformation des problèmes ARP en VRP qui ont été développées dans l’état de l’art. Ensuite, nous proposons notre nouvelle technique de transformation du sparse ARP en sparse VRP sur un graphe approprié sous-jacents creux. Nous détaillons les différentes formulations mathématiques du problème sparse CARP ainsi que leur validation et les différents résultats théoriques qui en découlent. Le quatrième chapitre est consacré aux approches algorithmiques basées sur des métaheuristiques adaptées pour résoudre le problème étudié. Le chapitre se termine par un ensemble d’expériences numériques qui montrent l’efficacité des approches et des méthodes développées.

Dans le cinquième chapitre, nous traitons le sparse CARP robuste sous incertitude de coûts de transport, où nous introduisons cette incertitude par un ensemble de scénarios sur ces coûts. Nous proposons un modèle mathématique du problème robuste. Ensuite, nous proposons différents algorithmes approchés. Un premier algorithme est de type glouton assurant la construction d’une solution réalisable sous forme de tâches ordonnancées complétées par un algorithme de type Dijkstra afin de construire la tournée au coût minimum du pire scénario. Une deuxième approche heuristique est une recherche de type tabou adapté qui démarre avec la solution obtenue par l’algorithme gouton qui a identifié le pire scénario. Il s’agit d’une approche améliorante sur ce pire scénario. Nous avons développé aussi un certain nombre de composantes (diversification, intensification, ...) pour améliorer la solution. A la fin du chapitre, nous présentons nos résultats numériques prometteurs qui montrent la robustesse des méthodes développées.

Nous terminons le mémoire de cette thèse par une conclusion générale qui résume notre travail de recherche et par un ensemble de perspectives en plus d’une bibliographie détaillée.
Chapter 1

Preliminaries, Basic Definitions and Notations

1.1 Introduction

Throughout this chapter, we present some basic definitions, notations and concepts of Graph Theory and Combinatorial Optimization that we are going to use in the following chapters. In the first section, we introduce the general concepts of graphs and their properties, and generalities about Combinatorics are presented in the second section. In the last section, we study the complexity of algorithms.

1.2 Generalities on Graphs

A graph is an abstract concept describing a set of objects where some pairs of objects are connected by links. Practically, we represent a graph by a diagram in a plane or space consisting of points (vertices) and lines (edges) as follows:

- A graph $G$ is a couple $(V(G), E(G))$, where $V(G)$ and $E(G)$ are two disjoint sets such that, $V(G)$ is the non-empty set containing the vertices of $G$ and $E(G)$ is the set of edges of $G$. More Precisely, we say $V := V(G) = \{1, \ldots, n\}$ and $E := E(G) = \{e; e = \{i, j\}, \text{where } i, j \in V(G)\}$. Henceforth, denote by $G(V, E)$ the graph $G$.

- The order of $G$, denoted by $v(G)$, is the number of vertices of $G$. 
The size of $G$, denoted by $e(G)$, is the number of edges of $G$.

In the following, we recall the definitions and properties of several types of graphs.

### 1.2.1 Directed and undirected graphs

#### Undirected graphs

We say that a graph $G$ is undirected (or non-oriented) if the precision of the sense of the link $(i,j)$ as well as the distinction between the initial extremity and the terminal one play no role i.e. $(i,j) = (j,i)$. In this case, a link $(i,j) \in E$ is said to be an edge in $G$ and it is represented graphically without any arrow (see Figure 1.1).

In our work, the graphs which we are working with are undirected, finite, simple and loop free.

#### Directed graphs

We say that $G$ is a directed graph (or digraph) if there is a distinction between the links $(i,j)$ and $(j,i)$ i.e. $(i,j) \neq (j,i)$. In this case, the link is said to be an arc. An arc $(i,j)$ is a directed edge from $i$ to $j$, where $i$ is said to be the tail of $(i,j)$ and $j$ is said to be the head of $(i,j)$, and we write $i \rightarrow j$ (see Figure 1.1). Generally, all the graphs are directed, and if the direction is symmetric then it is an undirected graph.

#### Symmetric graph

A graph $G$ is said to be symmetric graph if between every two nodes $i, j \in V$, the number of the arcs of form $(i,j)$ is equal to the number of the arcs of form $(j,i)$.

#### Subgraphs

- A subgraph $H$ of $G$ is a graph verifying $(V(H) \subseteq V(G))$, and $(E(H) \subseteq E(G))$. When $H \subset G$ but $H \neq G$, then we call $H$ a proper subgraph of $G$. When $V(H) = V(G)$, then $H$ is said to be a spanning subgraph of $G$. 
1.2. GENERALITIES ON GRAPHS

- An induced subgraph $H$ of $G$ is a subgraph of $G$ such that $V(H) \subseteq V(G)$, and $\{i, j\} \in E(H)$ if and only if $\{i, j\} \in E(G)$, for all $i, j \in V(H)$.

- A subgraph $H$ of $G$ is said to be stable if $E(H) = \emptyset$.

Neighbors and degrees

**First case** $G$ is an undirected graph, and $v \in V(G)$.

- The neighborhood of $v$, denoted by $N(v)$, is the set of vertices that are adjacent with $v$ in $G$.

  More precisely, $N(v) = \{u \in V(G) \text{ such that } \{u, v\} \in E(G)\}$.

- The degree of $v$ in $G$, denoted by $d_G(v)$, is defined by $d_G(v) = |N(v)|$.

- We define the maximum degree of $G$ by $\Delta(G) = \max \{d_G(v); v \in V(G)\}$. On the other hand, denote by $\delta(G) = \min \{d_G(v); v \in V(G)\}$ the minimum degree of $G$.

- We say that a graph $G$ is $k$-regular if $d_G(v) = k$ for all $v \in V(G)$.

- The neighborhood of an edge $e$ is the set of edges which are adjacent with $e$ in $G$.

**Second case** $G$ is a digraph, and $v \in V(G)$.

- The set of the out-neighbors of $v$ is denoted by $N^+(v)$ and is defined by $N^+(v) = \{u \in G; (v, u) \in E(G)\}$.

  The out-degree of $v$ is the cardinal of $N^+(v)$, and we write $d^+_G(v) = |N^+(v)|$. The maximum out-degree of $G$ is denoted by $\Delta^+(G)$, and the minimum out-degree of $G$ is denoted by $\delta^+(G)$.

- The set of the in-neighbors of $v$ is denoted by $N^-(v)$ and is defined by the following set.

  $N^-(v) = \{u \in V(G); (u, v) \in E(G)\}$. The in-degree of $v$ is the cardinal of $N^-(v)$, and we write $d^-_G(v) = |N^-(v)|$. The maximum in-degree is denoted by $\Delta^-(G)$, and the minimum in-degree is denoted by $\delta^-(G)$.

- A regular directed graph must satisfy the condition that the in-degree and the out-degree of all the vertices are equal to each other.
Remark 1.2.1. For every $v \in G$, we have $d_G^-(v) + d_G^+(v) = d_G(v)$.

The following statement is sometimes called the “The First Theorem of Graph Theory” or “The Handshake Lemma” \[149\].

**Proposition 1.2.1. (Degree-Sum Formula)**

Let $G$ be a graph. We have $\sum_{v \in V(G)} d_G(v) = 2|E(G)|$.

Indeed, every edge $xy$ is counted twice. Once in $d_G(x)$ and once in $d_G(y)$.

**Paths and cycles**

- A path $P$ is a graph $P = v_1v_2\ldots v_t$ such that $v_i \neq v_j \forall i \neq j$, with a set of vertices $V(P) = \{v_1, v_2, \ldots, v_t\}$ and set of edges $E(P) = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{t-1}, v_t\}\}$. In other terms, a path is a succession of vertices and consecutive edges which are incident. The length of $P$ is $l(P) = e(P) = t - 1$. As an example, refer to Figure 1.1.

- A $u-v$ walk is defined as a sequence of vertices starting at $u$ and ending at $v$, where consecutive vertices in the sequence are adjacent vertices in the graph.

- A cycle $C$ is a graph $C = v_1v_2v_1$ such that the set of vertices of $C$ is $V(C) = \{v_1, v_2, \ldots, v_t\}$ and the set of edges is given by $E(C) = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{t-1}, v_t\}, \{v_t, v_1\}\}$. with $t \geq 3$.

  For instance, in Figure 1.1 we give a cycle over 6 vertices.

  The length of $C$ is $l(C) = e(C) = t$. We say that $C$ is even(resp. odd) if its order is even(resp. odd).

- The girth of a graph $G$, denoted by $g(G)$, is the length of a shortest cycle contained in $G$.

**1.2.2 Particular graphs and their properties**

**Simple graph**

A loop is an edge that has the same initial and end vertex, and multiple edges are edges joining the same two vertices i.e. it is a cycle of length two. We call $G$ a simple graph if it contains neither loops nor multiple edges.
1.2. GENERALITIES ON GRAPHS

Complete graph

A complete graph $K_n$ is a graph of order $n$ such that for any two distinct vertices $i$ and $j$ of $G$, there exists an edge $\{i, j\}$. Note that $e(K_n) = \frac{n(n-1)}{2}$. In other words, $G$ is complete graph if and only if $\{i, j\} \notin E \Rightarrow \{j, i\} \in E$.

See Figure 1.1. In contrast, An empty graph is a graph formed of isolated vertices i.e. a graph with no edges.

Mixed graph

A mixed graph on $v$ vertices is an ordered pair $(V, C)$ where $V$ is a set of vertices, $|V| = v$, and $C$ is a set of ordered and unordered pairs of elements of $V$. An ordered pair is called an arc and an unordered pair is called an edge.

Figure 1.1: Different types of graphs.
Connected graph

- A graph $G$ is said to be connected if for all $i,j \in V$, there exists an $ij$-path i.e. a path of initial vertex $i$ and terminal vertex $j$.

- A connected component of $G$ is a connected induced subgraph of $G$ maximal with this property.

Dense graphs

To give a simple definition of dense graphs, we can say that a dense graph is a graph $G$ in which the number of edges of $G$, $e(G)$, is close to the maximal number of edges.

Adjacency matrix of a graph

Let $G$ be a graph with $V(G) = \{1, 2, \ldots, n\}$ and $E(G) = \{e_1, e_2, \ldots, e_m\}$. The adjacency matrix of $G$, denoted by $A(G)$, is the $n \times n$-matrix where the rows and the columns are indexed by $V(G)$.

The $(i,j)$-entry of $A(G) = \begin{cases} 0, & \text{if the vertices } i \text{ and } j \text{ are non-adjacent with } i \neq j; \\ 1, & \text{if the vertices } i \text{ and } j \text{ are adjacent with } i \neq j. \end{cases}$

The $(i,i)$-entry of $A(G)$ is 0 for all $i = 1, 2, \ldots, n$.

Networks

A network is a graph which is carried out by additional information such as real costs, distances, capacities, etc. For instance, The links of the graph can hold distances and the vertices may have some demands in certain cases of applications.

Sparsity

In this subsection, we give a definition of the concept of sparsity in graphs. In fact, sparsity is a very wide notion that characterizes some particular fields as sparse matrix, sparse graph and sparse set.

The definition of sparsity is specific for each problem, but this concept could be generalized to fit them all. We aim in this section at presenting a
1.2. GENERALITIES ON GRAPHS

well-concerned definition of the sparse graphs as we are interested in studying the capacitated arc routing problem over sparse graphs. The authors in [126] present a detailed study about sparsity in graphs, structures and algorithms. There is no exact and specific definition for sparse graphs, but they are characterized by comparing them with other graphs i.e. they are considered sparse relative to other graphs. As said before, sparsity is a vague notion which is applied to a class of graphs. Studying the sparse structures is related basically to the conservation of the global properties of these structures which is not the case in the dense structures. The authors study the dichotomy between the sparse and dense graph in detail, once from a topological point of view where geometrical graph transformations are considered such as minors and immersions and they restrict their use with further local constraints to ensure the stability of the global shape, and another time by considering the generalizations of chromatic numbers, the homomorphism paradigm, limits and universal structures, structural Ramsey theory, various tree-and branching-structures as a measure of approximation and various notions related to topological graph theory. This mixture of topology and geometry on the one hand and of combinatorics and algebra on the other hand is central to their analysis.

However, to be simpler in such definition especially for what concerns our work, a sparse graph is a graph which has a relatively small density which is defined to be the ratio of the number of actual edges of the graph to the maximum possible number of edges in the graph. It is a graph $G = (V, E)$ with a small density $d(G)$. Let $X \subseteq V$ and $\gamma(X)$ be the number of the induced edges in $X$. For each $X (\neq \emptyset) \subseteq Z$, if $\gamma(X) \leq k|X| - \ell$ then we say that $G$ is $[k, \ell]$-sparse in $Z$ ($Z(\neq \emptyset) \subseteq V$). If $k + 1 \leq \ell \leq 2k - 1$, then we say that $G = (V, E)$ is $[k, \ell]$-sparse in $Z$ ($Z(\neq \emptyset) \subseteq V$) if $G$ is loop-less and $\gamma(X) \leq k|X| - \ell$ holds for each $X(\neq \emptyset) \subseteq Z$ ($|X| \geq 2$) [69]. Similarly, we say that a graph $G = (V, E)$ is $[k, \ell]$-sparse if $|E| = k|V| - \ell$ and $G$ is $[k, \ell]$-sparse in $V$. For more details on both the sparsity and the $[k, \ell]$-sparsity, the reader can refer to [69] and [126].

In this work and throughout the following chapters, we identify the sparse structure of the graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ by measuring
the density of the latter which is given by the following formula:

\[ d(G) = \frac{2m}{n(n-1)} \] (1.1)

The detailed determination of our sparse graphs is given in the becoming chapter. However, a graph with only a few edges is a sparse graph. Note that a more precised definition and details about sparse and dense graphs will be introduced later in Chapter 3.

### 1.3 Combinatorial Optimization

Combinatorial Optimization is a branch of Mathematical Optimization with a wide number of applications. Technically speaking, Combinatorial Optimization is concerned with finding an optimal or close to (near) optimal solution among a finite collection of possibilities. The finite set of possible solutions is typically described through mathematical structures, like graphs, matroids or independence systems. The focus in Combinatorial Optimization lies on efficient algorithms which, more formally, means algorithms with a running time bounded by a polynomial in the input size. Therefore, two of the arguably most prominent questions in Combinatorial Optimization are:

- How quickly can one find a single (or all) optimal solution(s) of a given problem?
- When dealing with a problem where, due to complexity-theoretic reasons, it is unlikely that an optimal solution can be found efficiently: what is the best solution quality that an efficient algorithm can guarantee?

Over the last decades, Combinatorial Optimization has grown into a very mature field with strong links to various other disciplines like discrete mathematics (graph theory, combinatorics,...), computer science (data structures, complexity theory,...), probability theory, continuous optimization, and many application areas. Advances in modern Combinatorial Optimization thus often happen through clever combinations of ideas from several fields.
1.4. COMPLEXITY OF AN OPTIMIZATION PROBLEM

Let $\mathcal{F}$ be a family of subsets of a finite set $E$ and let $w : X \rightarrow \mathbb{R}$ be a real-valued weight function defined on the elements of $E$. The objective of the combinatorial optimization problem is to find $F^\ast \in \mathcal{F}$ such that:

$$w(F^\ast) := \min_{F \in \mathcal{F}} w(F)$$

where $w(F) := \sum_{e \in F} w(e)$.

To translate the combinatorial optimization problem into an optimization problem in $\mathbb{R}^X$, we can represent each $F \in \mathcal{F}$ by its incidence vector. Let

$$\xi^F_e = \begin{cases} 1 & \text{if } e \in F; \\ 0 & \text{otherwise}. \end{cases}$$

Then, if we let $S = \{\xi^F : F \in \mathcal{F}\} \subset \{0, 1\}^X$ be the set of incidence vectors of the sets in $\mathcal{F}$, the corresponding optimization problem is:

$$\min\{w^T x : x \in S\}.$$

An algorithm for a combinatorial optimization problem is a step-by-step procedure used to solve the problem within a finite number of steps, usually implemented as a computer program. Depending on which strategy we use to solve the combinatorial optimization problem (CO), the obtained solution if it exists is an exact (optimal) solution or a near-optimal one. The first strategy requires in general huge consuming time and is only efficient for small size problems. However, the second strategy is also consuming time one but with less intensity.

We say that an algorithm solves a combinatorial optimization problem if it determines a feasible solution for each instance of the problem, and efficient algorithms are characterized by their running time; the most efficient algorithm is supposed to be the fastest one and here the importance of time complexity appears.

### 1.4 Complexity of an Optimization Problem

There exist several types of complexities \[52, 53\]. One can mention the time and the space complexities. As mentioned above, the running time of an
algorithm represents the time complexity of this algorithm. To measure the performance and the efficiency of a used algorithm, generally we consider the running time. However, the time complexity is depending on the size of the problem instance to solve. Basically, there are two large classes of optimization problems: the class $\mathbf{P}$ that contains the polynomial problems, and the class $\mathbf{NP}$ that contains the NP-hard problems. For more details, the reader may refer to [79].

All the studied problems in this thesis are NP-hard problems.

1.5 Research Motivations

In this section, I present the research plan that motivates the study of the arc routing problems and their graphs. We especially consider the study of ARP on sparse feasible graphs. Throughout the research, all arc routing problems were considered on their graphs. The main study leads us to build a transformation technique in order to convert an ARP on sparse graph into a VRP on a sparse graph. This thesis work is inspired by the thesis in [128] where the author has considered the vehicle routing problem on its original sparse graph due to the following motivations:

1. Converting a sparse instance into a complete one requires an all-pairs shortest path algorithm.

2. Despite of the performance of the shortest path algorithm, converging may be time-consuming.

3. Some special structures which could be beneficial for solving the problem of the original instance could be lost by converting.

4. The resulting instance needs much more memory the original sparse one to be stored.

5. The transformed instance may have many alternative optima that were not present in the original one.

Based on these motivations, the novelty of our work is consecrated in the following contributions:
• We consider for the first time the capacitated arc routing problem on a sparse feasible graph. The importance of this work lies in its real life applications that are mainly concerned with transportation and distribution in the crisis cases such as wars and natural disasters.

• A transformation technique of the capacitated arc routing problem over sparse graph into capacitated vehicle routing problem over sparse graph is developed. The novelty of this technique is that it conserves the sparsity of the original underlying graph as well the structure of the problem.

• We present a robust multi-scenario capacitated arc routing problem over sparse graphs. In detail, we study the capacitated arc routing problem over sparse graphs under travel costs uncertainty, and we apply an adapted robust optimization method to solve the problem.

All these motivations in addition to the hard work done in [128] that considers the VRP on its original sparse graph encourage us to consider a new transformation technique that transforms an ARP problem into VRP one conserving all the properties of the original problem including the sparsity itself.

1.6 Coming Chapters

In Chapter 2, we give a survey about the vehicle and arc routing problems concentrating on the last research that deals with sparse graphs. Definitions and mathematical formulations in addition to the variants and applications of these problems are all briefly presented. In Chapter 3, we present the most known transformation techniques that transform an ARP into VRP, and we introduce our new technique that transforms the arc routing problem over sparse graph into vehicle routing problem over sparse graph with some illustration examples about the techniques. In Chapter 4, we formulate new models of the capacitated arc routing problems over sparse graphs. Exact methods and adapted near-optimal methods are developed to solve the problem. Computational experiments and numerical results show the effectiveness of the built algorithms. A survey about robust optimization is introduced in Chapter 5 where we present the capacitated arc routing problem over sparse graph under travel costs uncertainty. In other terms, we
study the robust multi-scenario capacitated arc routing problem over sparse graph where the uncertain travel costs are represented in a set of scenarios. Adapted algorithms give promising numerical results in the end of Chapter 5.
Chapter 2

Node and Arc Routing Problems

Introduction

Routing problems occupy a very wide place in the domain of combinatorial optimization and integer programming. This is due to the importance of their applications and practical contexts in the real life. Among the different variants of these problems, the most studied and concerned in the literature are the vehicle routing problems also called Node routing problems. In the opposite, Arc Routing Problems have been less studied during the last decades, and start to be more and more considered by the routing research community. In this chapter, we present a literature review about these two types of routing problems, and we show several algorithms and approaches which have been built and used to solve each of these two problems and their most known variants.

2.1 Node routing problem

The node routing problem is the most studied and well developed in the research. The seminal version of the node routing problem is due to the traveling salesman problem (TSP) where a salesman has to visit a number of customers, where every customer is serviced exactly once, before returning to the point of departure, by minimizing the overall distance traveled. Later, the multiple TSP appeared, known as Vehicles Routing Problem (VRP),
where instead of one salesman as in the simple TSP, there are \( m \) vehicles, all located in one point known as a depot (origin), in which each vehicle must serve a number of customers meeting the same condition of departure and return to the depot with minimizing the overall distance traveled by all vehicles. Introduced in 1959 \[51\], VRP has become a basic problem in distribution management. One can notice that the most extensively studied variant of the node routing problem is the capacitated vehicle routing problem (CVRP). The basic capacitated vehicle routing problem (CVRP) can be described in the following way. We are given a set of homogeneous vehicles each of capacity \( Q \), located at a central depot and a set of customers with known locations and demands to be satisfied by deliveries from the central depot. Each vehicle route must start and end at the central depot and the total customer demand satisfied by deliveries on each route must not exceed the vehicle capacity \( Q \). The objective is to determine a set of routes for the vehicles that will minimize the total cost. The total cost is usually proportional to the total distance traveled if the number of vehicles is fixed and may also include an additional term proportional to the number of vehicles used if the number of routes may vary \[140\]. The term “capacitated” stands for each customer demand to meet regarding the capacity of each vehicle. The vehicle routing problem has many other variants due to its numerous applications such as the multi-depot VRP and the VRP with heterogeneous fleet of vehicles, where the capacities of the vehicles are different.

Concerning the CVRP and from a graph point of view, it can be represented as follows:

Consider an undirected graph \( G = (V, E) \) where \( V = \{0, 1, \ldots, n\} \) is the set of vertices where the customers are located in addition to the depot, and \( E = \{(i, j) | i, j \in V, i \neq j\} \) is the set of edges. Let \( c_{ij} \in \mathbb{R}_+ \) be the cost of the edge \((i, j)\) in \( E \) and \( d_i \) be the demand associated to the customer \( i \in V \). A fleet of identical homogeneous vehicles is located at the depot where each is of unique capacity \( Q \). Each customer has to be served by one single vehicle. The vehicle routing problem is an NP-hard combinatorial problem \[155\].
2.1. NODE ROUTING PROBLEM

2.1.1 Mathematical formulation of the capacitated vehicle routing problem

The capacitated vehicle routing problems can be described by different formulations modeling the capacitated vehicle routing problem such as those presented in [1, 27, 38, 109, 113]. In the following we present the integer linear programming of the CVRP and the formulation of the sparse CVRP.

Mathematical model of CVRP

Let $G = (V, E)$ be a complete undirected graph where $V$ is the set of nodes and $E$ is the set of edges. Let $c_e > 0$ be the cost associated to each edge $e = \{i, j\}$. Let the node 0 denote the depot where a fleet of identical vehicles, each of capacity $Q > 0$, are located. Each customer $i$ has a required demand $0 < q_i \leq Q$ has to be served by one and only one single vehicle respecting the condition of not violating its capacity. The objective is to determine a set of vehicle route of a minimum cost where each vehicle departs from the depot and returns back to it upon finishing its service. Let $R$ be a subset of $V \setminus \{0\}$ and let $q(R) = \sum_{i \in R} q_i$ be the total demand of this subset. Denote by $k(R)$ the minimum number of vehicles needed to serve the customers in $R$.

For each edge $e \in E$, define the integer variable $x_e$ representing the number of times the edge $e$ is traversed by the vehicle.

\[
\min \sum_{e \in E} c_e x_e \tag{2.1}
\]

subject to:

\[
\sum_{e \in \delta(i)} x_e = 2 \quad i \in V \setminus \{0\} \tag{2.2}
\]

\[
\sum_{e \in \delta(R)} x_e \geq 2k(R) \quad R \subseteq V \setminus \{0\}, |R| \geq 2 \tag{2.3}
\]

\[
x_e \in \{0, 1\} \quad e \in \delta(0) \tag{2.4}
\]

\[
x_e \in \{0, 1, 2\} \quad e \notin \delta(0) \tag{2.5}
\]

A mathematical formulation of CVRP
The objective function (2.1) aims at minimizing the total costs. Constraints (2.2) show that the customers are visited exactly once where $\delta(i)$ is the set of arcs having $i$ as one of their extremities. The capacity inequalities (2.3) ensure that a minimum number $k(R)$ vehicles enter and leave $R$ imposing that the routes are connected where $\delta(R)$ denotes the set of arcs having one extremity in $R$ and the another one in $V \setminus R$. Constraints (2.4) and (2.5) are integrality conditions.

This is the standard mathematical modeling of the standard CVRP. However, there is no obvious capacity constraints. Other formulations that encounter commodity flow variables have been introduced later. For more details about this, refer to [155] and [134].

In the following, we aim at studying the capacitated arc routing problem on sparse graphs. Without loss of generality, we call also the CARP on sparse graph as Sparse CARP. we introduce in the following part a brief section about the sparse capacitated vehicle routing problem called as “Graphical CVRP”.

### 2.1.2 Sparse CVRP

Starting from the sparse variant of the traveling salesman problem known as graphical TSP [128], a sparse CVRP can be considered as a generalization of it. Dealing with the graphical TSP has been done by transforming it into the classical TSP. This can be easily done over the sparse CVRP although it would lead to a huge increase in the number of edges. A mathematical formulation of the sparse CVRP introduced by Fleischmann [70] is given below:

\[
\min \sum_{e \in E} c_e x_e \quad \text{(2.6)}
\]

subject to:

\[
\sum_{e \in \delta(i)} x_e \geq 2 \quad \text{and even} \quad i \in V \quad \text{(2.7)}
\]

\[
\sum_{e \in \delta(R)} x_e \geq 2 \left\lceil \frac{q(R)}{Q} \right\rceil \quad R \subset V \quad \text{(2.8)}
\]
The objective function \((2.6)\) aims at minimizing the total transportation costs. Constraints \((2.7)\) show that customers are visited can be visited more than once but an even number of times. Constraints \((2.8)\) ensure that a minimum number of \(\left\lceil \frac{n(R)}{Q} \right\rceil\) vehicles enter and leave \(R\). Decision variables constraints are given in Constraints \((2.9)\).

This formulation has a drawback as pointed out by [70] that there exist integer solutions which do not represent feasible solutions to the sparse CVRP. Moreover, showing that an integer solution of the formulation is also a feasible solution of the problem is an NP-hard problem. In addition to that, the sparsity and the capacity do not appear obviously in this formulation though it is a mathematical formulation of a sparse CVRP.

In [128], the author has developed a research work on the sparse vehicle routing problem and the Steiner-\(m\)TSPTW while keeping them defined over their original sparse graphs without transforming into their complete ones. He was the first who studied such problems. As presented in his work, a sparse model should allow the presence of non-required vertices where the required vertices must be visited at least once to do the service and the non-required ones may be visited if necessary. Working on the original Steiner-\(m\)TSPTW on its sparse graph rather than converting it into its standard form over the complete graph was motivated by the following reasons:

1. Considering the original sparse graph would save some time as finding the shortest paths between each pair of vertices as well as between the depot and each vertex would be time-consuming especially in the case of large graphs.

2. Such transformation from sparse graph into complete one is valid only if there exists a high correlation between the cost and the time i.e. the cheapest path is also the quickest path which is not always the case. Moreover, if such a correlation does not exist, there will be many shortest paths and to choose one of them is risky in the means of not losing the optimal solution. To avoid this risk, it is required to compute all the Pareto-optimal paths between every two customers and this is hard to be implemented and would be again time-consuming.
3. Applying the column generation method to price the augmented graph
is easily done if time-cost correlation is attained. Otherwise, it is not.

Due to all these motivations, the problem was considered directly on its
original sparse graph, and it was explored and studied from two different per-
spectives. The first one is done by considering the underlying graph where
a subset of vertices are non-required just like the case in real life, and the
second perspective is done by building a well-adapted algorithm for the prob-
lem where a column generation method has shown its efficiency. Moreover, a
sparse VRP+1D which is a sparse vehicle routing problem with single dead-
line not of Steiner type is defined in [128]. This is because there are no
non-required vertices. It is not of graphical type as every customer must be
visited and serviced exactly once by one only single vehicle, in addition to
the condition that each edge cannot be traversed more than one time. A
MILP formulation of the sparse VRP+1D is given as follows [128].

Let $A$ be the set of duplicated arcs i.e. each edge $\{i, j\} \in E$ is duplicated
into two arcs $(i, j)$ and $(j, i)$. Denote $\delta^+(i)$ is the set of arcs leaving $i$ and
$\delta^-(i)$ is the set of arcs entering it. Let $y_{ij}$ be a binary variable equal to 1
if and only if the arc $(i, j)$ is traversed and 0 otherwise, and $f_{ij}$ be a non-
negative variable indicating the time elapsed once the vehicle reaches the
vertex $j$. $t_{ij}$ indicates the traversal time, $T$ is the deadline and $s_i$ a constant
indicating the service time of vertex $i$. The MILP formulation of the sparse
VRP+1D is given in the following:

\[
\min \sum_{(i, j) \in A} t_{ij} y_{ij} \tag{2.10}
\]

subject to:

\[
y(\delta^+(i)) = 1 \quad i \in V \setminus \{0\} \tag{2.11}
\]

\[
y(\delta^-(i)) = 1 \quad i \in V \setminus \{0\} \tag{2.12}
\]

\[
f(\delta^+(i)) = f(\delta^-(i)) + \sum_{j \in \delta^+(i)} t_{ij} y_{ij} + s_i \quad i \in V \setminus \{0\} \tag{2.13}
\]

\[
f_{ij} \leq Ty_{ij} \quad (i, j) \in A \tag{2.14}
\]
y_{ij} \in \{0, 1\} \quad (i, j) \in A \quad (2.15)

f_{ij} \in \mathbb{R}_+ \quad (i, j) \in A \quad (2.16)

Constraints (2.11) and (3.10) ensure that exactly one vehicle departs from each customer as well as exactly one vehicle arrives at each customer. Time accumulation while building the routes and sub-tour elimination are attained in constraints (3.11). Constraints (3.12) show the respect of the deadline without being violated by any vehicle. The last constraints are decision variable constraints.

Note that \( y(\delta^+(0)) \) indicates the number of vehicles used in the solution. Therefore adding constraints that bound the number of vehicles can strengthen the formulation. These constraints are:

\[
y(\delta^+(0)) \leq K_{\text{max}} \\
y(\delta^+(0)) \geq K_{\text{min}}
\]

(2.17) \hspace{1cm} (2.18)

Where \( K_{\text{max}} \) and \( K_{\text{min}} \) are an upper bound and a lower bound of the number of vehicles respectively. On the other hand, the formulation can be strengthened by adding other constraints. For example, replace constraints (3.12) by the following one:

\[
(t_0 + t_{ij}) y_{ij} \leq f_{ij} \leq (T - t_{j0}) y_{ij} \quad (i, j) \in A
\]

(2.19)

As noticed here, these bounds are not trivial as those given in (3.12), and it is clear that if the vehicle arrives at vertex \( j \) from \( i \), it must have already traveled from the depot to \( i \), and it still has to travel from \( j \) to the depot. Also, the constraints that impose the unique traversal of incoming and outgoing edges for each required vertex build up the formulation. These constraints are given by:

\[
y_{ij} + y_{ji} \leq 1 \quad \text{for each } \{i, j\} \in E
\]

(2.20)

According to [128], the sparse VRP+1D was not only considered as above, but also by a set partitioning approach where the pricing problem was studied in two different cases; the first where only elementary paths are permitted and the second where non-elementary paths are generated.
2.2 Arc routing problem

The arc routing problem finds its origin in the famous Königsberg bridge problem proposed by Leonhard Euler for which he presented a solution to it in 1735 [139]. The problem is given as follows. Given the seven bridges of the city of Königsberg, is it possible to go for a walk, starting and ending the same place and passing each of the bridges exactly once? Euler had represented this problem from a math point of view and he stated it as given below:

Problem 2.2.1. Given a connected graph \( G = (V,E) \). Find a tour that visits every edge in \( E \) exactly once, or determine that no such tour exists.

This problem is known as Euler tour problem, and he showed that such a tour exists if and only if every node in \( G \) has an even degree. An algorithm for constructing Euler tour had been built by Fleury [72].

Following the problem presented by Euler, arc routing problems consist of determining a least cost traversal of some arcs or edges of a graph, subject to side constraints. In other words, it refers to routing problems where the key service is to cover arcs on a transportation network [7, 66]. The arc routing problems are encountered in a variety of practical situations where each street segment must be covered in its entirety. Such of these practical contexts are garbage collection, school bus routing, mail delivery, road maintenance, salt gritting and meter reading, ... Moreover, we mention that there are some problems that do not occur on road networks, but they can also be modeled as arc routing problems. For instance, robot exploration, analyzing interactive system and web site usability (e.g. [133]), and analyzing DNA e.g. ([126]) are all examples of such problems.

In this chapter, we present a brief survey of the arc routing problems and its variants as it was studied in the literature, and we focus on the Capacitated Arc Routing Problem (CARP) by introducing a review and the different formulations and several heuristic algorithms.

Remark 2.2.1. All variants of the routing problems which are presented in the two following sections are of one only vehicle i.e. they are considered of the type Single-Vehicle Arc Routing Problems.
2.2. Arc Routing Problem

2.2.1 The Chinese Postman Problem (CPP)

The Chinese Postman Problem has been documented by [89]. The problem is to find the shortest walking distance for a mailman who has to cover his assigned segment before returning back to the post office. From a graph theory point of view, this problem can be defined as follows. In all what follows, let $G = (V, E \cup A)$ be a graph where $V$ denotes the set of vertices, $E$ denotes the set of (undirected) edges and $A$ denotes the set of (directed) arcs. In general, it is always assumed that $G$ is strongly connected i.e., it is always possible to reach any vertex from any other vertex. Let $c_{ij}$ be the associated cost to the edge or the arc $(v_i, v_j)$. The CPP consists of determining a least-cost traversal of all the edges and the arcs of $G$.

Some different variants of CPP are the undirected chinese postman problem (UCPP) [60, 61], the directed chinese postman problem (DCPP) [61, 127, 14], the mixed chinese postman problem (MCPP) [129, 39, 112, 18, 164], the windy postman problem (WPP) [123, 90], the hierarchical chinese postman problem (HCPP) [59].

2.2.2 The Rural Postman Problem (RPP)

The Rural Postman Problem (RPP) is simply the CPP arising in rural areas. There is a set of streets that must be serviced and another set that do not need any service but may be used for traveling [146], i.e. there is a subset $R \subseteq E \cup A$ that contains the edges requiring a service which are called required edges. In general, $R$ is given by $E_r \cup A_r$, where $E_r$ and $A_r$ are respectively the sets of required edges and arcs of $G$. Both the directed and the undirected RPP were proven to be NP-hard in [114], but it can be solved in a polynomial time in the case where the graph induced by the required edges is connected. There are several variants of the RPP, one may list the undirected rural postman problem (URPP) [127, 41, 47, 81, 93, 94], the directed rural postman problem (DRPP) [40, 42], the mixed rural postman problem (MRPP) [49, 49], The periodic rural postman problem (PRPP) [82], The rural postman problem with deadline classes (RPPDC) [118].

2.2.3 Multi-vehicle Arc Routing Problems

Throughout this section, the most important multiple-vehicle ARPs in the literature will be surveyed. Note that the majority of real life ARPs are of
this type which involve more than one vehicle.

**K-Chinese Postman Problem (K-CPP)**

This is a variant of the CPP where $K$-postmen are involved in the routing problem instead of one single postman. These $K$-postmen depart from the same depot and do the required service where the objective is to find $K$ minimum cost routes. Knowing that this problem can be solved in a polynomial time if the input graph $G$ is even \[132\], but it is NP-hard in its general case to the contrary to its directed and undirected versions ($K$-UCPP and $K$-DCPP) which are always solvable in polynomial time \[166\].

**Min-Max K-Chinese Postman Problem (MM K-CPP)**

The MM $K$-CPP is a variant of the routing problems which requires not only to find $K$ minimum cost routes on $E$, but also to find them in such a way that minimizes the maximum cost among all feasible sets of $K$ routes. This goal is attained by a $\min - \max$ objective. This problem was first mentioned in \[78\]. Later, heuristics and tabu search algorithm in addition to exact solution method based on a branch-and-cut approach have been developed.

### 2.2.4 Capacitated Chinese Postman Problem (CCPP)

This problem has been defined by Christofides \[37\] in 1973 and was shown to be strongly NP-hard by Golden and Wong \[84\]. From a graph theory point of view, this problem is defined on an undirected graph $G = (V, E)$ where each edge $e$ has a demand $d_e > 0$ to be serviced, and a fleet of $K$-identical vehicles are located at a single depot with each of capacity $Q$. Each edge must be serviced by exactly one vehicle, but can be traversed without service by an unlimited number of times, and always the capacity of each vehicle must not be violated. The objective is to find a minimum cost set of vehicle routes where each vehicle has to depart from the depot and return back to it while ending the service, such that each required edge is serviced exactly one time by one single vehicle.

### 2.2.5 Capacitated Arc Routing Problem (CARP)

One of the most important variant of the arc routing problem is the capacitated arc routing problem. The Capacitated Arc Routing Problem (CARP)
was defined in [84] and they have proposed the first formulation of it. The CARP is NP-hard [58]. Concerning the resolution methods, branch-and-bound was applied to small instances in [96] and in [103]. Later, other methods such as the cutting plane algorithm have been applied in [12]. This problem can be considered as a generalization of the CCPP, RPP and CPP detailed above.

We recall the definition of the capacitated arc routing problem CARP can be stated as follows:

**Definition 2.2.1.** Given a connected undirected graph $G = (V, E, C, D)$, where $C$ is a cost matrix and $Q$ is a demand matrix, and given a number of identical vehicles each with capacity $Q$ (where $Q \geq \max d_e, e \in E$). Find a number of tours such that:

1. Each arc with positive demand is serviced by exactly one vehicle.
2. The sum of demands of those arcs serviced by each vehicle does not exceed $Q$.
3. The total cost of the tours is minimized.

**Solution approaches of CARP**

Dealing with the capacitated arc routing problem have been done by different approaches. Any optimization problem can be solved by using one of the following four distinct classes: approximation algorithms, problem specific heuristics, meta heuristics, and exact algorithms. However, CARP has not been studied by the approximation algorithms except as a particular case of the capacitated general routing problem.

**Exact Algorithms for solving CARP**

Exact algorithms give optimal solutions of the studied problem. Knowing that CARP is NP-hard problem, this type of algorithms give the optimal solution but in an exponential worst running time. The most popular exact algorithms for solving CARP are branch and bound and cutting plane algorithms and their variants. These two methods depend on the integer linear programming that represents a mathematical formulation of the problem. This formulation is given by an objective function that aims at minimizing or maximizing some function over some decision variables subject to a set
of constraints that should be respected by any feasible solution. The first integer linear programming for the CARP was presented in [84]. Another formulation was given in [12].

Branch and Bound Algorithm

This method constitutes of building a tree where the solutions are enumerated in addition to the use of lower bounds and solution values to restrict the exploration area over the most promising part of the tree. An algorithm of branch and bound was developed in [96] to solve the CARP and they could find the optimal solution of instances whose required edges were between 15 and 50.

Cutting Plane Algorithm

In this algorithm an LP relaxation of the problem is solved. Moreover, searching for valid inequalities to cut off the current fractional solution conserving the integer solutions. The different methods of constructing the valid inequalities lead to different variants of this algorithm. There is no certainty that assures the possibility of finding an integer solution due to the fact that not all the valid inequalities are known, and even if they are known, there is at least one class of these inequalities cannot be generated in a polynomial time as CARP is basically NP-hard. Some classes of valid inequalities were introduced in [11] and later were used to present a cutting plane algorithm of CARP in [12]. This algorithm allows to reach for the best existing lower bound for all instances.

Branch and Cut Algorithm

This is a combination of the branch and bound algorithm with cutting planes algorithm. Valid inequalities could be added to the problem before branching in each node of the search tree. Adding cuts complicates the calculation of the lower bound, thus a trade-off between adding cuts and branching takes place.

Branch, Cut and Price Algorithm

This is a combination of the branch and bound, cutting plane and column generation algorithms. This algorithm has been applied to the vehicle routing
problem (VRP) but, to the most of our knowledge, not over CARP.

Approximate algorithms for solving CARP

The different heuristics that were used for solving the CARP are construct-strike algorithm, augment-merge algorithm, path-scanning algorithm, parallel-insert algorithm, Modified construct-strike algorithm, modified path-scanning algorithm and augment-insert algorithm. In this part, we give a brief idea about each of these heuristics.

**Construct-Strike Algorithm:** This heuristic is constructed basically to solve the capacitated chinese postman problem (CCPP) and it was developed in [37]. The algorithm is composed into two steps; the first is accomplished by constructing the tours respecting the capacity of the vehicle not to be violated and keeping the graph connected upon deleting the required edges. In the second step, the remaining graph is made even and artificial edges of a minimum cost perfect matching between the odd nodes in the graph are added in order to make the degree of the depot at least two. When all the required edges are covered by the tour, the algorithm stops. This algorithm can be adapted to the CARP by adding artificial edges to make the graph which is induced by the required edges connected.

**Augment-Merge Algorithm:** This is the first heuristic which has been developed to solve the CARP and it constitutes of two phases. The cycles are constructed in the first phase in such a way that each cycle contains one only required edge. Second, if demand on a smaller cycle can be moved to a larger cycle, without changing the route for that cycle, the demand is moved and the smaller cycle is discarded. In the second phase if merging the cycles does not cause the violation of the capacity constraints and leads to get a solution of lower cost, then the cycles are merged. This algorithm was first suggested in [84], and then explained later in [8].

**Path-Scanning Algorithm:** The Path-Scanning Algorithm is the second heuristic made to solve the CARP [8]. In this algorithm, the tours are built one by one in a repeated way where the next edge is selected according to one of the following criteria. Let \((i, j)\) be a required edge.

1. The distance \(c_{ij}\) per unit demand is minimized.
2. The distance \(c_{ij}\) per unit demand is maximized.
3. The distance from the node $j$ to the depot is minimized.

4. The distance from the node $j$ to the depot is maximized.

5. If the vehicle is less than half-full, the distance from $j$ to the depot is maximized, otherwise this distance is minimized.

Note that when the vehicle capacity cannot be loaded any more, a shortest path taking the vehicle back to the depot is used.

**Parallel-Insert Algorithm:** The first step of this algorithm is to estimate how many tours are needed where the building of each tour starts with an edge chosen to be far from the depot. The next step is to continue building the tours in parallel. Starting from a given edge, identify the tour in which this edge is to be inserted minimizing the insertion cost. Now, starting from a tour, determine which of the remaining edges should be inserted in this tour. Maybe not all required edges are served, then extra initial tours are needed. This heuristic is presented in [34] in which the authors aim at not only minimizing the total cost, but also at constructing routes that are well balanced.

**Modified Construct-Strike Algorithm:** This algorithm is a new mixed version of the two heuristics, the construct-strike algorithm and the path-scanning algorithm [130]. Here the tours are constructed by choosing the edge that maximize the least quantity path back to the Depot in a repeated way. Now, after removing the required edges of the constructed tours, if the remaining graph is disconnected, a minimum spanning Tree between the components is constructed before the graph is made even in the same way as in the original algorithm.

**Augment-Insert Algorithm:** In this algorithm the required edges are considered in decreasing order with respect to the distance from the depot. For each edge, the shortest tour containing the edge is built and the tour is augmented with the required edges already on the tour, and the serviced edges are removed from the graph. This procedure is repeated until all the required edges are disconnected from the depot. Ending this step, all edges are restored and the process is repeated, except that now the augmentation may include edges that are not already on the tour. The cost of this insertion may be depending on a parameter, which in turn is changed over several runs of the algorithm, and the best solution is chosen [131].
2.2. ARC ROUTING PROBLEM

Meta-heuristics for solving CARP

Meta-heuristics are constructed to work on optimization problems for which we can define a cost structure and a neighborhood \[152\]. Identifying the solution cost structure as well as defining the neighborhood structure lead to apply the meta-heuristics over specific problems such as the CARP. In the following, we present a review about the meta-heuristic algorithms that have been developed to solve the CARP.

\textbf{Simulated Annealing Algorithm:} This algorithm is developed in \[62\] for a winter gritting problem which is modeled as a CARP in addition to some complicated constraints. In this algorithm, any solution in the neighborhood is picked and if the solution is worse than that in the hand, then one shifts towards it with a probability \( p \), while if it is better then one shifts towards it always. The value of the probability \( p \) decreases through the process according to some specified cooling scheme.

\textbf{Tabu Search Algorithm:} In this algorithm there is a tabu list that contains set of moves or set of solution which are forbidden and not allowed to be performed or taken. This list has a determined size and is updated at each stage of the algorithm. We aim at finding the best solution in the constructed neighborhood with the condition that this solution is not tabu. Considering the CARP, different tabu search algorithms are developed to solve it. Starting from the first Tabu search algorithm for CARP developed in \[94\] up to the one presented in \[5\] which was made for the multi-depot version of the problem, reaching to the algorithm in \[87\] where a combination between tabu search and a so-called scatter search took place and this is known as a tabu scatter search for the CARP.

\textbf{Genetic Algorithms:} Starting from a generation which is a set of solutions, then applying crossover between two solutions to construct two new ones. Finally we get a new set of solutions. This set is added to the initial one and define a new generation to be the best half between these two sets. This procedure is done several number of times. In \[105\] the authors present a generic algorithm for the CARP. A variation of a generic algorithm, which
they refer to as a memetic algorithm for the CARP is given in LPR04. Here the crossover is performed on a giant tour which is split subsequently.

**Ant Colony System**: An ant colony heuristic is constructed to solve the CARP by [57]. Considering the same procedure followed by the ants to get their food by secreting the Pheromone on the ground depending on the distance between them and their food so that paths with large amounts of pheromone are more likely to be used by other ants.

**Local Search Algorithms**: In guided local search the objective is improved in every iteration until a local optimum is reached. In [31] the authors presented a guided local search algorithm for the CARP. In their algorithm the objective to be minimized is the dead-heading distance i.e. the distance of traversing the edges without serving them, where the distance of each edge is penalized according to some function, which is adjusted throughout the algorithm.

### 2.2.6 CARP Applications

The importance of the capacitated arc routing problem rises in its real life contexts and applications. Here we present some of the applications that were treated throughout the literature.

#### Refuse Collection

This problem is a real CARP as it requires servicing along the streets in addition to the capacity of the vehicles which is represented by the amount of the refuse in each truck and by the number of working hours. This problem is presented by [26]. The author introduces an analysis of the routing of sanitation vehicles and aims at minimizing the operating cost. The presented model considers not only the routing part of the problem, but also the whole phase of collecting and storing the data for easy access by the routing algorithms, the generation of the networks and the final reporting. Constructing one giant tour handles the routing problem as presented similarly in [28]. This tour is then partitioned into a set of relatively identical tours i.e. tours...
of relatively same size respecting the capacity constraints.

Another consideration of this application has been done by [120]. The authors focus on minimizing the maximal load of the workers when the collection of refuse is done a fixed number of times a week. Hence, they are concerned with the work schedule of the refuse collection problem, and not so much with the routing of the vehicles. They give some interesting discussions on how the problem changes when considering a system where the refuse is divided into types (paper, organic refuse etc.) that need to be handled separately.

**Electric Meter Reading**

This problem arises in the countries that have municipal public service agencies which periodically have readers going from house to house to collect data for billing purposes. The electric meter reading was considered in [58]. The readers are transported to the beginning of their route, work for a number of hours and are free to leave afterward. The companies aim at minimizing the labor cost and thus at minimizing the dead-heading time and thereby the number of workers. As the problem involves required services which are the streets and a limited time of work by each reader, then this problem can be modeled as CARP. However, the solution of this problem is in the form of paths and not cycles due to the absence of the connections to and from the Depot node. The problem is solved by constructing a giant tour which is in turn split, using simple forward splitting, into small tours corresponding to one reader.

**Airline Scheduling**

Many parts of airline scheduling can also be modeled as CARP. This includes the scheduling of planes to flight legs and the scheduling of crew. When doing this, each node in the graph corresponds to a destination and an arc in the graph corresponds to a flight leg. That way the problem is modeled as a directed arc routing problem. Many extra constraints must be taken into account when considering airline problems, including that the fleet of planes is not homogeneous and a complicated set of union rules for the crew. The costs in airline scheduling are high compared to the other applications that we have considered and so a reduction of even few percent in operation cost
can improve revenue by millions of dollars. In [6], [43] and [10], the authors consider the crew scheduling problem and describe ways to model and solve the problem using a subproblem method. In [98], a primal-dual subproblem simplex which extensively speed up the solution of the crew planning problem is presented. In [136] and [137], a stochastic model of the airline operation is given. The authors consider different ways of disrupting the operation and suggest various methods for recovering the schedules. They describe a simulation of airline operation called SimAir to simulate the schedules and the influence of the different recovery procedures on the longer term.

Winter Gritting

In [63], the authors consider the problem of spreading a de-icing agent on roads to prevent them from becoming dangerously slippery. This is often referred to as Winter Gritting and can be modeled as a CARP, since the problem is to service a set of streets in a network. The streets are serviced by capacitated vehicles. The capacity constraint is on the amount of de-icing material the truck can keep and the time the whole gritting takes. The objective of the problem is to service the streets with a minimum cost with respect to the constraints just mentioned. In contrast to the case of street sweeping, when considering winter gritting each street can be serviced in both directions at once with the exception of highways with multiple lanes in each direction. This means that the relative simple solution method cannot be applied to this situation. The authors do not go into details about how they choose to solve the problem. In stead they give an interesting discussion on how efficiency of the constructed solutions can be measured, and how the type of network (rural versus urban area) influences this measure. In [62] the authors also considered the routing of winter gritting vehicles. The author considers a setup where the streets are partitioned into a number of categories, where those in category 1 must be treated within two hours of call-out, those in category 2 must be treated within four hours of call-out etc. This means that the problem becomes time constrained. These time constraints could be handled as wide time windows or, as is the case in the paper, by considering the categories separately. As before this winter gritting problem is also constrained by the capacity of the vehicles and thereby by the street length they can service. In addition the problem here is a multiple depot problem, i.e. there are several possibilities as to where the vehicles can be stationed and loaded for gritting. The network is partitioned into elementary
2.2. ARC ROUTING PROBLEM

cycles at every node with a degree of more than two and a cycle-node network is then constructed containing a node for each of these cycles. Two nodes in the cycle-node network are connected by an edge if the corresponding cycles in the original network have a node in common. The location of the Depots are now fixed, and tours are constructed by making rooted trees in the cycle-node network. If necessary tours are merged. Finally the tours are used as a starting solution for a simulated annealing algorithm where the time and capacity constraints are relaxed and penalized in the objective function. In [32] the problem of spreading salt for winter gritting is also considered. The authors mainly consider a setup with multiple Depots and are interested in design of districts for the gritting operation. A district is a geographical area associated with a single Depot, and are constructed on a long-term planning level, whereas the routing of the vehicles within each district can be modified on a short term level. The authors assume that the location of the Depots are fixed and describes an algorithm to construct the districts in the following way. The network is first partitioned into elementary cycles which are in turn assigned to a depot as a whole. The assignment of cycles to Depots (and thereby to districts) is done such that cycles that are much closer to one depot than to any other depot are assigned to that depot, and so on, while making sure that the borders between districts are relatively clear. Finally, each district is considered in turn and the routing of the gritters are done using the augment-insert heuristic [131].

Street Sweeping

In [28] the problem of routing street sweepers in the cities is studied. This problem is clearly a variation of the CARP since it can be modeled as a graph where the links (edges and/or arcs) must be serviced. The objective here is to minimize the total travel cost. The application studied by the authors has the restriction that a street cannot be swept during parking hours. This results in a problem in which each link has a time window associated in which the service of that link must take place. We will refer to this problem as the CARP with Time Windows (CARP-TW). In [29] the authors give a detailed description of a computer system to solve the problem. Their algorithm starts by solving a transportation problem to decide which links must be traversed an extra time without being serviced. Then, to take care of further restrictions such as U-turns that must be penalized, an assignment problem is solved, and finally an Euler tour is constructed. This algorithm is finally
put in a larger framework where capacity constraints and time windows are taken care of. It may be noted, that the time windows are handled in a serial manner and are not included in the solution algorithm itself. Street sweeping in rural areas are considered by [65]. The authors argue that unlike in urban areas, there are very few one-way streets in rural areas. Also, because of the type of roads it is possible to perform U-turns in these environments, and because of lack of parking restrictions one does not have to worry about time windows on the streets. For these reasons, the routing of street sweepers become easier in rural areas. Furthermore, because the area is often larger there will be several tip sites and a good routing algorithm must take this into consideration. The authors note that because each road can be represented by two arcs pointing in opposite directions, a tour can always be constructed that services both sides of a street on the same tour. In a sense this is done by noting that a tour can be considered a tree on the street network. The tour can then be constructed by traveling along the tree while always choosing a left must turn if possible, and if not by making a U-turn. Modifications are then made to deal with dead-ends and the likes, and the tip site is chosen as the one nearest to the end of the tour.

2.2.7 Different Variants of CARP

In this section, we consider variations of the classical CARP. Each of the variations considered reflects situations occurring in real life applications. The variants of the CARP are distinguished according to the variation occurring in one of the characters of the problem that lead to a variation in the classical CARP.

Multi Depot CARP

As mentioned in its name, this variant of CARP includes several depots in which the fleet of vehicles is located and from where the tours must begin and end. This is different from the classical CARP where there exists one single depot. In this variant, vehicles may return to a depot different from that where they departed as the importance is to return back to any depot. Moreover, vehicles locating at a depot may have another capacity than those located at another depot.

The multi depot CARP; MD-CARP, appears clearly in the applications of the classical CARP while considering large areas as street sweeping and
2.2. ARC ROUTING PROBLEM

refuse collection, but these problems are usually decomposed into classical single depot ones according to some geographical criteria as towns or neighborhoods. The division of the MD-CARP was first considered in [32] and [33]. The smaller areas obtained by some division were called districts that represent a long-term planning level, whereas the construction of vehicle tours are on a short-term planning level. Each district should be a connected geographical area with clear boundaries, and the depot of each one is preferred to be locating as central. The streets that naturally belong together, such as the streets in a small neighborhood are contained in one district. These districts should be almost identical i.e. having similar size. Concerning the method of resolution of this variant, a first approach is to partition the network into small units of edges that should belong to the same district making the network Eulerian and making a cycle decomposition of the resulting network, then these units are merged heuristically in order to build the districts. Two different strategies for merging the units were used. The first is to simply assign each unit to the nearest Depot and the second is based on minimizing the lower bound of the required number of vehicles needed to service the edges of the resulting districts.

Another opposite approach for such a problem was considered in [5] where the districts are built according to a routing strategy and not as long term decisions. In other terms, the tours are constructed as a first step, and the districts are then formed as the streets serviced by vehicles emerging from a specific Depot. The problem is solved by transforming it into an arc-constrained capacitated minimum spanning tree problem (CMST) after constructing the giant tour, then heuristically and the solution is then improved by local search. The obtained tours are then developed by a simple route optimization procedure and the solution of the multi-depot problem is derived from the solution that is found for the CMST.

Another variant of the MD-CARP is the CARP with intermediate facilities known to be as CARP-IF. This problem was considered in [80], and it is defined with one single depot but with a set of nodes known as intermediate facilities. These intermediate facilities are the locations where any vehicle may recharge its capacity. For example, they can be dump sites for refuse or storage halls for salt for winter gritting, with conserving the condition for the
vehicles to start and end at the depot. Two lower bounds and two heuristics were suggested by the authors for this problem. Concerning the lower bounds, the first is relatively tight RPP lower bound based on branch-and-cut to bound the CARP-IF as the RPP may be viewed as a particular case of CAEP-IF, and the second lower bound is a relaxation lower bound of an ILP formulation based on dead-heading variables. For the heuristics, the first is based on constructing an RPP-tour and splitting the tour in appropriate portions while connecting to the intermediate facilities, and the second one is based on solving the classical CARP in a modified network, transforming the solution to a CARP-IF solution in the original network and making some adjustments to restore feasibility.

**CARP with Mobile Depots**

In this version of CARP, only one vehicle discharge its load in the depot as for example in the case of refuse collection. In detail, there are two types of servicing vehicles where those of small capacity unload in one of those of large capacity which in turn move to the depot. In addition to the routing of each type of these vehicles, it is required to decide at what time two vehicles must meet at some node in order to perform this unload action. The authors in [71] solve the problem with a modified version of the variable neighborhood descent algorithm first presented in [95].

**CARP with Time Windows**

The CARP with time windows (CARPTW) is just the classical CARP with one additional requirement that imposes the services of the required arcs to be started within a determined interval or time window. In addition to costs, travel times are associated with all edges, and the tours must be constructed to respect the additional restriction, that all edges are serviced within their Time Window. Concerning the practical contexts of this version, flight legs in airline scheduling have a fixed departure time and can therefore be considered as having a time window of zero length. Street sweeping [28] has restrictions with respect to the time during which the sweeping may be performed. Moreover, routing of winter gritters [62] where some streets must be serviced within well determined number of hours is also an application of this variant. Two mathematical formulations of the problem are presented by [163] where
the first depends on constructing a node duplicated network on which the ILP model is built, and the second depends on a transformation to the equivalent node routing problem, the VRPTW. Moreover, two heuristics were mentioned in \[163\]. A path-scanning algorithm which chooses edges based on their time windows is presented and the preferable neighbor heuristic is suggested as a new heuristic.

### CARP on Directed or Mixed Graphs

Due to the fact that several real life applications of CARP consider one-way-streets and streets where the two sides must be serviced in parallel, the definition of the CARP over directed and mixed graphs is required. A Directed CARP (DCARP) has been considered in \[160\] in order to derive optimal solutions. The author presents valid inequalities and separation algorithms for an ILP formulation of this problem. However, the mixed CARP (MCARP) is considered extensively in \[13\], where three problem specific heuristics, augment-merge, path-scanning and ulusoyâ’s heuristic are improved and changed to fit the problem. Furthermore, the memetic algorithm in \[106\] is adapted to the MCARP. Finally, the authors give a supersparse LP formulation of the MCARP, which is used in a cutting plane algorithm to obtain strong lower bounds for the problem. Computational experiments show that the gap between the lower bound and their memetic algorithm is less than one percent for the test instances.

### CARP with Alternative Objective Functions

Usual objective functions of routing problems aim at minimizing the total traveled distance, but this is not the case in all the real life applications. There are some applications in which their corresponding objective functions aim at minimizing the number of vehicles used or the length of the longest tour.

A version of the CARP where the vehicles do not have the same capacity as well each vehicle includes a fixed cost if it is used is considered by \[156\]. In this version, the objective is not only to minimize the total travel cost, but also to minimize the total fixed cost incurred by the use of the vehicles. The problem is considered when the number of each vehicle type is unlimited and when this number is bounded. The author presents a heuristic formed
of two steps where the first constructs a giant tour and the second is to split
the tour by solving a shortest path problem which takes vehicle capacities
and costs into consideration.

The multi objective CARP [107, 108] is defined to be a classical CARP
with one more objective that is to minimize the makespan i.e. the length
of the longest tour. The authors present a generic algorithm for solving the
multi objective CARP which can be seen as a mix between the CARP and
the min-max $K$-chinese postman problem (MM $k$-CPP) which in turn can
be viewed as a CARP where the vehicle capacity is infinite. The goal is
to minimize the length of the longest tour[78]. The first heuristic of the
MM $k$-CPP presented in [78] is based on constructing a giant tour which
is subsequently partitioned into $k$ tours of roughly equal length. Another
extensive study of the problem is done by [2] where the author presents a
heuristic based on the augment-merge algorithm for the CARP along with
a new algorithm based on the cluster first-route second idea. A tabu search
algorithm is presented for the MM $k$-CPP in [3].

**Periodic CARP**

The periodic CARP (PCARP) is defined as the CARP where a long time
horizon is considered such that each required edge must be serviced more than
once. This is the case of refuse collection where each household is serviced
two or three times a week on a rolling schedule. Note that the problem
may require a minimum and maximum number of days between each service
of the same street. The problem is formulated mathematically in [44], and
three heuristics are also suggested for obtaining feasible solutions. In [106]
the authors suggest a generic algorithm for solving the problem. A scatter
search algorithm is suggested in [45].

**Stochastic CARP**

In this variant of CARP, the demand of the required edges is a random
variable as it is the case in refuse collection, mail delivery, and snow removal
where the exact demand is not known. The stochastic CARP (SCARP) was
first introduced in [74] and [76]. The problem was then studied concerning
the quality of solutions when the solutions are obtained with algorithms
for the classical deterministic CARP [75]. The authors examine how the
robustness of the solution changes when the deterministic problem is solved with a slightly smaller fictive vehicle capacity. A memetic algorithm for the SCARP, which is an extension of the algorithm suggested in [106] is presented [73]. The results obtained are compared to the results generated by algorithms for the classical CARP based on the average demand.

CARP with Vehicle/Site Dependencies

The problem is defined such that not all edges can be serviced or traversed by all types of vehicles [146]. In [9] the authors suggest a vehicle decomposition algorithm for solving an instance of this problem which they encountered in a refuse collection application. A solution procedure for the problem is given in [30]. This procedure depends on a composite approach consisting of an initial fleet mix generator, a mathematical programming procedure and a measure of goodness function.

2.2.8 Different Formulations of the CARP

In the CARP problem literature, many different formulations are Proposed. Excellent references are dedicated to the presentation of the CARP problem in which we mention here that in [64] and [15]. In this section we present four formulations of CARP problem Which are determined by the number of variables used. Throughout the following, we consider the following notations.

- $K$: the number of vehicles.
- $R$: the set of the required edges (or required arcs).
- $S$: a subset of the set of vertices of $G; V$.
- $\delta(S)$: a cut set of $S$ that contains incident edges to $S$ i.e. edges having one extremity in $S$ and another one in $V \setminus S$.
- $E(S)$: set of edges each having the two extremities in $S$.
- $A(S)$: set of directed edges (arcs) each having two extremities in $S$.
- $\delta_r(S), E_r(S)$ and $A_r(S)$: the restrictions of $\delta(S), E(S)$ and $A(S)$ respectively over $R$. 
• $\delta^+(S), \delta^-(S)$: the respective sets of arcs $(i,j)$ leaving and entering to $S$.

• $\delta^+_r(S), \delta^-_r(S)$: the respective restrictions of $\delta^+(S)$ and $\delta^-(S)$ over $R$ i.e. $\delta^+_r(S) = \delta^+(S) \cap R$ and $\delta^-_r(S) = \delta^-(S) \cap R$.

The next four formulations to be discussed are:

1. The “supersparse” formulation that contains $|E|$ variables.

2. The “sparse” formulation that contains $2K|E|$ variables.

3. The “sparse directed” formulation that contains $4K|E|$ variables.

4. The “dense” formulation that contains $2|E|^2$ variables.

**The “Sparse Directed” Formulation**

This formulation is based on a transformation of non-directed graph $G = (V,E)$ into a directed one. Each edge $e = \{v_i, v_j\} \in E$ of $G$ is presented by two arcs $a_1 = (v_i, v_j)$ and $a_2 = (v_j, v_i)$. $a_1$ and $a_2$ are evaluated by the same cost and the same demand of $e$. Only one arc between the two arcs that are derived from the same edge of the set $E_r$ must be serviced by only one vehicle. Two sets of binary variables are used in this formulation: $x^p_{ij} = 1$ if the vehicle $p$ traverses the arc $(v_i, v_j)$ and 0 otherwise, and $y^p_{ij} = 1$ if the vehicle $p$ services the arc $(v_i, v_j)$ and 0 otherwise.

\[
\text{SD-CARP} \quad \min \sum_{p=1}^{K} \sum_{e \in E} (x^p_{ij} + x^p_{ji})w_e \quad (2.21)
\]

subject to:

\[
x^p(\delta^+(v_i)) = x^p(\delta^-(v_i)) \quad \forall v_i \in V, p = 1, \ldots, K \quad (2.22)
\]

\[
\sum_{p=1}^{K} (y^p_{ij} + y^p_{ji}) = 1 \quad \forall e = \{v_i, v_j\} \in E_r \quad (2.23)
\]

\[
x_{ij}^p \geq y^p_{ij} \quad \forall e = \{v_i, v_j\} \in E_r; p = 1, \ldots, K \quad (2.24)
\]
This formulation was first proposed by [84] and then improved by [160]. It can be represented by the following linear model. Let $Q$ be the capacity of each vehicle, $d_e$ be the demand of the edge $e$ and $w_e$ its cost.

The objective function (2.21) aims at minimizing the total traveling cost. Constraints (2.22) assure that each vehicle must leave a node once it enters it. The fact that each edge is served one and only once in a single vehicle and in one direction is provided by constraints (2.23). Constraints (2.24) impose the passage of each vehicle by a given edge for the service. The capacity constraint is presented by the inequality (2.25) and the constraints of connectivity are provided by (2.27). The last two constraints are decision variable constraints.

To reinforce this formulation, the authors in [160] proposed two constraints. These constraints ensure the use of all vehicles. The second constraints are called “aggregated parity constraints” because the sum is done over all the vehicles by the following inequality:
The “Sparse” Formulation

This NP-hard problem is defined over a connected non-directed graph \( G = (V, E, C, D) \) representing the network, where \( V \) is the set of \( n \) nodes, \( E \) is the set of \( m \) edges, \( C \) is the matrix of costs and \( D \) is the matrix of demands. We suppose that there is one single depot node in the network, say 1. In this depot, there is a fleet of \( n_v \) identical vehicles, each of capacity \( Q \). We associate for every edge \( e \in E \ (e = (i, j); i \in N \text{ and } j \in N) \), a cost \( c_e = c_{ij} \geq 0 \) and a demand \( d_e = d_{ij} \geq 0 \). \( R \) is always the set of required edges with \(|R| = m_r\).

Note that, it is not necessary to traverse each non-required edge (those of null demand), that the total demand of each tour must not exceed the capacity of the vehicle associated to this tour, and that the same edge of \( E \) can be traversed by many vehicles. For more details, the reader may refer to [11].

Let \( I = \{1, \ldots, K\} \) be the set of \( K \) vehicles, \( R^* \) be the set of \((x, y)\) vectors, where \( x \in \mathbb{R}^{R \times I} \) and \( y \in \mathbb{R}^{E \times I} \). Thus for every vehicle, we associate a vector \((x, y) \in R^*\) defined by:

\[
x_{ep} = \begin{cases} 
1 & \text{if the vehicle } p \text{ services } e \in R; \\
0 & \text{otherwise}
\end{cases}
\]

and \( y_{ep} = \) the number of times that the vehicle \( p \) traverses the edge \( e \in R \) without servicing.

Let \( x_p(R') = \sum_{e \in R'} x_{ep} \) and \( y_p(E') = \sum_{e \in E'} y_{ep} \) where \( p \in I, R' \subseteq R E' \subseteq E \) and \( N \) denotes the set of all the nodes including the depot.

The linear mathematical model of this problem is given by the following:
The objective function \((2.32)\) minimizes the costs of service by all the required edges of all tours (the first term), and minimizes the number of times to pass through unnecessary as shown by the second term.

Constraints \((2.33)\) impose that each required edge is obligatory serviced by one and only one vehicle. The capacity constraints that assure that each vehicle must not exceed its capacity are shown in \((2.34)\). Constraints \((2.35)\), called the constraints of connectivity, impose that every serviced edge by a vehicle in a tour will be connected to the depot.

Constraints \((2.36)\) assure that each tour is induced by an even graph i.e. a graph whose all vertices are of even degree. Constraints \((2.37)\) impose the integrity of the decision variables.

The “Supersparse” Formulation

The “supersparse” formulation has been proposed in \([11]\) and \([12]\). This formulation is based on \(|E|\) “aggregated variables”. Due to the efficiency of the
“aggregated parity constraints” and the “aggregated capacity constraints” (see 2.2.8), the “supersparse” formulation of the CARP is given as follows: Let $w_e$ denotes the cost of traversing the edge $e$ and $z_e$ a binary variable equal to 1 if $e$ is traversed and 0 otherwise.

$$\text{SS-CARP} \quad \min \sum_{e \in E} w_e z_e$$

subject to:

$$z(\delta(S)) \geq 1 \quad \forall \ S \subseteq N \setminus \{\text{depot}\} \text{ such that } |\delta_r(S)| \text{ is even}$$

$$z(\delta(S)) \geq 2K(S) - |\delta_r(S)| \quad \forall \ S \subseteq N \setminus \{\text{depot}\}$$

$$z_e \in \mathbb{Z}^+_0 \quad \forall \ e \in E$$

The objective function of this model contains only the sum of the costs of traversing the edges and it does not contain the sum of the service costs of the required edges denoted by $w_r$ that is fixed and will be added finally to the objective function. This sum is given by:

$$w_r = \sum_{e \in E_R} w_e$$

This formulation was also presented in [116]. In [12] the authors have shown that with this formulation we have a risk of having infeasible solutions. They have added three different classes of “Disjoint path inequalities” to complement and reinforce the model.

The “Dense” Formulation

The origin of this formulation goes to [117]. The idea is to transform first the undirected graph $G = (V, E)$ into another graph $G' = (V', E')$ according to the following way: we assign the integers 1, 2, ..., $|E_r|$ to all the required edges of $G$. The new set of nodes $V'$ is composed of $1+2|E_r|$ nodes that are: The depot in the graph $G'$ denoted by $v_1$, the nodes $v_{i+1}$ and $v_{i+|E_r|+1}$ for $i = 1, \ldots, |E_r|$ that represent respectively the origin and the extremity of
each required edge \( e_i \) in \( E_r \).

Let \( E'_r = \{\{v_{i+1}, v_{i+|E_r|+1}\}, i = 1, \ldots, |E_r|\} \) a subset of the edges of \( E' \) representing the set of required edges of \( E_r \) in the graph \( G' \). The other edges in the graph \( G' \) i.e. \( E' \setminus E'_r \) is composed of fictional edges that are evaluated by the costs of shortest paths between their corresponding extremities in \( G \). The following dense formulation of the CARP gets its name from its quadratic number of variables (\(|E/E_r| = 2|E_r|^2\)). the author in [117] suggests a binary variable \( x_e \) for every \( e \in E' \setminus E_r \) equal to 1 if it is traversed and 0 otherwise.

\[
\text{D-CARP} \quad \min \sum_{e \in E' \setminus E'_r} w_e x_e \tag{2.43}
\]

subject to:

\[
x(\delta(v_i)) = 1 \quad \forall i = 2, \ldots, 2|E_r| + 1 \tag{2.44}
\]

\[
x(\delta(S)) \geq 2K(S) \quad \forall S \subseteq V' \setminus \{depot\}, S \text{ is “unbroken”} \tag{2.45}
\]

\[
x_e \in \{0, 1\} \quad \forall e \in E' \setminus E'_r \tag{2.46}
\]

\( S \subseteq V' \setminus \{depot\} \) is said to be unbroken if it verifies the following property: \( \forall i = 2, \ldots, |E_r| + 1, S \) contains \( v_i \) if and only if \( S \) contains \( v_{i+|E_r|} \). The two constraints (2.44) and (2.45) assure respectively that all the required edges are connected and that the capacity of the vehicles is not violated.

The theoretical interest of this formulation is more important than the practical one due to the great number of variables. A relation between a feasible solution of this formulation and a feasible solution of its CVRP corresponding equivalent problem is established [117]. The author has proposed new valid inequalities for the CARP starting from the formulation of the CVRP.
2.3 Robust Optimization

A central problem in optimization lies in addressing the data uncertainty in mathematical programming models. Data uncertainty has been addressed by two basic methods: the stochastic programming and the robust optimization.

- The Stochastic Programming:
  This approach was introduced in [50] where the data uncertainty was represented by scenarios for the data occurring with different probabilities. The two challenging difficulties of this method concern that:

  1. The enumeration of the scenarios capturing the exact distribution of the data that should be known is rarely attained.
  2. The size of the resulting optimization model increases drastically as a function of the number of scenarios.

- The Robust Optimization:
  In the Last two decades, a new optimization approach has been developing in which we optimize against the worst instances that might arise by using a min-max objective. Starting from its name, a robust solution means a strong solution that is feasible for all data uncertainty or modification as this could happen in an optimization problem. As a result of such disruption or perturbation, an optimal solution can be affected and could become infeasible. Thus it is important to find a robust solution that remains feasible whatever the perturbation in the data is. An approach integrating goal programming formulations with scenario-based description of the problem data is presented in [125]. Another approach that proposes a linear optimization model to construct a solution that is feasible for all input data such that each uncertain input data can take any value from an interval tends to find over-conservative solutions. This approach is proposed in [148]. However, this over-conservatism of the robust solutions is addressed in [16] [18] and [67] [68] by allowing the uncertainty sets for the data to be ellipsoids. However, the resulting formulations which are also robust involve conic quadratic problems [17] which cannot be applied to discrete optimization. A different approach that retains the complexity of the nominal problem of a combinatorial optimization problem and offers the modeler the ability to control the tradeoff between cost and robustness is given in [22] [23]. The given approach allows to control
the degree of the conservatism of the solution, and it is computationally tractable both practically and theoretically. Moreover, according to the authors in [24], “instead of trying to protect the solution in some probabilistic sense to stochastic uncertainty, the decision maker seeks a solution that is feasible for any realization of the uncertainty in a given set”. Briefly, The aim of this type of optimization is to optimize the worst case value over all the uncertain data.

Applying robust optimization is not random, but it requires three important decisions [104]: (1) the model of the uncertain data, (2) the criteria of the selected robust optimization such as min-max and (3) the choice of a mathematical model and the methods to generate robust solutions. The authors in [104] consider the combinatorial optimization problems and study their algorithmic complexity of their robust versions. For example, finding a shortest path problem in a graph is a polynomial problem, while the robust version with two scenarios for the arc costs is NP-hard.
Chapter 3

The Sparse CARP: Mathematical Formulation and Transformations

3.1 Introduction

Sparse graphs appear in real life problems where its importance lies in its practical contexts especially in the exceptional convoys, distribution of dangerous chemical materials and in the case of getting the injured people while wars for example. Dealing with this problem has been done by two classes of approaches; the first class is a straight one which consists to give mathematical formulations of the problem, then to build algorithms (exact and near-optimal) according to the studied variant \[5, 12, 64\]. The second class is non-straight and consists of transforming the problem into an equivalent node routing problem and then solving the latter to give a solution to the initial problem \[77, 119, 133\]. In this chapter we are interested in the second class of approaches to deal with the capacitated arc routing problem defined over sparse feasible graph.

Moreover, throughout the literature, node routing problems were deeply studied and well developed concerning all its variants, and they have been considered by mathematical programming, heuristics and meta-heuristic algorithms. This is due to the wide real life applications of this problem. Thus several researchers have put the question and the possibility to transform arc routing problems into some equivalent node routing problems which
could enable the use of the results which are already obtained for the VRPs [77, 119, 133]. The most common problem among all the transformations presented in the literature is that it does not conserve special structures of the original graph where the original ARP is defined, and this is because once an ARP is transformed, the new considered graph is the complete graph over the new nodes obtained by the corresponding transformation [128]. As we are interested in studying the arc routing problem over a sparse feasible graph, and as that the special structure that identifies our graphs is the sparsity, then we aim to conserve this property while applying a transformation of the ARP into an equivalent VRP.

In our thesis research, we propose a new transformation technique that conserves not only the original problem with all its properties, but also the characteristics of the original graph on which the problem is defined. This chapter consists of four sections. The second section discusses the previous mathematical models of the problem and introduces another two new formulations of the problem. In section 3, we present briefly the previous transformations of the arc routing problem into node routing problem. In section 4 we propose and present a new transformation technique for the sparse ARP. We give the corresponding theoretical results for such transformation, and we give some numerical results that validate the proposed transformation in section 4. In the end of the chapter, we present a conclusion.

3.2 Mathematical formulations of the sparse CARP

First of all, we may highlight a major remark which consists that one should not confuse between a sparse formulation of the CARP (for example see [11]) and a sparse CARP that we are considering in this research work. Let $G = (V, A)$ be a graph on which a capacitated arc routing problem is defined. As a fact, a sparse formulation of the CARP is a formulation whose number of variables is proportional to the number of edges or arcs of $G = (V, A)$ and this was first introduced by [11]. However, a capacitated arc routing problem is said to be sparse if its graph or network $G$ is sparse from graph theory point of view, i.e. a sparse graph is a graph which has a relatively small density which is defined to be the ratio of the number of actual edges of the
3.2. MATHEMATICAL FORMULATIONS OF THE SPARSE CARP

Graph to the maximum possible number of edges in the graph. In our study, we deal with sparse capacitated arc routing problem such that the maximum degree held by any vertex of the graph does not exceed 3. For more details and other definitions about sparsity, the reader may refer to [69] and [126]. In what follows, different mathematical models are presented in which each study the CARP under sparse graphs. Recall that our sparse graphs are graphs in which the degree of each vertex does not exceed 3.

3.2.1 Mathematical formulations of the sparse capacitated arc routing problems

In this section, we introduce for the first time the mathematical formulation of the sparse CARP. In the following, we detail the decision variables, the data and the parameters of the model. In this formulation, $K$ is always the set of the homogeneous vehicles and $Q$ is its capacity. For each edge $e$, let $c_e$ be its cost.

Denote by $\Delta_{\text{accum}}(e)$ the accumulated total demand served by the vehicle arriving at the service $e$ including the demand of $e$. This quantity is obviously less than or equal to $Q$. Let $N_e$ be the neighborhood of the edge $e$ and define a binary variable $x_{e,f}$ such that $x_{e,f} = 1$ if the edge $f$ is traversed directly after the edge $e$ by the same vehicle and 0 otherwise.

Remark 3.2.1. For the required edges, $x_{e,f}$ corresponds to traversing as well as servicing to the contrary of the non-required ones. In other terms, each required edge must be traversed exactly once while doing the required service simultaneously, whereas each non-required edge can be traversed at most one time.

Formally, in the following we present a mathematical formulation for the CARP where it is tractable for the sparse CARP too.

$$\min \sum_{e,f \in E} c_{e}x_{e,f} \quad (3.1)$$

subject to:

$$\sum_{f \in N(e)} x_{e,f} - \sum_{f \in N(e)} x_{f,e} = 0 \quad \forall e \in E \setminus \{0, 1\} \quad (3.2)$$
\[
\sum_{f \in N(0)} x_{f,0} = \sum_{f \in N(1)} x_{f,1} = K \quad (3.3)
\]

\[
\sum_{f \in N(0)} x_{f,0} = \sum_{f \in N(1)} x_{1,f} = 0 \quad (3.4)
\]

\[
\sum_{f \in N(e)} x_{e,f} \leq 1 \quad \forall e \notin R, e \notin \{0, 1\} \quad (3.5)
\]

\[
\sum_{f \in N(e)} x_{e,f} = 1 \quad \forall e \in R \quad (3.6)
\]

\[
\Delta_{\text{accum}}(f) \geq \Delta_{\text{accum}}(e) + d_f + (d_f + Q) \times (x_{e,f} - 1) \quad \forall f \in R, \forall e \in N(f) \quad (3.7)
\]

\[
x_{e,f} \in \{0, 1\}, \quad \Delta_{\text{accum}}(f) \leq Q, \quad \forall e, f \in E \quad (3.8)
\]

The objective function (3.1) minimizes the total distance traversed. Constraints (3.2) represent the conservation of flows between edges (number of successors is equal to the number of predecessors). All the vehicles must depart from the depot and return back to it while ending the services, this is shown in constraints (3.3) and constraints (3.4). Constraints (3.5) show that for non-required edges, each edge can be traversed by at most one time. Each required edge must be serviced while traversing exactly one time as shown in constraints (3.6). Constraints (3.7) assure that if \( f \) is traversed directly after \( e \), then the total demand done at the level of \( f \) is greater than or equal to the total demand done at \( e \), and otherwise the difference between these demands is less than \( Q \) which is trivial too. The last constraints (3.8) are for capacity and decision variables.

Another variant of the mathematical formulation

In this variant of the above mathematical model, we propose to consider several traversals \( \omega \geq 1 \) of the arcs. We have built the mathematical model by considering the following assumptions:

1. the high sparsity of the graph which is characterized by having a low density (down to 0.001 and less in some cases);
2. the maximum degree of any vertex is equal to 3 i.e. each vertex in the graph has at most 3 incident edges;

3. the set of the required edges is relatively small.

In this formulation, we allow each edge to be traversed a constant number of times. We call this constant the edge capacity and we denote it by $\omega_i$ for each edge $i \in E$. One should not confuse between this constant for each edge and the aggregated variables presented in [11] and [115]. These aggregated variables are decision variables that give the number of times an edge is traversed. However, our constant $\omega_i$ imposes a maximum number of times the edge $i$ can be traversed and is not considered as a decision variable. This is the first time such a constraint is included in a mathematical formulation of CARP as for all the formulations, the number of times for traversing an edge is whether 1 or a decision variable.

Consider the following notations:

- $K$: the total number of the vehicles where the fleet of vehicles is homogeneous;
- $Q$: the capacity of each vehicle;
- $\text{dem}(i)$: the demand of the edge $i$;
- $\Delta_{\text{accum}}(i)$: the accumulated total demand served by the vehicle arriving at the service $i$ including the demand of $i$ itself which is by definition less than or equal to $Q$;
- $c_i$: the cost of the edge $i$;
- $N(i)$: the neighborhood of the edge $i$;
- $\omega_i$: the capacity of edge $i$ i.e. the maximum number of times for which an edge can be traversed;
- $x_{e,f}^i$: a binary variable which is equal to 1 if and only if the service at $f$ is successive to the service at $e$ by the same vehicle and the chosen shortest path between $e$ and $f$ includes the consecutive adjacent edges $i$ and $j$, and 0 otherwise;
• \( y_{e,f} \): a binary variable equal to 1 if \( f \) is serviced directly after \( e \), and 0 otherwise.

We denote by 0 the edge representing the depot while the departure of the vehicles, and by 1 the edge representing the depot while returning back from doing the services.

A mathematical model:

\[
\min \sum_{e,f \in R, i \in E, j \in N(i)} c_{i}x_{i,j}^{e,f} \quad (3.9)
\]

subject to:

\[
y_{e,f} + y_{f,e} \leq 1 \quad \forall e, f \in R \quad (3.10)
\]

\[
\sum_{f \in R} y_{0,f} = K \quad (3.11)
\]

\[
\sum_{f \in R} y_{1,f} = 0 \quad (3.12)
\]

\[
\sum_{f \in R} y_{e,f} = 1 \quad \forall e \notin \{0, 1\} \quad (3.13)
\]

\[
\sum_{e \in R} y_{e,1} = K \quad (3.14)
\]

\[
\sum_{e \in R} y_{e,0} = 0 \quad (3.15)
\]

\[
\sum_{e \in R} y_{e,f} = 1 \quad \forall f \notin \{0, 1\} \quad (3.16)
\]

\[
\Delta_{\text{accum}}(f) \geq \Delta_{\text{accum}}(e) + \text{dem}(f) + (\text{dem}(f) + Q) \times (y_{e,f} - 1) \quad \forall e, f \in R, e \neq f \quad (3.17)
\]

\[
\sum_{j \in N(i)} x_{i,j}^{e,f} - \sum_{j \in N(i)} x_{j,i}^{e,f} = 0 \quad \text{if} \quad i \neq e, i \neq f, e, f \in R \quad (3.18)
\]

\[
\sum_{j \in N(e)} x_{e,j}^{e,f} - \sum_{j \in N(e)} x_{j,e}^{e,f} = y_{e,f} \quad e, f \in R \quad (3.19)
\]
3.2. MATHEMATICAL FORMULATIONS OF THE SPARSE CARP

$$\sum_{j \in N(f)} x_{e,f}^{e,f} - \sum_{j \in N(f)} x_{j,f}^{e,f} = -y_{e,f} \quad e, f \in R \quad (3.20)$$

$$\sum_{e,f \in R, j \in N(i)} x_{i,j}^{e,f} \leq \omega_i \quad \text{with} \quad \omega_i \geq 1 \quad \text{if} \quad i \neq 0 \quad (3.21)$$

$$\sum_{e,f \in R, j \in N(0)} x_{0,j}^{e,f} = K \quad (3.22)$$

$$\sum_{e,f \in R, j \in N(i)} x_{j,i}^{e,f} \leq \omega_i \quad \text{with} \quad \omega_i \geq 1 \quad \text{if} \quad i \neq 1 \quad (3.23)$$

$$\sum_{e,f \in R, j \in N(1)} x_{j,1}^{e,f} = K \quad (3.24)$$

$$x_{i,j}^{e,f}, y_{e,f} \in \{0, 1\}, \Delta_{\text{accum}}(e) \leq Q \quad \forall e, f \in R, i, j \in E \quad (3.25)$$

The objective function \((3.9)\) aims at minimizing the total costs. Constraints \((3.10)\) show that either \(e\) is serviced before \(f\) or vice versa. Constraints \((3.11)\) show that all the vehicles depart from the only departure of the depot which is represented by edge 0. Constraints \((3.12)\) are used to identify that the depot has only one departure. The number of successors of each edge \(e \notin \{0, 1\}\) is determined by Constraints \((3.13)\) where there exists one and only one successor. Constraints \((3.14)\) show that all the vehicles have to return back to the depot after finishing their corresponding services. Constraints \((3.15)\) determine that there is only one return to the depot and this return is given by the edge 1. The number of predecessors is determined by Constraints \((3.16)\) where each edge \(e \notin \{0, 1\}\) has exactly one only predecessor. Constraints \((3.17)\) assure that if \(f\) is served directly after \(e\) then the total demand at edge \(f\) is greater than or equal to the total demand at edge \(e\). Otherwise, the difference between these demands is less than \(Q\). Shortest path constraints are represented in \((3.18)\) to \((3.20)\). Constraints \((3.21)\) to \((3.24)\) determine the number of times an edge can be traversed, and this number is denoted by \(\omega\) and assigned to be named as the capacity of the edge for each edge of the graph. In detail, Constraints \((3.21)\) and \((3.23)\) impose that the number of times of traversing an edge is bounded by some \(\omega\) where this \(\omega\) is constant for the edge and may vary from one edge to another. For the departure and the return of the depot in \(G\), the maximum number of traversals is given by \(\omega=K\) ensuring the journey of all the vehicles from and
to the depot. Constraints (3.25) stand for capacity and decision variables.

In the next section, we study with the problem from a different approach. We introduce a brief survey on the transformation techniques as they exist in the literature which aim to transform an arc routing problem into an equivalent node routing problem. These transformations allow taking the advantage from the developed approaches to solve VRP, and which in turn allow to solve the original ARP.

3.3 From ARP to VRP: a general review

In this section, we start by recalling several transformations from ARP into VRP that have been previously developed in the literature. Throughout the following, we recall about the three most well-known transformation techniques of arc routing problems into node routing problems. In all what follows, let $G = (V,E)$ be a simple connected non-directed graph where $V$ is the set of vertices and $E$ is the set of edges or arcs. Let $R \subseteq E$ be the set of the required arcs i.e. the set of arcs with services such that $|R| = r$. The arc routing problems are encountered in a variety of practical situations where each street segment must be covered in its entirety. Such of these practical contexts are mentioned in Chapter 1 as garbage collection, school bus routing, mail delivery, road maintenance, salt gritting and meter reading...

In all what follows, we denote by $G' = (V',E')$ the undirected graph that is obtained by each of the techniques presented below i.e. The instance of the VRP obtained problem.

3.3.1 A first transformation technique

This transformation was presented in [133]. In this technique, every required arc in the original problem is replaced by three nodes. In detail, an edge $(i,j)$ in $R$ is associated to three vertices; $s_{ij}$ and $s_{ji}$ referred to be as side vertices and $m_{ij}$ referred to be as a middle vertex (see Figure 3.1). The new capacitated vehicle routing problem (CVRP) which is equivalent to the original capacitated arc routing problem (CARP) is defined on the complete undirected graph $G' = (V,E')$ where:
3.3. FROM ARP TO VRP: A GENERAL REVIEW

\[ V' = \bigcup_{(i,j) \in R} \{ s_{ij}, s_{ji}, m_{ij} \} \bigcup \{ \text{depot} \}. \quad (3.26) \]

Concerning the cost of the edges and the demands, they are defined as follows. The edge costs are given by \( d : E' \to \mathbb{Z}^+ \) as

\[
d(s_{ij}, s_{kl}) = \begin{cases} \frac{1}{4}(c_{ij} + c_{kl}) + \text{dist}(i, k), & \text{if } (i, j) \neq (k, l), \\ 0, & \text{if } (i, j) = (k, l). \end{cases}
\]

\[
d(\text{depot}, s_{ij}) = \frac{1}{4}c_{ij} + \text{dist}(0, i),
\]

\[
d(m_{ij}, v) = \begin{cases} \frac{1}{4}c_{ij}, & \text{if } v = s_{ij} \text{ or } s_{ji}, \\ \infty & \text{otherwise}. \end{cases}
\]

where \( \text{dist}(i, j) \) is the value of the shortest path from vertex \( i \) to the vertex \( j \) in \( G \) calculated according to the costs \( c \) in the original network.

Finally, the demands of the nodes in the VRP over the new network \( G' \) are defined by the function \( q : V' \to \mathbb{Z}^+ \) as:

\[
q(s_{ij}) + q(m_{ij}) + q(s_{ji}) = w_{ij} \quad \text{for all } (i, j) \quad (3.27)
\]

where \( w_{ij} \) is the demand of the arc \( (i, j) \in E \) and \( q(n) \) is the demand of the node \( n \in V' \). Since each required arc in \( G \) is replaced by three nodes in \( G' \), then the resulting instance has \( 3r + 1 \) vertices.

**Proposition 3.3.1.** [133] There is an equivalence between the VRP problem and the original arc routing problem on \( G \).

**Proof.** To this end, first note that any feasible solution to the CCPP can be transformed into a feasible set of tours for the VRP of no greater cost. Indeed, given a cycle occurring in the CCPP solution, one can replace each arc \( (i, j) \) that is serviced in the cycle by the three nodes associated to it. These node triplets are visited in the same order as the arc is traversed in the original solution. In other terms, if we move from \( i \) to \( j \) in the original graph, then we will move from \( s_{ij} \) into \( s_{ji} \) passing through \( m_{ij} \). Arcs that are traversed but not serviced in the CCPP solution, called deadhead arcs, do not appear in this transformation. To see that the resulting VRP tours
do not cost more than the original CCPP solution, note that the cost of any serviced arc is fully incurred in the VRP tour: one-half of this cost is paid by going from $s_{ij}$ to $s_{ji}$ via $m_{ij}$, while the other half is accounted for in travel costs into $s_{ij}$ and out of $s_{ji}$. Next, consider deadhead arcs in the CCPP solution. Suppose that $(i, j)$ and $(k, l)$ are two consecutive serviced arcs with $j \neq k$ so that a vehicle has to deadhead from $j$ to $k$. In the transformed VRP tour, node $s_{ji}$ is connected to $s_{kl}$ at a cost of $\frac{1}{4}(c_{ij} + c_{kl}) + dist(j, k)$. The term $dist(j, k)$ cannot exceed the cost of deadheading between $j$ and $k$ in $G$, while the terms with the $\frac{1}{4}$ factor are absorbed into traversal costs for the serviced arcs $(i, j)$ and $(k, l)$, as outlined above.

The above shows that any feasible CCPP solution can be transformed into a VRP solution of no greater cost. Conversely, the optimal VRP tours correspond to CCPP cycles of equal cost. This follows since any middle node $m_{ij}$ must always be visited in the sequence $s_{ij}, m_{ij}, s_{ji}$ or $s_{ji}, m_{ij}, s_{ij}$ to avoid encountering infinite distances or visiting either of the side nodes twice. Thus, the triplets $(s_{ij}, m_{ij}, s_{ji})$ in each tour of the VRP solution are in 1:1 correspondence with the serviced arcs of a CCPP cycle of equal cost. This establishes the equivalence of the original arc routing problem and its node routing counterpart.

![Figure 3.1: Transforming an edge in $G$ into three nodes in $G'$ allocating the edge demand to the new nodes.](image)
The drawback of this transformation is that it is limited in the practical application to instances where the number of vertices does not exceed 100.

### 3.3.2 A second transformation technique

In the previous transformation in section 3.3.1, each required edge is replaced by three nodes in the new graph, two lateral nodes and one central node. The only purpose of the central node is to oblige any route that passes through any lateral node to pass in a sequence through the associated to the central vertex and then through the another associated lateral node. In other words, supposing that the required edge is \((i, j)\), then the replacing nodes are \(s_{ij}, m_{ij}, s_{ji}\) arranged in the sequence \(s_{ij} \rightarrow m_{ij} \rightarrow s_{ji}\) or \(s_{ji} \rightarrow m_{ij} \rightarrow s_{ij}\). The transformation presented in [119] proposed the elimination of the central node \(m_{ij}\), but still ensuring that the lateral nodes are traversed sequentially (see Figure 3.2). The new obtained node routing problem graph has \(2r + 1\) nodes where the graph \(G' = (V', E')\) with:

\[
V' = \bigcup_{(i,j) \in R} \{s_{ij}, s_{ji}\} \bigcup \{\text{depot}\}. \tag{3.28}
\]

The costs of the edges is given by the function \(d : A \rightarrow \mathbb{Z}^+\) where the costs between a lateral node and the depot is defined as

\[
d(\text{depot}, s_{ij}) = \text{dist}(\text{depot}, i); \tag{3.29}
\]

and the costs between two lateral nodes are defined by the following equation:

\[
d(s_{ij}, s_{kl}) = \begin{cases} 
0, & \text{if } (i, j) = (k, l), \\
c(i, j), & \text{if } (i, j) = (l, k), \\
\text{dist}(i, k) & \text{if } (i, j) \neq (k, l) \text{ and } (i, j) \neq (l, k).
\end{cases}
\]

Where \(\text{dist}(i, j)\) represents again the cost of the shortest path between the nodes \(i\) and \(j\) in the original graph \(G\).

The new demands are given by the function \(f : V' \rightarrow \mathbb{Z}^+\) defined as:

\[
q(s_{ij}) + q(s_{ji}) = w(i, j) \tag{3.30}
\]

with \(w(i, j)\) is as defined in the previous section.
Discussion

In the two previous transformations, [133, 119], the authors give the demands function as well the costs function as functions into $\mathbb{Z}^+$, however there is no guarantee for such supposition.

### 3.3.3 A compact transformation technique of arc routing problem into node routing problem

In this transformation [77], the number of the obtained nodes in the new graph $G'$ is $r + 1$ i.e. each required edge is replaced by one single node (see Figure 3.3). The fact that the previous transformations create a graph with an excessive number of nodes compared to the original arc routing problem leads to develop a new transformation that conserves the number of the required arc in the original graph with the number of the obtained required nodes in the new graph. Thus, the resulting graph $G'$ has a number of nodes equal to the number of the required edges in $G$ plus one which is the depot.

The set of vertices of $G'$ is defined as

$$V' = \{m_{ij}|\{i, j\} \in R \quad \text{and} \quad i < j\}. \quad (3.31)$$
The demand associated to each required edge of \( R \) is assigned to its corresponding node in \( V' \) as:

\[
q(m_{ij}) = w(i, j), \quad \forall \{i, j\} \in R
\]  

(3.32)

For the costs of the edges in the new graph, the authors in [77] covered the degenerate cases that may occur and studied the feasibility of the routes in \( G \) as well as the division of the costs of the edges of \( G \). Dividing edge costs in \( G \) consider four possible cases. Suppose that it is required to serve the two edges \( \{i, j\} \) and \( \{k, l\} \) consecutively in \( G \) with the minimum possible cost, then there are four possible ways to do this service:

1. start from the edge \( \{i, j\} \) in the direction \( i \rightarrow j \), then follow a shortest path in \( G \) from \( j \) to \( k \) and then traverse the edge \( \{k, l\} \) in the direction \( k \rightarrow l \);

2. start from the edge \( \{i, j\} \) in the direction \( i \rightarrow j \), then follow a shortest path in \( G \) from \( j \) to \( l \) and then traverse the edge \( \{k, l\} \) in the direction \( l \rightarrow k \);

3. start from the edge \( \{i, j\} \) in the direction \( j \rightarrow i \), then follow a shortest path in \( G \) from \( i \) to \( k \) and then traverse the edge \( \{k, l\} \) in the direction \( k \rightarrow l \);

4. start from the edge \( \{i, j\} \) in the direction \( j \rightarrow i \), then follow a shortest path in \( G \) from \( i \) to \( l \) and then traverse the edge \( \{k, l\} \) in the direction \( l \rightarrow k \).

The cost of (1) is given by the cost of \( (i, j) \) added to the minimum cost of any subwalk between the nodes \( j \) and \( k \) in \( G \) and to the cost of \( (k, l) \).

\[
c(i, j) + \text{dist}(j, k) + c(k, l)
\]  

(3.33)

Note that the cost of \( (i, j) \) is the same as the cost of \( (j, i) \) for all required edge \( \{i, j\} \in R \).

For the costs of (2), (3) and (4), they are determined in a similar way. On the other hand, the equivalent subtours in the graph \( G' \) serve the node \( m_{ij} \) and then the node \( m_{ji} \) in a consecutive successive way. Moreover, a subwalk in \( G \) that serves the required edge \( \{i, j\} \) is equivalent to a subtour in \( G' \) that serves the node \( m_{ij} \), and this requires a subdivision of the cost \( c(i, j) \) into
two halves; the first is for the edge arriving at the node $m_{ij}$ and the second is for the edge leaving the node $m_{ij}$.

For the edge costs in $G'$, the authors introduce a couple binary variables in order to identify the direction of the traversal of the edges in $G'$, and they show the equivalence of the problems upon applying their transformation. By using of an adapted version of branch-cut-and-price algorithm for a capacitated node routing problem on the transformed graph, an effective approach for solving the capacitated arc routing problem is given.

The theoretical results of this transformation are reviewed below:

**Theorem 3.3.1.** [77] Suppose that the proposed transformation is used to turn a given arc routing problem instance $AR(G)$ say, based on a graph $G$, into a node routing problem $NR(H)$ say, based on a graph $H$. Let $z_{AR(G)}^*$ be the cost of the optimal solution to $AR(G)$. Further, for all $a \in \{0, 1\}^r$, let $z_a^*$ be the optimal solution to the instance of $NR(H)$ with edge costs specified by $a$. Then

$$\min_{a \in \{0, 1\}^r} z_a^* = z_{AR(G)}^*. \quad (3.34)$$

**Proof.** Due to a very long proof, we have preferred to refer the reader directly to the paper by [77]. \qed
Theorem 3.3.2. ([77]) Suppose the proposed transformation is used to turn a given arc routing problem instance $AR(G)$ say, on a graph $G$, into a node routing problem instance $NR(H)$ say, on a graph $H$. Then, when the proposed BCP algorithm is applied to $NR(H)$, it will identify the least-cost set of $q$-routes among all possible edge cost specifications for $NR(H)$ arising from all $a \in \{0,1\}^r$.

Proof. Due to a very long proof, we have preferred to refer the reader directly to the paper by [77]. □

3.4 A transformation technique of sparse CARP into CVRP with sparse feasible graph

To the most of our knowledge, no transformation does exist in the literature that addresses the CARP over sparse graphs and conserves both the sparsity of the original graph and the structure of the problem too. The previous transformations consider the obtained CVRP on its complete graph and this is in total contradiction with our study which aims at studying the problem over sparse graphs.

3.4.1 The steps of the transformation technique

Searching for a specific transformation from sparse CARP into sparse CVRP is basically important to us in order to conserve some characteristics such as the structure of the original graph without affecting the structure of the problem itself. Several attempts to build such a transformation has been done but without attaining our main goal as every time we were stuck in losing the structure of the original graph. Note that we are interested in conserving the structure of the original graph due to the fact that it may help us in solving the problem in some cases as for minimizing the time of resolution for example. As we aim at studying the arc routing problem over sparse graphs, and due to the fact that all the obtained node routing problems are studied over the complete graphs, this means that there is no transformation out of those presented above can be adapted to our case (the sparsity hypothesis). In the next, we detail the transformation that we use for the sparse CARP.
An ARP defined over a graph $G = (V, E)$ is transformed into an equivalent VRP over a graph $G' = (V', E')$ where $G'$ corresponds to be just the line graph of $G$; $G' = L(G)$. Every edge $e = \{i, j\} \in E$ is represented by one single node $m_{ij}$ in $G'$ (not only the required edges are considered and transformed into required nodes, but also the non-required ones). Hence, the resulting VRP problem is defined now over $L(G) = (V', E')$ in which $V' = \bigcup_{(i,j) \in E} \{m_{ij}\} \bigcup \{\text{depot}\}$ (see Figure 3.4).

Concerning the motivations for applying this method for solving a sparse arc routing problem, this lies mainly in the concept of choosing the line graph [161, 54]. The structure of almost all the connected graph can be recovered completely from its line graph as the edges of $G$ are equivalent to vertices in $L(G)$, and the incident edges in $G$ are equivalent to adjacent vertices in $L(G)$. Moreover, line graphs of connected graphs can be recognized in a linear time [159].

![Graph G](image1)

![Line Graph L(G)](image2)

**Figure 3.4:** An illustrating example of the new transformation technique.
3.4.2 Costs Division

Throughout the following, each non-required edge is considered with a travel cost, and each required edge is considered with travel and service costs. Upon transforming the sparse ARP in a graph $G$ into an equivalent sparse VRP in $G' = L(G)$, we need to guarantee not to exceed the original costs. The following strategy of dividing the costs illustrates the equivalence of costs between the two problems. Suppose that $A$ is an arc routing problem in $G$, and let $S_G(A)$ be any feasible solution of $A$ in $G$. Let the set of routes of $S_G(A)$ be denoted by $\{W_k \mid k = 1, \ldots, K\}$ for some $K \geq 1$. For $k = 1, \ldots, K$, let $R_k \subset R$ denotes the set of the required edges that are serviced by $k$ in $W_k$. Let $W_k$ be as follows:

$$W_k = \{(0, v_1), (v_1, v_2), \ldots, (v_{p-1}, v_p), (v_p, 0)\}, \quad (3.35)$$

where $v_i \in V, e = (v_i, v_j) \in E$ and $0$ denotes the depot.

To determine explicitly the set of the required edges that are serviced in the corresponding walk $W_k$, we introduce a new index in order to enumerate these edges according to their rank in service. Set $R_k$ as:

$$R_k = \{(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4}), \ldots, (v_{i_{q-1}}, v_{i_q}), \ldots, (v_{i_{2q-1}}, v_{i_{2q}})\}, \quad (3.36)$$

Note that $q \leq p$. The cost of the corresponding walk is given by

$$c(W_k) = \sum_{e \in W_k} c_e = \sum_{(v_i, v_j) \in W_k} c(v_i, v_j), \quad (3.37)$$

with $c(v_i, v_j)$ is the total cost of the edge $e = (v_i, v_j)$ where this cost includes the routing cost $r_e$ and the task cost $t_e$ if the corresponding edge is required, and only the routing cost $r_e$ if it is non-required.

$$\begin{cases} c_e = r_e + t_e & \text{if } e \in R; \\ c_e = r_e & \text{if } e \in E \setminus R \end{cases}$$

where $r_e$ denotes the routing cost of $e$ and $t_e$ denotes the task cost of it if it is required.

Now, let $T(W_k)$ be the set of the routes which correspond to the solution of the obtained problem by the proposed transformation. The image of an edge $(v_i, v_j) \in G$ is the node $m_{v_i,v_j} \in L(G)$. 
CHAPTER 3. THE SPARSE CARP: MATHEMATICAL
FORMULATION AND TRANSFORMATIONS

For simpler notations, let \( R_k = \{(i_u, i_{u+1}); u = 1, \ldots, 2q - 1\} \). Hence, the set of nodes of \( T(W_k) \) in \( L(G) \) is given by:

\[
T(W_k) = \{0, m_{1,2}, m_{2,3}, \ldots, m_{p-1,p}, 0\}, \tag{3.38}
\]

and that of the required nodes in \( L(G) \) which are serviced in \( T(W_k) \) is

\[
T(R_k) = \{m_{i_u,i_{u+1}}; u = 1, \ldots, 2q - 1\}. \tag{3.39}
\]

To determine the cost of \( T(W_k) \), consider the following two cases:

1. **Case 1**: \( T(W_k) \) does not contain consecutive adjacent required nodes.
2. **Case 2**: \( T(W_k) \) contains some consecutive adjacent required nodes.

**Case 1**:

\[
c(T(W_k)) = c(0, m_{v_1,v_2}) + \sum_{v_t \notin T(R_k)} \sum_{u=1}^{2q-1} c(m_{i_u,i_{u+1}}, m_{i_{u+1},v_t}) + \sum_{v_i,v_j,v_l \notin T(R_k)} c(m_{v_i,v_j,m_{v_j,v_l}}) + c(m_{v_{p-1},v_p}, 0) \tag{3.40}
\]

Where the costs are:

\[
\begin{align*}
c(0, m_{v_1,v_2}) &= r(0, v_1) + \frac{1}{2} r(v_1, v_2); \\
c(m_{i_u,i_{u+1}}, m_{i_{u+1},v_t}) &= \frac{1}{2} [r(i_u, i_{u+1}) + r(i_{u+1}, v_t) + t(i_u, i_{u+1})]; \\
c(m_{v_i,v_j,m_{v_j,v_l}}) &= \frac{1}{2} [r(v_i, v_j) + r(v_j, v_l)]; \\
c(m_{v_{p-1},v_p}, 0) &= r(v_p, 0) + \frac{1}{2} r(v_{p-1}, v_p)
\end{align*}
\]

**Remark 3.4.1.** If the nodes \( m_{v_1,v_2} \) and \( m_{v_{p-1},v_p} \) are required nodes, then we should also add the task costs of the edges \( (v_1, v_2) \) and \( (v_{p-1}, v_p) \) to the first term in \( c(T(W_k)) \), and exclude them out of the second term of (3.40).

**Case 2**:

In this case, we decompose the set of the required edges \( R_k \) into two subsets; \( R_k^1 \) which contains the required edges that are adjacent and consecutive, and \( R_k^2 \) that contains the remaining required which are neither adjacent nor
3.4. A TRANSFORMATION TECHNIQUE OF SPARSE CARP INTO CVRP WITH SPARSE FEASIBLE GRAPH

consecutive. Similar to the first case, we just modify the second term of (3.4.2) into two terms where the first one represents the costs of the edges in \( R^1_k \) and the another one represents the costs of the edges in \( R^2_k \). This can be illustrated as as follows:

\[
\sum_{(v_{iu},v_{i+1}) \in R^1_k} c(m_{iu},i_{iu+1},m_{iu+1},i_{iu+2}) + \sum_{v_t \notin T(R_k)} \sum_{(v_{iu},v_{i+1}) \in R^2_k} c(m_{iu},i_{iu+1},m_{iu+1},v_t) \tag{3.4.1}
\]

For the first term in (3.4.2), note that it also includes half the routing and the task costs of the corresponding edges. Moreover, the previous remark is still necessary for the second case.

**Proposition 3.4.1.** The original sparse CARP and its transformed CVRP with sparse feasibility graph are equivalent problems.

**Proof.** On one hand, if \( S \) is a solution of the CARP in \( G \) then \( S \) is a true cycle (closed walk respectively) which has an isomorphic cycle in the VRP (closed walk or path respectively) as its image due to the distribution of the costs. On the other hand, if \( T \) is a solution of the obtained CVRP, then as \( T \) is in the line graph of the initial graph, we consider the following two cases:

i. \( T \) is a cycle, then \( T \) is a line graph of some graph in \( G \) and both are isomorphic. This is enough to show that as \( T \) is feasible (or optimal) then its inverse in \( G \) is also feasible (or optimal) (Figure 3.5).

ii. \( T \) is a closed walk and since \( T \) does not contain any of the 9 subgraphs mentioned in [138, 145, 88] as induced subgraphs, then \( T \) admits an inverse that is isomorphic too.

This transformation conserves not only the structure of the problem, but also the sparsity of the original graph. To show the validation of this new transformation, it is required to present theoretical results and numerical experiments. In this research, we are interested in studying the arc routing problem over particular sparse graph as will be shown in the following section.
CHAPTER 3. THE SPARSE CARP: MATHEMATICAL FORMULATION AND TRANSFORMATIONS

A cycle in $G$ has its image as a cycle in $L(G)$

Suppose the solution is the closed walk 1-2-3-6-3-4-5-6-7-1,
Then its image is a path: $m_{1,2} - m_{2,3} - m_{3,6} - m_{3,4} - m_{4,5} - m_{5,6} - m_{6,7} - m_{4,7}$

Figure 3.5: Examples about the solution in $G$ and $L(G)$.

3.5 Graph density conservation

In this section, we show the graph density conservation of the underlying sparse graph of the CARP and the underlying sparse graph of the CVRP obtained by the proposed transformation. In the following, we present the theoretical results that validate our transformation. We denote by $d_G(i)$ the degree of the vertex $i$ in the graph $G$ and $\Delta(G)$ the maximum degree in $G$.

Remark 3.5.1. The used formulas of densities of the graph $G = (n, m)$ and
its line graph $L(G)$ are given by the following.

$$
density(G) = \frac{2m}{n(n-1)} \quad \text{and} \quad density(L(G)) = \frac{\sum_{v \in V(G)} d^2_G(v) - 2m}{m(m-1)} \quad (3.42)
$$

**Theorem 3.5.1.** Let $G$ be a sparse connected graph of $n$ vertices and $m$ edges such that $n=m$. Suppose that the maximum degree of $G$ is $\Delta(G) = 3$. If there is one and only one vertex $i$ such that $d_G(i) = \Delta(G) = 3$, then

1. $\sum_{v \in V(G)} d^2_G(v) = 2(2n+1)$.

2. the density of the line graph of $G$ is given by $density(L(G)) = \frac{2(n+1)}{n(n-1)}$.

**Proof.** Starting from the formula of the density of the line graph which is given by

$$
density(L(G)) = \frac{\sum_{v \in V(G)} d^2_G(v) - 2m}{m(m-1)} \quad (3.43)
$$

We have $\sum_{v \in V(G)} d_G(v) = 2|E(G)| = 2m = 2n$ where $|E(G)|$ denotes the number of edges of $G$. As there is one and only one vertex of degree 3 in $G$ which is also the maximum one, then despite of this vertex, we have the sum of the degrees of the remaining vertices is $2n - 3$. However, the remaining vertices are $n-1$ in which each has a minimum degree of 1 (as $G$ is connected), and a maximum degree of 2, thus there are $n-2$ vertices for which each has degree 2 and only one single vertex with degree 1. As a result:

$$
\sum_{v \in V(G)} d^2_G(v) = (2)^2(n-2) + (1)^2(1) + (3)^2(1) = 4n + 2 = 2(2n + 1) \quad (3.44)
$$

Applying 3.43 we get

$$
density(L(G)) = \frac{4n + 2 - 2n}{m(m-1)} = \frac{2(n+1)}{n(n-1)} \quad (3.45)
$$

\hfill \Box
Corollary 3.5.1. Let $G$ be a graph satisfying the conditions of Theorem 1, and let $A$ be an arc routing problem defined over $G$. Then, upon transforming $G$ into its line graph $L(G)$, the sparsity of $G$ is conserved as well as the structure of the problem $A$ where all its properties are still conserved in the obtained node routing problem in $L(G)$.

Proof. We have $\text{density}(G) = \frac{2m}{n(n-1)} = \frac{2n}{n(n-1)}$. Then $\text{density}(L(G)) - \text{density}(G) = \frac{2(n+1)}{n(n-1)} - \frac{2n}{n(n-1)} = \frac{2}{n(n-1)}$ which tends to zero as $n$ tends to $+\infty$.

On the other hand, due to the strategy of dividing the costs in the new graph and due to the advantages of conserving all the properties of the initial graph by applying this transformation, this technique shows that the problem is conserved with the only difference that tasks are now over the nodes instead of being over the edges. □

Theorem 3.5.2. Let $G$ be a connected graph of $n$ vertices and $m$ edges such that $m = n + \alpha$ with $1 \leq \alpha \leq \frac{n}{2}$ if $n$ is even and $1 \leq \alpha \leq \frac{n-1}{2}$ if $n$ is odd. Suppose that the maximum degree held by the vertices of $G$ is $\Delta(G) = 3$. If there are exactly $2\alpha$ vertices having this degree, then

1. $\sum_{v \in V(G)} d_G^2(v) = 2(2n + 5\alpha)$.

2. The density of the line graph of $G$ is given by $\text{density}(L(G)) = \frac{2(n+4)}{n(n+1)}$.

Proof. Following the same strategy of proving Theorem 1, and starting from the definition of the density of the line graph, let’s first determine the value of $\sum_{v \in V(G)} d_G^2(v)$. We have $\sum_{v \in V(G)} d_G(v) = 2|E(G)| = 2(n + \alpha)$ since $m = n + \alpha$.

As there are $2\alpha$ vertices having the maximum degree 3, then by leaving these vertices aside and considering the remaining ones, we get that the sum of the degrees of the remaining vertices is $2n - 4\alpha$, $(2(n+\alpha) - 3(2\alpha) = 2n - 4\alpha)$. But the remaining vertices are $n - 2\alpha$ in which each has a maximum degree of 2 and a minimum degree of 1, thus each one of these vertices has a degree 2. As a result, $\sum_{v \in V(G)} d_G^2(v) = (2)^2(n - 2\alpha) + (3)^2(2\alpha) = 4n + 10\alpha$. Now, by substituting this value in the formula of the density of $L(G)$, we get $\text{density}(L(G)) = \frac{4n + 10\alpha - 2(n + \alpha)}{(n+\alpha)(n+\alpha-1)} = \frac{2(n+4\alpha)}{(n+\alpha)(n+\alpha-1)}$. □
Corollary 3.5.1. Let $G$ be a graph that verifies the conditions of Theorem 2. Let $A$ be an arc routing problem defined over $G$. Applying the proposed transformation on $G$ yields to the conservation of the sparsity of it while obtaining a new node routing problem over its line graph $L(G)$. Moreover, the structure of the problem $A$ is also conserved.

Proof. We have $\text{density}(G) = \frac{2m}{n(n-1)} = \frac{2(n+\alpha)}{n(n-1)}$. Then $\text{density}(L(G)) - \text{density}(G) = \frac{2(n+4\alpha)}{(n+\alpha)(n+\alpha-1)} - \frac{2(n+\alpha)}{n(n-1)}$ which tends to zero as $n$ tends to $+\infty$. This shows clearly that sparsity is conserved by this transformation. For the conservation of the structure of $A$, this is evident thanks to the properties of the proposed technique and to the division of the costs in the new obtained graph. 

Remark 3.5.2. For $\alpha = \frac{n}{2}$ if $n$ is even and $\alpha = \frac{n-1}{2}$ if $n$ is odd, the density of the line graph is smaller than that the density of the original graph, and this shows that the sparsity is not lost.

3.5.1 Validation of the transformation

To validate the transformation technique, we present some numerical examples that show the conservation of sparsity between the original graph and its corresponding line graph upon the conditions of the theorems presented above. Note that the following density values are rounded to the nearest $10^{-3}$.

| $|V(G)|$ | $|E(G)|$ | Density of $G$ (%) | $|V(L(G))|$ | $|E(L(G))|$ | Density of $L(G)$ (%) |
|-------|-------|-------------------|-------|-------|-------------------|
| 35    | 35    | 0.059             | 35    | 36    | 0.061             |
| 50    | 50    | 0.041             | 50    | 51    | 0.041             |
| 72    | 72    | 0.028             | 72    | 73    | 0.029             |
| 112   | 112   | 0.018             | 112   | 113   | 0.018             |
| 130   | 130   | 0.015             | 130   | 131   | 0.016             |
| 190   | 190   | 0.01              | 190   | 191   | 0.01              |

Table 3.1: Conservation of sparsity between $G$ and $L(G)$ for $n = m$

As we deal with sparse graphs whose densities are relatively very small (between 0.3 and 0.001), and since there are no benchmark instances respecting our criteria of graphs, we generated our own instances for which we applied the exact method. In the following, we introduce some numerical...
### CHAPTER 3. THE SPARSE CARP: MATHEMATICAL FORMULATION AND TRANSFORMATIONS

| $|V(G)|$ | $|E(G)|$ | Density of $G$ (%) | $|V(L(G))|$ | $|E(L(G))|$ | Density of $L(G)$ (%) |
|---|---|---|---|---|---|
| 30 | 35 | 0.08 | 35 | 50 | 0.084 |
| 40 | 50 | 0.064 | 50 | 80 | 0.065 |
| 50 | 65 | 0.053 | 65 | 110 | 0.053 |
| 100 | 120 | 0.024 | 120 | 180 | 0.025 |
| 130 | 160 | 0.019 | 160 | 250 | 0.02 |
| 200 | 250 | 0.013 | 250 | 400 | 0.013 |

Table 3.2: Conservation of sparsity between $G$ and $L(G)$ for $n$ even and $n + 1 \leq m \leq \frac{3n}{2}$

| $|V(G)|$ | $|E(G)|$ | Density of $G$ (%) | $|V(L(G))|$ | $|E(L(G))|$ | Density of $L(G)$ (%) |
|---|---|---|---|---|---|
| 35 | 45 | 0.076 | 45 | 75 | 0.076 |
| 47 | 60 | 0.056 | 60 | 99 | 0.056 |
| 59 | 72 | 0.042 | 72 | 111 | 0.043 |
| 113 | 130 | 0.021 | 130 | 181 | 0.022 |
| 127 | 153 | 0.019 | 153 | 231 | 0.02 |
| 191 | 250 | 0.014 | 250 | 427 | 0.014 |

Table 3.3: Conservation of sparsity between $G$ and $L(G)$ for $n$ odd and $n + 1 \leq m \leq \frac{3n}{2}$
3.5. GRAPH DENSITY CONSERVATION

results which we have obtained by CPLEX Java. In Table 3.4 the instances are represented where we indicate the number of vertices of the graph \(|V(G)|\), the number of edges \(|E(G)|\), the number of required edges \(|R|\), the density of \(G\), the number of vehicles and their capacity. In Table 3.5 we present the number of the instance with the corresponding optimal value determined by Cplex, the best lower bound BLB, the given gap by Cplex and the CPU consuming time. Note that the total demand of each of the following instances is 100.

| Instance | \(|V(G)|\) | \(|E(G)|\) | \(|R|\) | Density of \(G\) (%) | \(K\) | \(Q\) |
|----------|--------|--------|-------|----------------|-----|-----|
| Inst 1   | 10     | 15     | 5     | 0.33           | 3   | 40  |
| Inst 2   | 10     | 15     | 5     | 0.33           | 3   | 45  |
| Inst 3   | 20     | 25     | 7     | 0.132          | 3   | 45  |
| Inst 4   | 20     | 25     | 10    | 0.132          | 3   | 45  |
| Inst 5   | 25     | 32     | 13    | 0.107          | 3   | 45  |
| Inst 6   | 25     | 32     | 13    | 0.107          | 4   | 45  |
| Inst 7   | 45     | 55     | 13    | 0.056          | 3   | 45  |
| Inst 8   | 45     | 55     | 13    | 0.056          | 4   | 45  |
| Inst 9   | 45     | 55     | 13    | 0.056          | 5   | 30  |
| Inst 10  | 45     | 55     | 13    | 0.056          | 5   | 45  |
| Inst 11  | 45     | 63     | 26    | 0.064          | 4   | 45  |
| Inst 12  | 60     | 84     | 50    | 0.047          | 4   | 45  |
| Inst 13  | 92     | 129    | 77    | 0.031          | 4   | 45  |
| Inst 14  | 100    | 150    | 40    | 0.03           | 4   | 45  |
| Inst 15  | 123    | 172    | 103   | 0.023          | 5   | 35  |
| Inst 16  | 167    | 234    | 147   | 0.017          | 3   | 50  |
| Inst 17  | 167    | 250    | 100   | 0.018          | 3   | 50  |

Table 3.4: Table of different instances

The main factors that affect the resolution are clearly the sparsity of the graph and the number of used vehicles. This is due to the fact that as the sparsity increases, the chances of building routes to service the customers decreases. In detail, the condition imposed over the edges not to be traversed more than once leads to this result. Moreover, due to this condition, the number of vehicles cannot exceed randomly as this will make finding a feasible solution more difficult. However, this number should be chosen in such a way the customers are all served and the capacity of the edges is not violated. As the size of the graph increases, the resolution becomes more difficult and it
## CHAPTER 3. THE SPARSE CARP: MATHEMATICAL FORMULATION AND TRANSFORMATIONS

<table>
<thead>
<tr>
<th>Instance</th>
<th>Best/Opt</th>
<th>BLB</th>
<th>Gap (%)</th>
<th>CPU (T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst 1</td>
<td>578.1933</td>
<td>578.1933</td>
<td>0</td>
<td>0.01 s</td>
</tr>
<tr>
<td>Inst 2</td>
<td>880.9252</td>
<td>880.9252</td>
<td>0</td>
<td>0.01 s</td>
</tr>
<tr>
<td>Inst 3</td>
<td>1175.7633</td>
<td>1175.7633</td>
<td>0</td>
<td>0.01 s</td>
</tr>
<tr>
<td>Inst 4</td>
<td>1277.5154</td>
<td>1277.5154</td>
<td>0</td>
<td>0.01 s</td>
</tr>
<tr>
<td>Inst 5</td>
<td>Infeasible solution</td>
<td>-</td>
<td>-</td>
<td>0.11 s</td>
</tr>
<tr>
<td>Inst 6</td>
<td>1547.2404</td>
<td>1547.2404</td>
<td>0</td>
<td>0.01 s</td>
</tr>
<tr>
<td>Inst 7</td>
<td>Infeasible solution</td>
<td>-</td>
<td>-</td>
<td>0.14 s</td>
</tr>
<tr>
<td>Inst 8</td>
<td>1432.5167</td>
<td>1432.5167</td>
<td>0</td>
<td>0.02 s</td>
</tr>
<tr>
<td>Inst 9</td>
<td>1432.5253</td>
<td>1249.5424</td>
<td>12.77</td>
<td>0.09 s</td>
</tr>
<tr>
<td>Inst 10</td>
<td>2697.2824</td>
<td>2697.2824</td>
<td>0</td>
<td>0.04 s</td>
</tr>
<tr>
<td>Inst 11</td>
<td>2682.5056</td>
<td>2620.7202</td>
<td>2.30</td>
<td>4.11 s</td>
</tr>
<tr>
<td>Inst 12</td>
<td>3865.8261</td>
<td>3857.5975</td>
<td>0.21</td>
<td>1.38 s</td>
</tr>
<tr>
<td>Inst 13</td>
<td>2388.2356</td>
<td>2387.9079</td>
<td>0.01</td>
<td>1 min 21 s</td>
</tr>
<tr>
<td>Inst 14</td>
<td>4327.4777</td>
<td>4300.0929</td>
<td>0.63</td>
<td>1 min 16 s</td>
</tr>
<tr>
<td>Inst 15</td>
<td>5748.3681</td>
<td>5731.8792</td>
<td>0.29</td>
<td>26 min 34 s</td>
</tr>
<tr>
<td>Inst 16</td>
<td>lack of memory</td>
<td>-</td>
<td>-</td>
<td>92 min 36 s</td>
</tr>
<tr>
<td>Inst 17</td>
<td>lack of memory</td>
<td>-</td>
<td>-</td>
<td>97 min 7 s</td>
</tr>
</tbody>
</table>

Table 3.5: Obtained numerical results
3.6. CONCLUSION

stops as we see in Table 3.5 at instance 14 where the graph has 167 vertices and 234 edges.

3.6 Conclusion

In this chapter, we define the capacitated arc routing problem over special class of sparse graphs. The sparse graphs that we are dealing with are graphs with a very small density that have not been studied before in the literature. We impose (1) a strong relation between the number of edges and the number of vertices and (2) a condition on the maximum degree in the graph which do not exceed 3 in our study. We distinguish between the sparse formulation of the CARP and the formulation of a sparse CARP. The first means that the number of edges is proportional to the number of variables, whereas the another represents a formulation of the problem under a sparse graph and for the classes of our graphs. This was not done before. Two variants of mathematical formulation of our sparse problem are presented in this chapter where the first imposes that the number of times of traversing an edge of the network is at most 1, while the second gives a constant maximum number of times of traversals for each edge. The number of times of traversing an edge can be less than or equal to this maximum constant, but it cannot exceed it. The difference between our formulation and the others presented in literature is that ours does not count the number of times of traversals. However, the number of times of traversals is not considered as a decision variable and is not included in the objective function. We generate our own instances since all the benchmarks generated before in the literature cannot be used since they do not respect the conditions of our problem. In the second part of this chapter, we propose a new transformation technique that converts the capacitated arc routing problem into capacitated node routing problem. This transformation show its validation for the CARP studied over the class of sparse graphs of our work. Not only the sparsity of the problem is conserved by this transformation, but also the structure of the problem itself. This is the first time a CARP over a sparse graph is studied in this particular way, and it is the first time where a transformation of CARP into CVRP conserving the original structure of the graph takes place. As shown above, the problem is very hard to solve to optimality. In the next chapter, we develop results which could enable to compute good lower bound of the solution. Also, further research works will focus on some near-
optimal approaches in order to tackle medium and large size instances based on heuristics and meta-heuristics techniques.
Chapter 4

Efficient Algorithms for Solving the Sparse CARP

Introduction

In this chapter, we propose two variants of a heuristic algorithm to solve Sparse CARP$^1$. We develop a tabu search algorithm under dynamic graphs to solve the problem. Our first approach CH is a constructive heuristic that is used to construct an initial feasible solution of the problem. This approach is improved to get another randomized heuristic IRP that allows us to improve the solution that is obtained by the first constructive approach. The last approach is an adapted tabu-search algorithm under sparse dynamic graph. This TS algorithm starts with the feasible solution obtained by IRP and tries to ameliorate it to get a better one. Extensive computational tests on randomly generated problem instances show the effectiveness of the proposed approach. The TS algorithm yields satisfactory results within reasonable computational time. The approach outperformed also the commercial solver CPLEX v12.71 which was able to solve only small instances with a relative big CPU time for medium size instances. A major point in our work is that all these approaches are applied over the new problem that is obtained from the transformation technique presented in the previous chapter. In other terms, we consider the sparse capacitated arc routing problem, then we apply the transformation technique to get an equivalent capacitated vehicle routing

$^1$Sparse CARP is our short notation of the capacitated arc routing problem with sparse feasible graph
problem, and as last step we apply the approaches of this chapter to solve the obtained equivalent problem.

4.1 Solution approaches for the sparse CARP

No heuristic in the literature could be adapted to our sparse CVRP problem due to the different and special structures of the used graph. In the following, we build tailored based heuristics to solve the studied problem.

4.1.1 An initial solution for the transformed CARP

We present a new tailored heuristic algorithm to solve our special sparse CARP. The constructive heuristic CH that we develop is applied on the CVRP which is obtained by the discussed transformation in Chapter 3. Prior to detail CH, we give a brief recall on the dynamic graphs [55] that we use for the algorithmic design.

4.1.2 Dynamic graphs

For recall, a graph is said to be a dynamic graph if it is subjected to some discrete changes such as insertion or deletion of edges or vertices. Such graphs are important in graph algorithms to design algorithmic techniques and data structures. Deleting or inserting edges or vertices of the graph is known by “update on a graph”. This update includes the changes that are associated with the edges or the vertices as the cost for example. The notion of the dynamic graph algorithm facilitates the computation of the solution of a problem efficiently instead of recomputing it from the scratch each time. For more details, readers may refer to [55, 56].

In our work, we use the dynamic graph concept due to the update of the edge capacities $\omega$ each time an edge is traversed.

4.1.3 The constructive heuristic CH

In the following, we give the heuristic algorithm, namely CH, for the sparse CARP. CH is a constructive based heuristic and is applied on the obtained CVRP which is equivalent to the original CARP where the used graph is a
dynamic graph. In all what follows, we use the concatenation for each union denoted by $\cup^c$.

The procedure takes as input the original graph $g$ with $n$ vertices, $m$ edges and $\omega_{\text{max}}$ the maximal edge capacity held by the edges of $g$.

**Input:** An original CARP instance with graph $G_0$ and parameters $(n, m, \omega_{\text{max}})$

**Output:** A feasible solution $S$ with value $V(S)$ for the transformed CARP

---

| Initialization | 1. $G \leftarrow \tau(G_0; n, m, \omega_{\text{max}})$; |
| 2. $d \leftarrow 0$; $s \leftarrow 0$; $\text{next} \leftarrow -1$; |
| 3. $\text{Serv} = \emptyset$; $P = \emptyset$; $S_0 = \emptyset$ |
| 4. $\forall i \in N(G)$ $\omega(i) \leftarrow \omega_0(i)$; |
| 5. $S \leftarrow S_0$; |

**Main Steps**

6. While ($d < D$) do
7.   { $\text{Serv} \leftarrow \text{Serv} \cup^c \{s\}$; |
| 8.       $\text{path} \leftarrow \text{Dijkstra}(s, G)$; |
| 9.       If (Not Finish($d', P$)) then |
| 10.      $\text{next} \leftarrow \text{Next}(s, P)$; |
| 11.      Else |
| 12.         $\text{next} \leftarrow 1$; |
| 13.         $\text{path} \leftarrow \text{Select}(s, \text{next}, P)$; |
| 14.         $S \leftarrow S \cup^c (\text{path} \setminus \{\text{next}\})$; |
| 15.         For ($i \in \text{path} \setminus \{\text{next}\}$) do |
| 16.             $\omega(i) \leftarrow \omega(i) - 1$; |
| 17.         For ($i \in N(G)$) do |
| 18.             If ($\omega(i) = 0$) then |
| 19.                 $G \leftarrow G \setminus \{i\}$; |
| 20.             If ($\text{next} \neq 1$) then |
| 21.                   { $s \leftarrow \text{next}; d \leftarrow d + \text{dem}(s); d' \leftarrow d' + \text{dem}(s)$; |
| 22.           Else |
| 23.               $S \leftarrow S \cup^c \{1\}$; |
| 24.               $s \leftarrow 0$; |
| 25.               $d' \leftarrow 0$; } |
| 26. } |
| 27. Exit with $S$ with value $V(S)$.

---

Figure 4.1: A constructive heuristic for the transformed sparse CARP into sparse CVRP: the CH
We introduce the following notations used by CH:

- $g$: the original graph where the ARP is defined;
- $\tau(g)$: the transformation function to transform $g$ of the sparse CARP into $G$ of its corresponding CVRP;
- $G$: the obtained line graph of $g$;
- $N(G)$: set of nodes of $G$;
- Serv: set of services;
- $d$: total demand served at a current step;
- $d'$: total demand served at a current tour;
- $\text{dem}(s)$: demand of node $s$;
- $D$: total demand to be served;
- $Q$: the capacity of each vehicle;
- $\omega_0(i)$: initial capacity of node $i$;
- $\omega(i)$: capacity of node $i$ at a current step of construction;
- $s$: current service;
- next: next service returned by the procedure Next();
- $P$: all paths between the current service and the possible next service;
- $Dijkstra(current, G)$: gives all the shortest paths between a current service and all possible next services excluding the extremities of the paths;
- Finish($d', P$): returns true if $d' + \text{dem}(S) > Q \ \forall S \in S(P)$ and 0 else; where $S(P)$ is the set of services at the end extremities of Dijkstra paths;
- path: current selected path between the current and the next service;
4.1. SOLUTION APPROACHES FOR THE SPARSE CARP

- **Select**(s, next, P): it selects a path from P between the current and the next service;
- **S**: the solution (set of nodes taken by the vehicles during their tour).

CH is a constructive heuristic which uses an opportunism strategy to compute the distance between the current service and the next service. We detail in the following the main steps of CH.

- **Steps 6-8**: while there is unserved demand, add the service s to the set of services and reach to this service by a Dijkstra path.
- **Steps 9-12**: If the sum of the served demands does not exceed the capacity of the vehicle, then select a next service and consider all the paths connecting the current service to the next one. Else, return back to the depot.
- **Step 13**: select a path from the set of paths P.
- **Step 14**: update the solution to be the existing concatenated with the chosen path.
- **Steps 15-19**: update the capacity for all the vertices used in the chosen path.
- **Steps 20-21**: allow selecting a next service once the destination is not the depot. Otherwise, return back to the depot and start again with a new service where the served demand is initialized at zero. This is done in the remaining steps.

The general steps are divided into 3 stages. Each vertex starts with the initial capacity \( \omega_0 \), i.e. each vertex starts with an initial maximal number of times to be traversed. Finally, the stopping criteria is attained when the served demand \( d \) is equal to the total demand \( D \).

**Stage 1**

- compute all the paths between the current service and all other services that are different from the completed services;
- sort the completed services are stored in the list “service”.

**Stage 2**

- identification of “the end of tour”: the termination of the tour is detected when the demand of each non-achieved service added to the
served demand $d'$ exceeds the vehicle capacity $Q$, or if each non-achieved service is not connected to the current service considering the configuration of the dynamic graph at the current step;

- if “the end of the tour” is not attained then we select the best service obtained by the Dijkstra’s algorithm. Otherwise, the next service is associated to the return to the depot;

- in all cases we select the paths between the current and the next services which are calculated by Dijkstra’s algorithm in a previous step;

- the nodes of this path are added to the end of the solution list except the node of the next service.

Stage 3
This stage can be considered as an “update stage”.

- we decrease the capacity $\omega(i)$ of each node $i$ in the selected path by 1 except for the last node of this path;

- each node $i$ with a current capacity $\omega(i) = 0$ is removed from the dynamic graph. All the edges which are incident to this node are removed and any isolated node is removed too;

- the next service is associated to the current service. The demands $d$ and $d'$ are increased by $\text{dem}(i)$;

- the current tour is terminated by returning to the depot “node 1”;

- the current service is updated as the depot exit “node 0’” and the current served demand of the coming tour is 0.

A new tour begins if the current service $s$ is set to 0. Each tour starts from node 0 of $G$ (edge 0 of $g$) which represents the departure from the depot and ends at node 1 of $G$ (edge 1 of $g$) which represents the return to the depot. Also each edge has a capacity less than or equal to $\omega_{\text{max}}$. The parameters $\omega$ stand for the capacity of the nodes of the line graph $G$ and the solution is represented by the concatenation of the paths of different vehicles (tours).
4.1. SOLUTION APPROACHES FOR THE SPARSE CARP

4.1.4 An improving randomized procedure based heuristic: IRP

The improving randomized procedure (IRP) consists of running for a fixed number of times the first constructive heuristic (CH) with a factor of randomness when selecting the next service. It looks like the algorithm GRASP where for each run, we select the best next services according to a probability \( p \) starting by the value 1, decreasing gradually by \( \varepsilon \) until reaching a minimal fixed value \( \varepsilon \) and then increasing directly to its maximal value of 0.99. However (IRP) is not GRASP because there is no list of choices to choose the best, but we use the best according to some probability, and it does not end by a local search. The number of trials is initially fixed. (IRP) can be described as a combination of two strategies: (i) the opportunism strategy which gives the priority to the quality improvement and (ii) the randomness which gives the priority to find a feasible solution by diversifying the search.

The search operation moves from one service to another depending on a set of the probability values which takes 1 as a first value while the trial decision is not included in a forbidden list called “list”. Thus the next service is chosen by opportunism. According to both the probability value and the distance between these services, the next service is inserted in the solution if the next trial belongs to “list”.

The number of runnings allows the solution to be improved progressively. The IRP procedure is detailed in the figure 4.2.

Remark

For a given value of \( p \), IRP\((p)\) returns an improved solution if it is feasible, otherwise it returns an empty set. In what follows, we give some details of the randomized approach:

- Serv: set of the required services;
- list: a list that contains a set of forbidden decisions for a solution \( S \). We define a decision by the couple of service and its rank. If the service \( s \) has the index \( i \) in the set “Serv”, then \((s, i)\) represents a possible decision for the solution \( S \);
- \( t \): counter representing the current trial (or run) for IRP\((p)\);
- \( p \): probability used for IRP\((p)\);
CHAPTER 4. EFFICIENT ALGORITHMS FOR SOLVING
THE SPARSE CARP

### Input:
An original ARP instance with graph $g$ and parameters $(n, m, \omega_{\text{max}})$

### Output:
An improved feasible solution $S_{\text{best}}$ with value $V(S_{\text{best}})$ for the transformed CARP

#### Initialization
1. $G \leftarrow \tau(g; n, m, \omega_{\text{max}})$;
2. $S \leftarrow \emptyset$; $S_{\text{best}} \leftarrow S$;
3. list $\leftarrow \emptyset$;
4. $p \leftarrow 1$;
5. $t \leftarrow 0$;
6. $t_{\text{max}} \leftarrow \text{const}$;

#### Main Steps
7. While $(t \leq t_{\text{max}})$ do
8. 
9. 
10. If $S \neq \emptyset$ then
11. 
12. 
13. If $|\text{list}| = \text{maxSize}$ then
14. 
15. If $(V(S) \leq V(S_{\text{best}}))$ or $(S_{\text{best}} = \emptyset)$ then
16. 
17. 
18. If $(p = p)$ then
19. 
20. EndWhile;
21. Exit with $S$ with value $V(S)$.

---

Figure 4.2: An improving feasible solution: the IRP approach.

- $p$: minimal accepted value of the probability $p$;
- maxSize: maximal size of “list”;
- oldest: oldest forbidden trial decision;
- $S_{\text{best}}$: best solution constructed up to the current trial;
- $V(S)$: objective value of the solution $S$.

The difference between CH and IRP procedures resides in the return of the function “Next()”. In IRP, at each stage of the routing, we select the next task to be visited by choosing the best non-forbidden next service that it is not in the list “list” (of a shortest path or a minimal cost) with a probability
4.1. SOLUTION APPROACHES FOR THE SPARSE CARP

$p$. If not chosen, then we select the second best non-forbidden next service with the same probability. We reiterate until we obtain a best service to add to the solution.

Using different probabilities, we obtain different solutions with different qualities and then many trials are done. The current best solution is saved and returned at the termination of the process.

4.1.5 A tabu search algorithm for the sparse CARP

Tabu search (TS) is a metaheuristic based local search method which can be used for solving combinatorial optimization problems \cite{83, 92}. The main concept of TS resides in the use of the adaptive memory that allows the search to guide the solution process by exploiting the search history. The adaptive memory is exploited to forbid the search to re-consider solutions that have already been visited. Our memory implementation employs functions that encourage search diversification and intensification. These two TS components permit the search to escape from local optimal solutions and in many cases to find optimal solutions.

In this section, we present a tabu search algorithm for our sparse capacitated arc routing problem. This tabu search algorithm is adapted to work under a dynamic graph.

A neighborhood of the current solution

The neighborhood construction of this algorithm depends mainly on the service move in the graph. At each iteration we choose randomly a non-tabu service to be inserted in another position and evaluate the new shortest paths in the new dynamic graph.

During the exploration, we use a dynamic graph (as for CH) initialized by the line graph $G = L(g)$ representing the obtained CVRP by the above transformation. We also use a variable node capacity noted by $\omega$ which determines the number of times a node can be traversed by the vehicles. These two parameters are updated at each new exploration. We consider a neighborhood structure based on one service move. This means, in order to build a neighbor of a current solution, we change one service schedule of the current solution (the rank of the service), then the service has two possibilities for its move: (i) a move to another schedule of the same vehicle
or (ii) a move to another schedule of a different vehicle. We give in figure 4.3 the main steps of the TS algorithm adapted to our problem.

| Input: An original CARP instance with a solution $S$ and value $V(S)$ |
| Output: An efficient solution $S_{best}$ with value $V(S_{best})$ for the transformed CARP |
| Initialization |
| 1. $S \leftarrow \text{IRP}();$ |
| 2. $S_{best} \leftarrow S;$ $V(S_{best}) \leftarrow V(S);$ |
| 3. $\text{tlist} \leftarrow \emptyset;$ |
| Main Steps |
| 4. While (Not stopping condition) do |
| 5. neighborhood $\leftarrow \text{FindNeighborhood}(S);$ // constructs neighbors of the current solution |
| 6. $S \leftarrow \text{Best}(\text{neighborhood}_G) \setminus \text{tList};$ // selects the best neighbor that it is non-tabu |
| 7. Update $(\text{tList});$ // updates the tabu list |
| 8. Update $(G);$ // updates the dynamic graph $G$ |
| 9. If $(V(S) \leq V(S_{best})$ then $S_{best} \leftarrow S;$ |
| 10. EndWhile; |
| 11. Exit with $S_{best}$ with value $V(S_{best}).$ |

Figure 4.3: A tabu search based algorithm for the obtained CVRP with sparse feasibility graph: TS

The function $\text{Best}(\text{neighborhood}_G)$ selects the best neighbor of the solution in the dynamic graph $G$ at its current situation. When the solution changes, then $\omega$ changes and the number of traversing determines whether a traversal is allowed or not, for this we update the dynamic graph by the function $\text{Update} \ (G).$

The main principle of the search process

At each iteration of the exploration, we choose a random service then we remove it from its rank in the service schedule in the neighbor. We update the neighbor by removing the paths relating this service to its successor and its predecessor services. We evaluate using Dijkstra the shortest path between the predecessor of the service and its successor, then we insert this path in the neighbor. We insert the service in a random new rank in the service schedule by selecting a new random predecessor. Finally, considering the previous update of the dynamic graph $G,$ we apply Dijkstra to find two shortest paths. The first path is between the new predecessor service and
the related service, and the second one is between the related service and the new successor service.

We update the rank of each service in the solution schedule for each new neighborhood construction. These updated ranks are saved in the list of services. At each construction, we update the remaining capacity of each node as well as the dynamic line graph $G$.

If a node's capacity decreases to 0, then this node is removed from $G$. If it increases to a positive value, we add this node to $G$. Through this exploration process and for each current solution, if the constructed neighbor improves the solution, it is then inserted in the current solution. If no improvement is possible, we keep the best found solution found so far and we construct a new neighbor.

**Remark**

The solution space is not always connected since while building the neighborhoods, some cases may cause a disconnection. Such cases are discussed below:

Let $X$ be a solution such that $\forall$ service $s$ in $X$ we have $\omega(s) = 0$, and let $y$ be a neighbor of $X$ i.e. $y \in \text{neighborhood}(X)$. Suppose that $s$ is the service moved from its position in $X$ to a new position which builds the neighbor $y$. Let $s_{\text{pred}}$ (resp. $s_{\text{next}}$) be the predecessor of $s$ in $X$ (resp. its successor). If $s$ belongs to the set of all the possible paths between $s_{\text{pred}}$ and $s_{\text{next}}$, and for an updated $\omega$ in the dynamic graph after removing $s$ from its position between $s_{\text{pred}}$ and $s_{\text{next}}$ there is no path between $s_{\text{pred}}$ and $s_{\text{next}}$ not containing $s$, since $\omega(s) = 0$, then $s$ cannot be moved from its position between $s_{\text{pred}}$ and $s_{\text{next}}$ without disconnecting them definitely.

Generally, the solution space is disconnected because of the high sparsity of the graph. This disconnection increases with small values of average $\omega$. To adapt our approach to this particularity of the problem, we use an exploitation process which consists to restart the tabu search from different initial solutions obtained by the IRP procedure.

**Intensification and diversification strategies**

During the search process, if after a certain number of constructions the current solution is not improved, we select the last explored neighbor to replace the current solution. We choose this strategy of neighborhood construction
to reduce the combinatorial explosion of the solution search and the memory space. To escape from local optima regions, at each exploration of a new best solution, we put tabu the move which was used to construct that solution where the tabu list size is equal to the number of services multiplied by 10.

If promising zones are detected, we intensify the search and if the current explored region becomes not interesting (because of stagnation), we diversify the exploration. Once we obtain a new best solution, we update “maxIter” by its value times 2 except if it is equal to its maximal value (2000) as an intensification process. However, when the number of explorations exceeds the tolerated value (20000 iterations), we update “maxIter” by its minimal value (200 iterations) as a diversification process.

For the aspiration criteria, if a tabu move improves the found best solution, then we remove the tabu status from the move of the corresponding solution. The “maxIter” which is the size of the explored neighborhoods is varying and its value depends on the intensification and the diversification processes. Our TS uses a multi-start process to avoid being stuck in some areas of the solution space.

4.2 Numerical Results

In this section, we detail the numerical experiments carried out in order to test the proposed approach. We have coded all the algorithms with C++ language on a Ubuntu Linux 14.04 HP ProBook Core i3, 2.4 Ghz and 6 Go of Ram. These results are compared to those obtained by Cplex V12.71. In the benchmark, we consider 9 families of instances such that each family represents a set of instances with \( g \) denoting the original graph for which the CARP is defined. We study each family with the same number of vertices \( |V(g)| \), the same number of edges \( |E(g)| \), same number of services \( |R| \) and the same total demand \( D \). We define \( d(g) = \frac{2m}{n(n-1)} \) the density of the original graph \( g \) with \( n \) the number of nodes and \( m \) the number of the edges (rounded to the nearest \( 10^{-3} \)). We choose the total number of vehicles and their capacities such that their product is close to the total required demand. Note that we do not compare the results of our computational experiments with any other results done in the literature due to the fact that our approach is the only one that deals with graphs with high degree of sparsity (graphs where the maximum degree held by the vertices is 3).

In table 4.1 we detail the families of instances for which we run both
4.2. NUMERICAL RESULTS

| Family | $|V(g)|$ | $|E(g)|$ | $|R|$ | $d(g)$ (%) | $D$ |
|--------|--------|--------|------|------------|------|
| Family 1 | 15     | 18     | 8    | 0.171      | 1000 |
| Family 2 | 20     | 25     | 10   | 0.132      | 1000 |
| Family 3 | 50     | 70     | 33   | 0.057      | 1000 |
| Family 4 | 100    | 120    | 59   | 0.024      | 1000 |
| Family 5 | 120    | 142    | 95   | 0.02      | 1000 |
| Family 6 | 163    | 181    | 110  | 0.014      | 1000 |
| Family 7 | 231    | 317    | 121  | 0.012      | 1000 |
| Family 8 | 257    | 362    | 191  | 0.011      | 1000 |
| Family 9 | 307    | 439    | 309  | 0.001      | 1000 |
| Family 10 | 400   | 600    | 357  | 0.008      | 1000 |

Table 4.1: A first set of different families of CARP problem instances

Cplex and the CH algorithm. 10 families with different sizes and densities of graphs are presented. The first column gives the number of the family, the number of vertices and the number of edges are given in the second and third column respectively. Column 3 gives the number of services, and the density of the graph is given in column 4. The amount of the total demand is given in the last column.

Table 4.2 presents the obtained results by Cplex and by CH. In the first 4 columns, we give the number of the family, the average value of its corresponding edge capacity $\omega$, the number of the homogeneous vehicles $K$ and their capacity $Q$. The whole number of run instances is given in column 5 titled by All, and the number of feasible and optimal obtained solutions by Cplex is given in column 6, and by CH is given in column 7. Column 8 represents the gap between the solution values determined by Cplex and those determined by CH. This gap is given by:

$$\frac{\text{value given by CH} - \text{value given by Cplex}}{\text{value given by Cplex}}$$  \hspace{1cm} (4.1)

The last two columns give the consuming CPU time by Cplex and by CH.

The results in table 4.2 show the effectiveness of the CH algorithm by comparing the results obtained by it to that obtained by Cplex especially for the families with big size instances where Cplex fails to perform. This is shown by the number of feasible and optimal solutions obtained by Cplex on one hand and by the CH heuristic on the other hand.
CHAPTER 4. EFFICIENT ALGORITHMS FOR SOLVING
THE SPARSE CARP

To detail the results between Cplex and CH, we present in Table 4.3 some instances of the first two families 1 and 2, the corresponding edge capacity $\omega$, the number of vehicles $K$ and its capacity $Q$, the gap given by Cplex, the gap of CH which is calculated by:

$$\text{gap} = \frac{\text{value given by CH} - \text{value given by Cplex}}{\text{value given by Cplex}}$$  \hspace{1cm} (4.2)

The CPU consuming time by Cplex in seconds and the corresponding CPU time needed by the heuristic CH in seconds to solve the problem.

We note that the optimality of the solution is attained by Cplex for the most of the instances of families 1 and 2 with a small CPU consuming time. However, CH performs well by giving solutions that are close to the optimal ones in a very small CPU consuming time. This gives an overview about the effectiveness of the CH algorithm which gives a feasible near optimal solution with a very small time.

Starting from family 3 which has a moderate-size instances (50 nodes and 70 edges), Cplex cannot perform the solution due to the lack of its memory. However, CH performs well and gives solutions.

In Tables 4.4, we present the average results of different instances of sub-families of families 1 and 2 solved by Cplex and by IRP. For the IRP approach, we remark that more the average capacity of the edges increases, the algorithm needs less time to solve the problem. Moreover, this increase is faced by a decrease in the number of the IRP failures which validates that the complexity of the problem increases as the sparsity of the graph increases.

The IRP approach realizes better solutions than CH since it starts with the same initial solution and improves it gradually with a number of improvements.

Consider the following notations:

- “All”: the number of all tested instances.
- “#Feas”: the number of the feasible instances out of the all tested instances;
- “Average gap”: the average gap between the CPLEX and the heuristic IRP;
### 4.2. NUMERICAL RESULTS

Table 4.2: Comparison between Cplex and CH for different families of instances by CH.

<table>
<thead>
<tr>
<th>Family</th>
<th>(\omega)</th>
<th>(K)</th>
<th>(Q)</th>
<th>All</th>
<th>(\text{Cplex}_{\text{sol}})</th>
<th>(\text{CH}_{\text{sol}})</th>
<th>Av. gap (%)</th>
<th>Av. CPU\text{Cpx}</th>
<th>Av. CPU\text{CH}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family 1</td>
<td>2</td>
<td>3</td>
<td>400</td>
<td>60</td>
<td>9 F - 21 O</td>
<td>9 F - 3 O</td>
<td>4.799</td>
<td>0.47 sec</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 1</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>54</td>
<td>16 F - 18 O</td>
<td>13 F - 2 O</td>
<td>6.315</td>
<td>0.95 sec</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 2</td>
<td>2</td>
<td>3</td>
<td>400</td>
<td>10</td>
<td>6 F - 0 O</td>
<td>0 F - 0 O</td>
<td>31.84</td>
<td>15 min 32 sec</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 2</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>10</td>
<td>10 F - 0 O</td>
<td>0 F - 0 O</td>
<td>30.554</td>
<td>8 min 51 sec</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 3</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>10</td>
<td>0 F - 0 O</td>
<td>2 F - 0 O</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 3</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>10</td>
<td>0 F - 0 O</td>
<td>6 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 3</td>
<td>12</td>
<td>3</td>
<td>400</td>
<td>10</td>
<td>0 F - 0 O</td>
<td>6 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 4</td>
<td>2</td>
<td>3</td>
<td>400</td>
<td>10</td>
<td>0 F - 0 O</td>
<td>0 F - 0 O</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 4</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>10</td>
<td>0 F - 0 O</td>
<td>0 F - 0 O</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 4</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>10</td>
<td>0 F - 0 O</td>
<td>2 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 4</td>
<td>12</td>
<td>3</td>
<td>400</td>
<td>10</td>
<td>0 F - 0 O</td>
<td>2 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 4</td>
<td>12</td>
<td>4</td>
<td>300</td>
<td>10</td>
<td>0 F - 0 O</td>
<td>1 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 5</td>
<td>2</td>
<td>3</td>
<td>400</td>
<td>300</td>
<td>0 F - 0 O</td>
<td>0 F - 0 O</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 5</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>144</td>
<td>0 F - 0 O</td>
<td>3 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 5</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>107</td>
<td>0 F - 0 O</td>
<td>19 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 5</td>
<td>12</td>
<td>3</td>
<td>400</td>
<td>115</td>
<td>0 F - 0 O</td>
<td>26 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 5</td>
<td>12</td>
<td>4</td>
<td>300</td>
<td>133</td>
<td>0 F - 0 O</td>
<td>17 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 6</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>106</td>
<td>0 F - 0 O</td>
<td>0 F - 0 O</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 6</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>111</td>
<td>0 F - 0 O</td>
<td>1 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 6</td>
<td>12</td>
<td>3</td>
<td>400</td>
<td>125</td>
<td>0 F - 0 O</td>
<td>3 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 6</td>
<td>15</td>
<td>3</td>
<td>400</td>
<td>118</td>
<td>0 F - 0 O</td>
<td>12 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 6</td>
<td>15</td>
<td>4</td>
<td>300</td>
<td>154</td>
<td>0 F - 0 O</td>
<td>6 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 7</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>101</td>
<td>0 F - 0 O</td>
<td>22 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 7</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>102</td>
<td>0 F - 0 O</td>
<td>33 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 7</td>
<td>15</td>
<td>3</td>
<td>400</td>
<td>102</td>
<td>0 F - 0 O</td>
<td>36 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 8</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>175</td>
<td>0 F - 0 O</td>
<td>26 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 8</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>160</td>
<td>0 F - 0 O</td>
<td>52 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 9</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>108</td>
<td>0 F - 0 O</td>
<td>26 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 9</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>115</td>
<td>0 F - 0 O</td>
<td>35 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 10</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>106</td>
<td>0 F - 0 O</td>
<td>19 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 10</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>110</td>
<td>0 F - 0 O</td>
<td>31 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Family 10</td>
<td>17</td>
<td>3</td>
<td>400</td>
<td>108</td>
<td>0 F - 0 O</td>
<td>49 F</td>
<td>-</td>
<td>-</td>
<td>&lt; 1 sec</td>
</tr>
</tbody>
</table>

F: Feasible; O: Optimal
### Table 4.3: Results of different instances of families 1 and 2 by Cplex and by CH

<table>
<thead>
<tr>
<th>Family</th>
<th>$\omega$</th>
<th>$K$</th>
<th>$Q$</th>
<th>Cplex sol</th>
<th>CH sol</th>
<th>Cplex gap(%)</th>
<th>CH gap(%)</th>
<th>CPU_Cplex</th>
<th>CPU_CH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>400</td>
<td>823.75</td>
<td>967.04</td>
<td>17.4</td>
<td>0</td>
<td>0.16</td>
<td>0.04</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>400</td>
<td>994.08</td>
<td>994.08</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>400</td>
<td>1308.96</td>
<td>1363.13</td>
<td>4.1</td>
<td>0</td>
<td>0.22</td>
<td>0.09</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>300</td>
<td>1594.64</td>
<td>1594.64</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0.06</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>300</td>
<td>1217.44</td>
<td>1752.12</td>
<td>43.9</td>
<td>0</td>
<td>16.46</td>
<td>1.74</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>1204.97</td>
<td>1371.75</td>
<td>13.8</td>
<td>0</td>
<td>0.97</td>
<td>0.04</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>643.54</td>
<td>643.54</td>
<td>0</td>
<td>0</td>
<td>0.14</td>
<td>0.03</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>836.33</td>
<td>941.008</td>
<td>7.4</td>
<td>12.4</td>
<td>0.92</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>1290.28</td>
<td>1668.05</td>
<td>0</td>
<td>29.3</td>
<td>0.38</td>
<td>0.03</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>817</td>
<td>838.41</td>
<td>4.22</td>
<td>2.6</td>
<td>0.48</td>
<td>0.04</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>1297.08</td>
<td>1496.25</td>
<td>0</td>
<td>15.4</td>
<td>0.3</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>935.31</td>
<td>1241.82</td>
<td>1.44</td>
<td>32.8</td>
<td>0.53</td>
<td>0.03</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>400</td>
<td>805.6</td>
<td>1238.37</td>
<td>16.65</td>
<td>53.7</td>
<td>2.08</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>400</td>
<td>1169.39</td>
<td>1236.03</td>
<td>0</td>
<td>5.69</td>
<td>0.24</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>400</td>
<td>1601.11</td>
<td>1743.96</td>
<td>7.33</td>
<td>8.92</td>
<td>7.86</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>3</td>
<td>400</td>
<td>1383.52</td>
<td>1591.66</td>
<td>4.31</td>
<td>15.04</td>
<td>3.4</td>
<td>0.08</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>3</td>
<td>400</td>
<td>1796.84</td>
<td>1879.53</td>
<td>4.27</td>
<td>4.6</td>
<td>30.37</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>3</td>
<td>400</td>
<td>1284.38</td>
<td>1369.62</td>
<td>7.58</td>
<td>6.64</td>
<td>15.26</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td>300</td>
<td>1782.05</td>
<td>1824.11</td>
<td>5.39</td>
<td>2.36</td>
<td>24.28</td>
<td>0.09</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td>300</td>
<td>1071.92</td>
<td>1270.03</td>
<td>0.05</td>
<td>18.48</td>
<td>4</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>1535.99</td>
<td>1576.34</td>
<td>8.25</td>
<td>2.63</td>
<td>5.51</td>
<td>0.09</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>1210.22</td>
<td>1652.57</td>
<td>7.39</td>
<td>36.55</td>
<td>9.27</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>1367.82</td>
<td>1463.09</td>
<td>4.69</td>
<td>6.97</td>
<td>1.25</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>1337.27</td>
<td>1680.85</td>
<td>2.73</td>
<td>25.69</td>
<td>5.51</td>
<td>0.07</td>
</tr>
</tbody>
</table>
4.2. NUMERICAL RESULTS

- “Best gap”: the best gap between the heuristic IRP and the CPLEX;
- “Worst gap”: the worst gap between the heuristic IRP and the CPLEX;
- “# 0 gap”: the number of instances where both Cplex and IRP obtain the same value;
- “Av. CPU_{Cpx}”: the average Cplex time taken to solve the problem;
- “Av. CPU_{IRP}”: the IRP consuming time;
- “#Fails”: the number of instances where the heuristic failed to solve the problem;
- “#Imp”: the number of improvements done by IRP.

We run a group of instances of a sub-family of each main family of Table 4.4. Each sub-family the main characteristics of its corresponding main family, and has its own fixed parameters $\omega, K,$ and $Q$. Every instance of this sub-family has the same parameters of the sub-family itself.

For the first two families, we give the results obtained by CPLEX and IRP. The experiments were done according to the different values of $\omega$. The first column of Table 4.4 gives the number of the main family, the second gives the corresponding average $\omega$, the third column stands for the total number of instances done in one sub-family, and the number of the obtained feasible instances is given in column 4. The average gap between the IRP and Cplex is given in column 5. Columns 6 and 7 give the best and the worst gap between IRP and Cplex. The number of instances where Cplex and IRP produce the same solution is shown in column 8. The average consuming times of Cplex and IRP are given in columns 9 and 10. Column 11 shows the number of instances where IRP failed to improve, and finally in the last column, the number of improvements done by IRP is given. For the gap between Cplex and IRP, it is given by:

$$\frac{\text{value given by IRP} - \text{value given by Cplex}}{\text{value given by Cplex}}$$

In Table 4.5, we show the the number of family with the corresponding $\omega$, the number of the vehicles and their capacity. The number of the all tested
CHAPTER 4. EFFICIENT ALGORITHMS FOR SOLVING THE SPARSE CARP

<table>
<thead>
<tr>
<th>Family</th>
<th>ω</th>
<th>All</th>
<th>#Feas.</th>
<th>Av. gap</th>
<th>Best gap</th>
<th>Worst gap</th>
<th># 0 gap</th>
<th>Av. CPU_Cplex</th>
<th>Av. CPU_IRP</th>
<th>#Fails.</th>
<th>#Imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>280</td>
<td>50</td>
<td>0.089</td>
<td>0</td>
<td>0.328</td>
<td>4</td>
<td>1.173</td>
<td>0.007</td>
<td>35</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>121</td>
<td>50</td>
<td>0.126</td>
<td>0.017</td>
<td>0.283</td>
<td>0</td>
<td>1.815</td>
<td>0.009</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>88</td>
<td>50</td>
<td>0.094</td>
<td>0</td>
<td>0.221</td>
<td>4</td>
<td>5.386</td>
<td>0.009</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>64</td>
<td>50</td>
<td>0.107</td>
<td>0.018</td>
<td>0.295</td>
<td>0</td>
<td>3.660</td>
<td>0.008</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>62</td>
<td>50</td>
<td>0.089</td>
<td>0</td>
<td>0.270</td>
<td>3</td>
<td>3.177</td>
<td>0.009</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>53</td>
<td>30</td>
<td>0.171</td>
<td>0.010</td>
<td>0.325</td>
<td>0</td>
<td>52.211</td>
<td>0.240</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>39</td>
<td>30</td>
<td>0.171</td>
<td>0.025</td>
<td>0.401</td>
<td>0</td>
<td>177.680</td>
<td>0.026</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>36</td>
<td>30</td>
<td>0.147</td>
<td>0.032</td>
<td>0.265</td>
<td>0</td>
<td>141.199</td>
<td>0.036</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>34</td>
<td>30</td>
<td>0.128</td>
<td>0.015</td>
<td>0.249</td>
<td>0</td>
<td>61.547</td>
<td>0.035</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4.4: Results of different sub-families of Families 1 and 2 performed by Cplex and by IRP

<table>
<thead>
<tr>
<th>Family</th>
<th>ω</th>
<th>K</th>
<th>Q</th>
<th>All tested</th>
<th>Feas.</th>
<th>Av. CPU_IRP</th>
<th>#Imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>400</td>
<td>29</td>
<td>20</td>
<td>0.2038</td>
<td>&lt;I</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>27</td>
<td>20</td>
<td>0.2106</td>
<td>I</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>20</td>
<td>20</td>
<td>0.2008</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>6</td>
<td>190</td>
<td>135</td>
<td>20</td>
<td>0.199</td>
<td>I</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>3</td>
<td>400</td>
<td>56</td>
<td>20</td>
<td>0.7802</td>
<td>I</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>3</td>
<td>400</td>
<td>25</td>
<td>20</td>
<td>0.8645</td>
<td>I</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>4</td>
<td>300</td>
<td>87</td>
<td>20</td>
<td>0.8227</td>
<td>I</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>6</td>
<td>190</td>
<td>912</td>
<td>20</td>
<td>0.8418</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>400</td>
<td>142</td>
<td>20</td>
<td>1.0126</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>400</td>
<td>27</td>
<td>20</td>
<td>1.2105</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>5</td>
<td>240</td>
<td>222</td>
<td>20</td>
<td>1.0651</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>6</td>
<td>190</td>
<td>325</td>
<td>10</td>
<td>1.1589</td>
<td>I</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>3</td>
<td>400</td>
<td>28</td>
<td>20</td>
<td>19.9628</td>
<td>I</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>3</td>
<td>400</td>
<td>24</td>
<td>20</td>
<td>20.303</td>
<td>I</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
<td>190</td>
<td>204</td>
<td>20</td>
<td>18.5081</td>
<td>I</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>3</td>
<td>400</td>
<td>31</td>
<td>20</td>
<td>54.0664</td>
<td>I</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>23</td>
<td>20</td>
<td>54.3984</td>
<td>I</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>3</td>
<td>400</td>
<td>23</td>
<td>20</td>
<td>52.6137</td>
<td>I</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>5</td>
<td>240</td>
<td>130</td>
<td>20</td>
<td>47.6028</td>
<td>I</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>6</td>
<td>190</td>
<td>229</td>
<td>20</td>
<td>46.136</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>3</td>
<td>400</td>
<td>28</td>
<td>20</td>
<td>115.8591</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>3</td>
<td>400</td>
<td>34</td>
<td>20</td>
<td>119.8138</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>5</td>
<td>240</td>
<td>33</td>
<td>20</td>
<td>102.522</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>6</td>
<td>190</td>
<td>58</td>
<td>20</td>
<td>100.1831</td>
<td>I</td>
</tr>
</tbody>
</table>

Table 4.5: Results obtained by IRP for sub-families of families 3, 4, 5, 8, 9 and 10
4.2. **NUMERICAL RESULTS**

instances, in addition to the number of feasible ones, the average CPU time and the number of improvements are all given in Table 4.5 too.

In Table 4.6 we present 4 families of different properties. Detailed results for some instances of families 1 and 2 are given in Table 4.7, and the average results for these two families are given in Table 4.8. In Table 4.7 we present the number of the family, the number of the selected instance, the best lower bound given by Cplex, the solution determined by Cplex, by IRP and by TS, and the CPU consuming times needed by Cplex and by TS in the last two columns. Similarly, some detailed instances of families 3 and 4 are given in Table 4.9, and the average results of each family are given in Table 4.10. We give the average gap between the solution obtained by Cplex and the one obtained by IRP which is given by:

\[
\frac{\text{value given by IRP} - \text{value given by Cplex}}{\text{value given by Cplex}}
\]

We also compute the deviation between the IRP and TS algorithms which is calculated according to:

\[
\frac{\text{value given by IRP} - \text{value given by TS}}{\text{value given by TS}}
\]

| Family | \(|V(g)|\) | \(|E(g)|\) | \(|R|\) | \(d(g) (%)\) | \(D\) | \(\omega\) | \(K\) | \(Q\) |
|--------|------------|------------|---------|----------------|-----|-------|-----|-----|
| Family 1 | 15 | 18 | 8 | 0.171 | 100 | 6 | 3 | 40 |
| Family 2 | 20 | 25 | 8 | 0.132 | 100 | 6 | 3 | 40 |
| Family 3 | 50 | 70 | 33 | 0.057 | 1000 | 7 | 3 | 400 |
| Family 4 | 100 | 120 | 59 | 0.024 | 1000 | 4 | 3 | 400 |

Table 4.6: Family instances for TS

In the following, BLB stands for the best lower bound determined by Cplex. On one hand, the average gap represents the gap between the solution determined by Cplex and that determined by the heuristic IRP. On the other hand, the deviation represents the deviation between the heuristic IRP and the TS solutions.

The results of the tabu search algorithm (TS) are given in tables 4.7, 4.8, 4.9 and 4.10. TS algorithm gives better results than Cplex which is big time consuming especially for big size instances. It also improves the solutions obtained by IRP. Table 4.7 shows that for example the problem instance 79,


**CHAPTER 4. EFFICIENT ALGORITHMS FOR SOLVING THE SPARSE CARP**

<table>
<thead>
<tr>
<th>Family</th>
<th>Instance</th>
<th>BLB</th>
<th>Cplex</th>
<th>IRP</th>
<th>TS</th>
<th>CPU_{Cplex} (s)</th>
<th>CPU_{TS} (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family 1</td>
<td>13</td>
<td>845</td>
<td>972</td>
<td>972</td>
<td>967</td>
<td>0.37</td>
<td>1</td>
</tr>
<tr>
<td>Family 1</td>
<td>26</td>
<td>1232</td>
<td>1232</td>
<td>1259</td>
<td>1232</td>
<td>0.39</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Family 1</td>
<td>39</td>
<td>1031</td>
<td>1279</td>
<td>1484</td>
<td>1423</td>
<td>1</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Family 1</td>
<td>42</td>
<td>996</td>
<td>996</td>
<td>1617</td>
<td>1617</td>
<td>0.12</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Family 1</td>
<td>72</td>
<td>740.88</td>
<td>838</td>
<td>1075</td>
<td>886</td>
<td>0.19</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Family 2</td>
<td>16</td>
<td>1103</td>
<td>1208</td>
<td>1394</td>
<td>1296</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Family 2</td>
<td>26</td>
<td>1429</td>
<td>1493</td>
<td>2080</td>
<td>2080</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Family 2</td>
<td>49</td>
<td>710</td>
<td>806</td>
<td>1020</td>
<td>1020</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Family 2</td>
<td>55</td>
<td>898</td>
<td>961</td>
<td>1061</td>
<td>1021</td>
<td>0.43</td>
<td>1</td>
</tr>
<tr>
<td>Family 2</td>
<td>79</td>
<td>1531.125</td>
<td>1590</td>
<td>1890</td>
<td>1546</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.7: Comparative results between Cplex, IRP and TS for instances of Family 1 and Family 2; BLB: Best Lower Bound

<table>
<thead>
<tr>
<th>Family</th>
<th>Av. gap</th>
<th>Best gap</th>
<th>Worst gap</th>
<th>Av. dev.</th>
<th>Biggest dev.</th>
<th>Smallest dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family 1</td>
<td>0.0422</td>
<td>0</td>
<td>0.384</td>
<td>0.0259</td>
<td>0.4261</td>
<td>0</td>
</tr>
<tr>
<td>Family 2</td>
<td>0.0411</td>
<td>0</td>
<td>0.2822</td>
<td>0.0146</td>
<td>0.2225</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.8: Average results of families 1 and 2, dev: deviation

<table>
<thead>
<tr>
<th>Family</th>
<th>Instance</th>
<th>IRP</th>
<th>TS</th>
<th>Dev.</th>
<th>CPU_{TS} (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family 3</td>
<td>29</td>
<td>2644</td>
<td>2644</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Family 3</td>
<td>55</td>
<td>4185</td>
<td>3078</td>
<td>0.3567</td>
<td>9</td>
</tr>
<tr>
<td>Family 3</td>
<td>62</td>
<td>3255</td>
<td>3149</td>
<td>0.0337</td>
<td>3</td>
</tr>
<tr>
<td>Family 3</td>
<td>75</td>
<td>3729</td>
<td>3384</td>
<td>0.1019</td>
<td>11</td>
</tr>
<tr>
<td>Family 3</td>
<td>85</td>
<td>7404</td>
<td>3775</td>
<td>0.9613</td>
<td>15</td>
</tr>
<tr>
<td>Family 4</td>
<td>35</td>
<td>5342</td>
<td>5342</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Family 4</td>
<td>41</td>
<td>7169</td>
<td>6983</td>
<td>0.0266</td>
<td>7</td>
</tr>
<tr>
<td>Family 4</td>
<td>48</td>
<td>8022</td>
<td>7526</td>
<td>0.0659</td>
<td>43</td>
</tr>
<tr>
<td>Family 4</td>
<td>72</td>
<td>7041</td>
<td>7006</td>
<td>0.005</td>
<td>10</td>
</tr>
<tr>
<td>Family 4</td>
<td>93</td>
<td>5830</td>
<td>5754</td>
<td>0.0132</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 4.9: Comparative Results between IRP and TS for instances of Family 3 and Family 4, Dev: deviation

<table>
<thead>
<tr>
<th>Family</th>
<th>Av. dev.</th>
<th>Biggest dev.</th>
<th>Smallest dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family 3</td>
<td>0.0715</td>
<td>0.3596</td>
<td>0</td>
</tr>
<tr>
<td>Family 4</td>
<td>0.0079</td>
<td>0.0659</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.10: Average results of families 3 and 4, dev: deviation
4.3. CONCLUSION

TS gives a solution (of value 1546) better than the one found by Cplex (of value 1590) and better than the one obtained by IRP (of value 1890). Also, if we consider for example the problem instance 26, the solution found by TS (of value 1232) is optimal and equal to solution obtained by Cplex (of the same value) and better than the one obtained by IRP (of value 1259). For the big size instances presented in table 4.9 and 4.10, TS improves the results of IRP with a small CPU time.

4.3 Conclusion

In this chapter, we introduce three different approaches to study the capacitated arc routing problem over a feasible sparse graph for which we propose. To solve the problem approximately, we built a constructive heuristic algorithm (CH) to obtain an initial feasible solution for the obtained CVRP with sparse feasibility graph. Later, we proposed an improving randomized procedure (IRP) for a first improvement of the solution obtained by CH. The IRP is based on a factor of randomness and the strategy tries to improve the initial solution by a fixed number of CH runs. Finally, an adapted tabu search algorithm (TS) under a dynamic graph was proposed to solve the problem. In our TS approach, we adapted the aspiration criteria and the intensification and diversification components especially because the nature of the solution space is disconnected in general.

Our algorithm improves the quality of the obtained solutions by the first 2 heuristics (CH and IRP). The efficiency of TS is measured by the quality of the obtained solution and by the CPU computational time. In many cases, TS obtained optimal solutions and all the cases highly improved quality ones. These outputs were especially obtained big size instances.

The problem that we have studied is a new one and as an introduction to the last chapter of our thesis, we are interested in studying the sparse arc routing problem under the uncertainty case for which we consider a set of scenarios (or data uncertainty) of arcs traversals. Robust optimization is in this case a good approach to deal with the uncertainty which relates to the problem as a robust arc routing problem with a graph sparsity characteristic.
Chapter 5

Robust Sparse CARP with Uncertain Travel Costs

Introduction

Problems of applied mathematics and operations research are mostly treated by the combinatorial optimization which can be considered as one of the most active fields at such interface. This type of optimization requires the advance knowledge of the data and the parameters of the problem. In addition, the data should be deterministic. However, real life applications may have high degree of uncertainty, thus any disturbance of the input data may affect the nature of the solution to be not optimal or even infeasible. In our study and mainly in the field of routing problems, such perturbations may appear in the uncertainty of travel times due to traffic [151], required demands by clients, arrivals of new clients or even in the travel costs. In particular, we are interested in studying the sparse capacitated arc routing problem with uncertain travel costs. Such work requires to study the problem in the terms of theoretical and practical issues by investigating the extensions of the sparse CARP under uncertain data. Modeling the uncertainty in the problems of optimization has been done by several methods. Stochastic programming could be considered to be the most famous among these methods. In stochastic programming, the uncertain data is modeled as random variable with known probability distribution [162] [144]. The stochastic nature of the uncertainties and the possibility of identifying the probability distribution are two main conditions to apply the stochastic programming. However, this
is not the case for the real applications since the nature of the uncertainty is not always stochastic in addition to the fact that it is not always possible to identify the probability distribution of the data. Unto now, most studies have dealt with stochastic programming, and in particular to what may refer to our work, one may see to [73]. A branch-and-price algorithm for the capacitated arc routing problem with stochastic demands has been presented in [36], and another study is given by [165]. As a consequence of the difficulty of attaining both conditions of stochastic programming, an alternative of the latter must be sought. Such alternative is the robust optimization which has been applied to find robust solutions i.e. solutions that remain solutions upon facing unpleasant impacts which is caused by the ambiguity or the imprecision of the input data [16, 17]. Providing robust solutions addresses three main challenges: (1) evaluation, (2) adaptation of heuristic methods and (3) assessment of performance. This criteria expands to identify (1) the modeling of the uncertain data (scenarios in our study), (2) the selection of appropriate criteria (min-max in our study) and (3) the mathematical model of the problem.

In this chapter, we are interested in studying the robust sparse capacitated arc routing problem with travel costs uncertainty. We give first a brief review about the robust optimization (definitions and criteria) and introduce a brief survey about the robust optimization methods of vehicle routing problem. Second, we introduce the mathematical model of our problem which is proposed to give a robust solution minimizing the worst scenario. Then, different heuristic algorithm and metaheuristic ones are developed according to a specific procedure. Computational experiments are performed for the proposed mathematical formulation and the greedy heuristics and metaheuristics. Throughout the whole chapter, we work with a graph $G$ whose number of vertices is $n$, number of edges is $n + \alpha$ with $1 \leq \alpha \leq \frac{n}{2}$, and the maximum vertex degree held in $G$ is 3.

5.1 Review about Robust Optimization

Starting from the definition of the word “robust” which means strong, the robust optimization is sufficient to give a solution that is strong enough to face any disruption in the data [135]. The data may be exposed to some uncertainty or unpredictability especially in future, and this would really affect the optimal solution that is computed before to be at last infeasible or not
optimal. Thus a robust solution is a solution that resists as much as possible disturbances. When the uncertain data has a probabilistic description, robustness can be attained by using the stochastic optimization which has been given in [51]. However, this type of programming has two drawbacks [22, 23]:

1. The underlying probability distributions must be already known and this is not always the case.

2. The solutions can become infeasible upon facing some random events or disruptions.

However, to avoid these drawbacks, robust optimization which is not stochastic but rather deterministic and set-based is considered to be the suitable framework. The aim of such optimization is to optimize the worst case value under all uncertain data. Representing uncertain data is mostly done by a convex set as a polyhedron, a cone or an ellipsoid as in [19] and in [24]. Other structures of uncertain data is done by assigning plausible values to each model parameter in which the two common ways of such modeling are the interval or discrete scenarios [141]. In our work, we represent the uncertain data by generating discrete scenarios. The authors in [23] propose a budget of uncertainty denoted by $\Gamma$ to limit the number of uncertain parameters allowed to deviate from their nominal values. This proposition is to control the degree of conservatism as robust solutions are often considered as conservative. Their approach which is applied to the cost and constraint coefficients of a linear or mixed integer program leads to a robust version with a moderate increase in size.

Robustness criteria include several families where the decisions can be made according to min-max, min-max regret, min-max relative regret and lexicographical min-max, etc. For more details about different robustness criteria, the reader may refer to [46, 104, 101]. In [23], an alternative robust optimization criterion to the min-max using discrete scenarios is proposed. The authors present a robust integer programming model that allows to control the degree of conservatism of a solution using probabilistic bounds and violation constraints. They give an algorithm for the robust network flows using the model that they have presented. Most of the facility location problems test the min-max with discrete scenarios, for example, we may state here the robust prize-collecting Steiner tree problems [3], robust knapsack problem [141, 124] and robust network loading problem with dynamic routing [121].
For more information about robustness criteria and robust classification criteria, see [147].
In our work, the robustness criterion which we follow is the min-max criterion in which we minimize the cost whenever the worst scenario occurs.

## 5.2 Uncertain Vehicle Routing Problems

Several forms of uncertainty of different variants of the vehicle routing problem have been studied by two approaches: The stochastic programming and the robust optimization. In this section, we present a brief review about these approaches to solve the vehicle routing problem with different aspects of uncertainty.

### 5.2.1 Stochastic Vehicle Routing Problem

A stochastic vehicle routing problem is a VRP where one or several of the components of the problem are random. For instance, VRPs with stochastic customers where a customer needs to be serviced with a given probability [20,158]. Another variant is the VRP with stochastic demands where the demands of the customers are known as probability distributions [142,110,35,143,150] and this was deeply in [21]. VRPs with stochastic travel times in which the service or travel times are modeled by random variables have been presented in [111,102,157]. A major issue for using the stochastic optimization is that the probability distributions which accurately model uncertainties must be known.

### 5.2.2 Robust Vehicle Routing Problem

As mentioned in the previous paragraph, same aspects of uncertainty are handled with the robust vehicle routing problem with a main investigation of that with uncertain demands which was firstly studied in [151]. The authors derive a robust counterpart for the vehicle routing problem with stochastic demands (VRPSD) where they show that the robust solution is favorable on average compared to the deterministic solution if the network structure allows a strategic distribution of the slack in the vehicles i.e. in the case where vehicles can share their slacks. Moreover, the authors show that the robust solution is superior to simple strategies of distributing the excess capacity.
among the vehicles especially when the network structure is more clustered. A limited work has been done for the case of uncertain travel times or travel costs. A robust scenario approach for the vehicle routing problem with uncertain travel times is studied in [91]. The works in [153, 154] handle the VRP with uncertain travel costs. The total travel cost is minimized and uncertainty is expressed as intervals.

For more details about the robust vehicle routing problems and their corresponding methods of optimizations, the reader may refer to [147]. On the other hand and to the most of our knowledge, rare are the studies about solving the arc routing problems with the robust optimization method. In the next section, we introduce a brief review about the capacitated arc routing problem under uncertain environment and we present a mathematical formulation for the robust sparse capacitated arc routing problem under travel costs uncertainty.

5.3 Capacitated Arc Routing Problem under Uncertainty

Throughout the literature, the uncertain capacitated arc routing problem has been characterized by four stochastic factors: (1) the presence of tasks, (2) the demands of the tasks, (3) the services costs and (4) the availability of a path between each pair of vertices. The most of the research work considers these factors separately or combine at most two of them together. For example the presence and demands of the tasks are combined in [21]. Later, these four stochastic factors of the problem were combined all together in [122]. The authors in [122] consider the uncertainty of each of these factors with random variables and by a certain probability distribution as a function of an environmental parameter. Moreover, they introduce a mathematical formulation for the problem. The developed algorithms by them showed excellent performance for static CARP, but they were not able to find robust solutions for the uncertain CARP.
5.3.1 Robust Capacitated Arc Routing Problem under sparse graph and under Travel Costs Uncertainty

The uncertain capacitated arc routing problem that we study is characterized by the uncertainty in the travel costs and by the sparse underlying graph on which it is defined. This uncertainty is represented by a finite set of scenarios where each required edge of the network has a different cost with respect to each scenario. We aim at determining a robust solution i.e. in an uncertain environment, the problem objective is no longer to find a single global optimal solution, but to find a solution with the best expected quality under all possible environments. In the following, we present a mathematical model of the capacitated arc routing problem over sparse graph under travel costs uncertainty. In other terms, we are concerned with the multiple-scenario min-max capacitated arc routing problem under sparse graphs.

**Remark 5.3.1.** We apply the robust optimization directly on the sparse capacitated arc routing problem, and not on the equivalent sparse CVRP by the proposed transformation in Section 3.4. This is due to the following:

A graphical representation of robust sparse CARP is given by a sparse multigraph in which each edge is represented \(s\) times where \(s\) denotes the number of scenarios. In other terms, the underlying graph of a robust sparse CARP under travel costs uncertainty is a multigraph. Each multiple edge represents a route with different costs according to each scenario. Upon transforming this multigraph into its corresponding line graph, the sparsity property will be lost. Thus, sparsity is not conserved in this case. For this reason, we choose to solve the sparse CARP problem directly without applying any transformation.

Throughout the following, let \(G = (V, E)\) be a graph where \(V\) denotes the set of vertices and \(E\) the set of edges. Denote by \(R \subseteq E\) the set of the required edges i.e. the set of edges having strictly positive demands to be serviced. Consider the following notations and variables:

- \(K\): the total number of the vehicles.
- \(Q\): the capacity of each vehicle.
- \(d_e\): the demand of the edge \(e\).
5.3. CAPACITATED ARC ROUTING PROBLEM UNDER UNCERTAINTY

- $\Delta_e$: the total demand served by the vehicle arriving at the service $e$ including the demand of $e$ itself which is by definition less than or equal to $Q$.
- $c^s_e$: the cost of the edge $e$ in scenario $s$.
- $N(e)$: the neighborhood of the edge $e$.
- $S = \{1, 2, \ldots, P\}$: the set of scenarios.
- $\omega_e$: the capacity of edge $e$ i.e. the maximum number of times for which an edge can be traversed.
- $x^e_{e', f'}$: a binary variable which is equal to 1 if and only if the service at $f'$ is successive to the service at $e'$ by the same vehicle, and the chosen shortest path between $e'$ and $f'$ includes the consecutive adjacent edges $e$ and $f$, and 0 otherwise.
- $y^e_{e', f'}$: a binary variable equal to 1 if $f'$ is serviced directly after $e'$, and 0 otherwise.

Recall that the graphs which we are working over are sparse with maximum degree equal to 3. Denote by 0 and 1 two incident edges to the depot node. The edge 0 denotes the edge of departure of the vehicles i.e. the exit from the depot, and 1 denotes the edge of returning back of the vehicle i.e. the entrance to the depot after accomplishing all the services of the corresponding vehicle. Moreover, we assume that these edges are required but with a null demand.

Mathematical Formulation

\[
\min \max_{s \in S} \sum_{e', f' \in R, e \in E, f \in N(e)} c^s_e x^e_{e', f'}
\]

subject to:

\[
y^e_{e', f'} + y^f_{f', e'} \leq 1 \quad \forall e', f' \in R
\]

\[
\sum_{f' \in R} y^0_{0, f'} = K
\]
CHAPTER 5. ROBUST SPARSE CARP WITH UNCERTAIN TRAVEL COSTS

\[ \sum_{f' \in R} y_{1,f'} = 0 \]  
(5.4)

\[ \sum_{e' \in R} y_{e',1} = K \]  
(5.5)

\[ \sum_{f' \in R} y_{e',0} = 0 \]  
(5.6)

\[ \sum_{f' \in R} y_{e',f'} = 1 \text{ if } e' \neq \{0,1\} \]  
(5.7)

\[ \sum_{e' \in R} y_{e',f'} = 1 \text{ if } f' \neq \{0,1\} \]  
(5.8)

\[ \Delta_{f'} \geq \Delta_{e'} + d_{f'} + (d_{f'} + Q) \times (y_{e',f'} - 1) \quad \forall e', f' \in R, e' \neq f' \]  
(5.9)

\[ \sum_{f \in N(e)} x_{e,f} - \sum_{f \in N(e)} x_{f,e} = 0 \quad \text{if } e \neq e', e \neq f', e', f' \in R \]  
(5.10)

\[ \sum_{f \in N(e)} x_{e,f} - \sum_{f \in N(e)} x_{f,e} = y_{e',f'} \quad \text{if } e = e', e', f' \in R \]  
(5.11)

\[ \sum_{f \in N(e)} x_{e,f} - \sum_{f \in N(e)} x_{f,e} = -y_{e',f'} \quad \text{if } e = f', e', f' \in R \]  
(5.12)

\[ \sum_{e',f' \in R, f \in N(e)} x_{e',f'} \leq \omega_e \quad \text{with } \omega_e \geq 1 \quad \text{if } e \neq 0 \]  
(5.13)

\[ \sum_{e',f' \in R, f \in N(0)} x_{0,f} = K \]  
(5.14)

\[ \sum_{e',f' \in R, f \in N(1)} x_{f,1} \leq \omega_e \quad \text{with } \omega_e \geq 1 \quad \text{if } e \neq 1 \]  
(5.15)

\[ \sum_{e',f' \in R, f \in N(1)} x_{f,1} = K \]  
(5.16)
5.4 Efficient algorithms for solving robust sparse CARP

In this section, we present a heuristic algorithm for solving the robust sparse capacitated arc routing problem under travel costs uncertainty. The initial solution which is obtained by this algorithm is then ameliorated by a well adapted tabu search algorithm.

5.4.1 A heuristic algorithm for solving the robust sparse CARP under travel costs uncertainty

This heuristic ends with a feasible initial solution of the problem. The procedure locates a worst scenario $\bar{S}$ and computes $Z(\bar{X}) = \max \sum c_{e,f}^S x_{e,f}$.

Let $e_1, e_2, \ldots, e_r$ be the required edges, and denote by $\lambda_{e_i}$ the efficiency of each edge $e_i$ which is given by the formula:

$$\lambda_{e_i} = \frac{\sum_{s \in S} c_{e_i}^S}{d_{e_i}}, \quad (5.18)$$
where $d_{e_i}$ denotes the demand of the required edge $e_i$.

This algorithm is valid for the two cases of $\omega = 1$, where each edge can be traversed one only time, and $\omega > 1$, where there is a constant maximal number of times for traversing an edge. The only difference between the two cases lies mainly in the procedure $\text{Update}$.

**Algorithm GH** (Greedy Heuristic)

**Input:** A Robust CARP Instance

**Output:** A feasible solution $\bar{X}$ and the corresponding worst scenario $\bar{S}$

---

**Initialization**

0. $E = R \cup NR; |E| = n + \alpha$
   
   $R = \{e_1, e_2, e_3, \ldots, e_{r-1}, e_r\}$
   
   $NR = \{f_1, f_2, \ldots, f_{m-r}\}$
   
   1. For $j = 2$ to $r - 1$
      
      Sort the required edges in the non-decreasing order of the efficiencies $\lambda_{e_j}$
      
      End For

2. Set $\bar{X} \leftarrow \emptyset; Z(\bar{X}) = +\infty; \bar{S} = 1$

**Main Steps**

1. $k \leftarrow 1$

2. While ($R <> \emptyset$) and $k <= K$ Do

3. $\Delta \leftarrow d_{e_2}; j \leftarrow 3; P(1) \leftarrow e_1; P(2) \leftarrow e_2; i \leftarrow 3; P \leftarrow \{e_1, e_2\}$

4. While ($((\Delta <= Q)$ and $j < r)$ Do

5. Subpath($e_j$)

6. End While

7. $P(i) \leftarrow e_r$

8. $\text{dim} P \leftarrow i$

9. $l(i) \leftarrow \text{dim} P$

10. Complete($P$)

11. $X \leftarrow \bar{X} \cup P$

12. Let $\bar{S} = \max_{1 \leq s \leq |S|} \{Z^s(\bar{X})\}$

13. $\text{Update}(R, NR)$

14. $k \leftarrow k + 1$

15. End While

16. Exit with a feasible solution $\bar{X}$ and with the corresponding worst scenario $\bar{S}$

---

Figure 5.1: A greedy heuristic algorithm for determining a starting feasible solution of the robust sparse CARP
In the following, we explain the steps of the previous greedy algorithm detailed in Figure 5.1.

- **Step 1**: $k \leftarrow 1$ to start with the first vehicle.

- **Step 2**: while the set of the required edges is not empty, and the number of the vehicles is less than or equal to the number of the available ones.

- **Step 3**: the total accumulated demand is fixed at the demand of the first required edge with strictly positive demand and greatest efficiency. The first edge in the path is the depot. The second edge in the path is the first required edge with a strictly positive demand.

- **Step 4**: while the accumulated demand respects the capacity of the vehicle and there are still required non-serviced edges.

- **Step 5**: call the procedure `Subpath()`, Figure 5.2, that tests if adding
CHAPTER 5. ROBUST SPARSE CARP WITH UNCERTAIN TRAVEL COSTS

116

1. For \((i = 2\) to \(\text{dim } P - 1\)) Do
2. \hspace{1em} For \((j = 1\) to \(|NR|\)) Do
3. \hspace{2em} If \((P(i) = (NR)_j)\) then
4. \hspace{3em} \(\omega((NR)_j) \leftarrow \omega((NR)_j) - 1;\)
5. \hspace{2em} End If
6. \hspace{1em} End For
7. \hspace{1em} For \((j = 2\) to \(r - 1\)) Do
8. \hspace{2em} If \((P(i) = e_j)\) then
9. \hspace{3em} \(\beta \leftarrow \omega(e_j);\)
10. \hspace{3em} \(a \leftarrow e_j;\)
11. \hspace{3em} For \((l(i) = j\) to \(r - 1\)) Do
12. \hspace{4em} \(e_{l(i)} \leftarrow e_{l(i) + 1};\)
13. \hspace{3em} End For
14. \hspace{1em} \(|NR| \leftarrow |NR| + 1;\)
15. \hspace{1em} \(NR(|NR|) \leftarrow a;\)
16. \hspace{1em} \(\omega(a) \leftarrow \alpha - 1;\)
17. \hspace{1em} \(r \leftarrow r - 1;\)
18. \hspace{1em} End If
19. \hspace{1em} End For
20. End For

Figure 5.4: The procedure Update for \(\omega > 1\)

the \(j^{th}\) required edge will not violate the capacity constraint. In this case, update the accumulated demand to be the last one added to the demand of \(j\), and place this edge in the \(i^{th}\) rank of the constructed subpath. Then, move to the next rank and then to the next required edge.

-Step 7 to Step 9: Once adding a required edge could violate the capacity, go back to the depot. The dimension of the constructed subpath is \(i\) where the depot entrance is the \(i^{th}\)-edge of this subpath i.e. the last edge. Assignment of the dimension of \(P\) to an auxiliary variable \(l(i)\).

-Step 10: call the procedure Complete, Figure 5.3. The procedure Complete tests if the predecessor of each required edge in \(P\) say at rank \(l(i)\) is not the required edge served in this path \(P\) and placed at the rank \(l(i) - 1\), then call Dijkstra and insert a shortest path of made of edges of \(NR\) between the edge at rank \(l(i)\) and the edge at rank \(l(i) - 1\). Each time an edge is inserted, the dimension of the path \(P\) is incremented by 1 and each edge at rank \(j - 1\)
5.4. EFFICIENT ALGORITHMS FOR SOLVING ROBUST SPARSE CARP

will be at the rank \( j \) to let the inserted edge compensate the emptied rank.

- **Step 11**: Update the constructed solution.
- **Step 12**: Determine the worst scenario that corresponds to the scenario giving the maximal solution cost.

- **Step 13**: Update the sets \( NR \) and \( R \): in the case where \( \omega \leq 1 \), Figure 5.4, the capacity \( \omega \) of any used edge in the path \( P \) is decremented by 1 each time the edge is used. The same is applied to the edges of the set \( R \) with the additional step that will be impose the removal of the used edges from \( R \) and then added to \( NR \). For the case of \( \omega = 1 \), Figure 5.5, the edges of both sets are removed once traversed or served.

- **Step 14**: Move to the next vehicle.

We determine by this heuristic algorithm an initial robust solution of the problem and its corresponding worst scenario.
5.4.2 An adapted tabu search algorithm for solving the robust sparse CARP under travel costs uncertainty

In this part, we develop an adapted tabu search algorithm for the robust sparse CARP under travel costs uncertainty. This algorithm starts with the initial solution that is determined by the above greedy heuristic.

**Algorithm TS**

**Input:** An initial feasible solution $\bar{X}$

**Output:** An efficient solution $X^\ast$ with the corresponding worst scenario $S^\ast$.

---

**Initialization**

0. $X^\ast \leftarrow \bar{X}$;
1. $L \leftarrow \phi$;
2. $Iter \leftarrow 0$;

**Main Steps**

1. While $(Iter < MaxIter)$ Do
2. If $(|L| \leq Th)$ Then
3. Build$_1$(N($X^\ast$));
4. If $(|L| > Th)$ Then
5. Build$_2$(N($X^\ast$));
6. For ($t = 1$ to $|N(X^\ast)|$)
7. If $((Z(X^\ast_t) \leq Z(X^\ast)) \& \& (X^\ast_t \notin L))$ Then
8. $X^\ast \leftarrow X^\ast_t$;
9. End If
10. Update $L, Iter$;
11. End For
12. End While
13. Exit with solution $X^\ast$ and with the corresponding worst scenario $S^\ast$

Figure 5.6: A tabu search algorithm for determining best feasible solution of the robust sparse CARP

Consider the following notations:

- $\bar{X}$: initial feasible solution with the corresponding worst scenario.
- $X^\ast$: best feasible solution determined by the tabu search algorithm.
5.4. EFFICIENT ALGORITHMS FOR SOLVING ROBUST SPARSE CARP

- $L$: the tabu list.
- $Iter$: number of iteration.
- $MaxIter$: maximum number of iterations.
- $N(X^*)$: neighborhood about the solution $X^*$.
- $S^*$: the worst scenario determined by the algorithm.
- $Th$: a certain threshold.

The algorithm contains several steps. Tabu search starts by an initial feasible solution obtained thanks to the greedy heuristic algorithm. All the visited solutions are feasible. The exploration of the solutions space is executed with some swaps. The elite solutions list is generated by improving the objective value where the worst scenario has already been identified. The core of the approach is to build neighborhoods and perform several local searches in order to reach a best solution. In Step 2 of the main steps of Figure 5, if a certain threshold $Th$ is not attained, we diversify the search by using the procedure $\text{Build}_1$ to build the neighborhood. In this step, we choose randomly two vehicles, $vehicle_1$ and $vehicle_2$, and we select two services of each chosen vehicle i.e. $service_1^1$, $service_2^2$, $service_1^2$ and $service_2^2$. We check whether the swap of these services (the first service of the first vehicle with the first service of the second vehicle, and the second service of the first vehicle with the second service of the second vehicle) respects the capacity of the vehicles, and we swap them as explained. In this way, we explore the neighbors and we choose the best that minimizes the cost for the worst scenario. The search progresses by iteratively moving from the current solution to an improved solution. In Step 4 of the main steps of this Figure, if the threshold $Th$ is attained, we intensify the search by using the procedure $\text{Build}_2$ to build the neighborhood which allows the exchange of two services of the same vehicle. The tabu based strategy incorporates a tabu list in the selection mechanism that forbids the selection of the non-improving solution for a certain tabu tenure. Each visited explored solution is then settled in the tabu list $L$ to not be visited again unless the tabu list reaches its expiration point i.e. the tabu status of a move is removed if it belongs to the list $L$ and it exceeds $MaxIter$ iterations.
CHAPTER 5. ROBUST SPARSE CARP WITH UNCERTAIN TRAVEL COSTS

For the intensification and the diversification of the search, both are achieved via the procedures $\text{Build}_1$ and $\text{Build}_2$.

5.5 Computational Experiments

In this section, we introduce a set of computational experiments for which we apply each of these algorithms: the heuristic algorithm and the tabu search one. The benchmark of instances has not been treated before. We generate randomly our own benchmark of instances. We have run all these instances with different number of scenarios and different densities of their corresponding networks in order to evaluate the performance of each of the presented algorithms. The proposed algorithms are coded in C++ and run on HP Intel(R) Core(TM) i7 laptop (with 2.80 Ghz and 16 Go of Ram). These test problem instances are studied for the first time and their optimal solution values are not known. None of the previous algorithms in the literature deals with such type of instances. We work with networks having a maximum degree of 3 as we deal with a sparse network. Consider the following notations:

- $WS_H$: the worst scenario which is determined by the heuristic algorithm.
- $Cost_H$: the cost of the solution which is determined by the heuristic algorithm.
- $CPU_H$: the time needed by the heuristic algorithm to determine a solution.
- $WS_T$: the worst scenario which is determined by the tabu algorithm.
- $Cost_T$: the cost of the solution which is determined by the tabu algorithm.
- $CPU_T$: the time needed by the tabu algorithm to determine a solution.

The robust optimization that we apply via the developed algorithms allows us not only to get a robust solution, but also it gives us the worst scenario that may change upon the improvement of the solution as we observe in the following tables. In other terms, a worst scenario of a solution determined by the heuristic algorithm is not necessarily the same worst scenario of the
solution obtained by the tabu algorithm after improvement. As a result, an improvement comes in two directions: (1) obtaining a better solution with a minimal cost and (2) improving the corresponding worst scenario.

In what follows, we show the tables of the numerical results that we have generated.

In Tables 5.1, 5.3, 5.5, 5.7, 5.9 and 5.11, we present sets of different families for the robust CARP instances. The number of vertices, the number of edges, the number of required edges (services), the number of scenarios, the number of the available homogenous vehicles and their capacity are all given. The results in Tables 5.2, 5.4, 5.6, 5.8, 5.10 and 5.12 present the worst scenario determined by the heuristic, the cost of the solution and the CPU consuming time of the heuristic too. Furthermore, it gives the worst scenario determined by the tabu search algorithm, the cost of the corresponding solution and the CPU consuming time of the tabu search algorithm.

<table>
<thead>
<tr>
<th>Instance</th>
<th></th>
<th>V(G)</th>
<th></th>
<th>E(G)</th>
<th></th>
<th>R</th>
<th>S K Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A0</td>
<td>10</td>
<td>13</td>
<td>5</td>
<td>10</td>
<td>2</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>2A0</td>
<td>20</td>
<td>27</td>
<td>10</td>
<td>10</td>
<td>3</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>3A0</td>
<td>50</td>
<td>70</td>
<td>25</td>
<td>10</td>
<td>5</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>4A0</td>
<td>100</td>
<td>150</td>
<td>59</td>
<td>10</td>
<td>5</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>5A0</td>
<td>231</td>
<td>331</td>
<td>121</td>
<td>10</td>
<td>10</td>
<td>130</td>
<td></td>
</tr>
<tr>
<td>6A0</td>
<td>257</td>
<td>362</td>
<td>191</td>
<td>10</td>
<td>12</td>
<td>190</td>
<td></td>
</tr>
<tr>
<td>7A0</td>
<td>307</td>
<td>439</td>
<td>260</td>
<td>10</td>
<td>12</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>8A0</td>
<td>400</td>
<td>600</td>
<td>350</td>
<td>10</td>
<td>15</td>
<td>240</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: A first set of different families of the robust CARP problem instances - Group A0

On one hand, we notice that our greedy heuristic succeeds to have the access to all the studied instances with a very small CPU consuming time regardless the quality of the found solution. On the other hand, the tabu algorithm did not succeed to have the access to the big size instances.

Recall that G denotes a graph where V(G) denotes the set of vertices and |V(G)| its cardinality, E(G) denotes the set of edges and |E(G)| its cardinality. The set of required edges is represented by R, and the set of different scenarios is represented by S. The used fleet of vehicles is homogeneous where K denotes the number of the available vehicles and Q represents the capacity of each one. The different instances are generated randomly i.e. the
<table>
<thead>
<tr>
<th>Instance</th>
<th>$WS_H$</th>
<th>$Cost_H$</th>
<th>$CPU_H(s)$</th>
<th>$WS_T$</th>
<th>$Cost_T$</th>
<th>$CPU_T(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A0</td>
<td>9</td>
<td>1105</td>
<td>0.001</td>
<td>9</td>
<td>1105</td>
<td>0.001</td>
</tr>
<tr>
<td>2A0</td>
<td>10</td>
<td>3534</td>
<td>0.008</td>
<td>7</td>
<td>1943</td>
<td>1.6</td>
</tr>
<tr>
<td>3A0</td>
<td>9</td>
<td>11476</td>
<td>0.01</td>
<td>6</td>
<td>5025</td>
<td>15</td>
</tr>
<tr>
<td>4A0</td>
<td>2</td>
<td>27274</td>
<td>0.056</td>
<td>2</td>
<td>16037</td>
<td>41</td>
</tr>
<tr>
<td>5A0</td>
<td>8</td>
<td>71083</td>
<td>0.241</td>
<td>8</td>
<td>46302</td>
<td>218</td>
</tr>
<tr>
<td>6A0</td>
<td>4</td>
<td>120473</td>
<td>0.416</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>7A0</td>
<td>3</td>
<td>183109</td>
<td>0.723</td>
<td>3</td>
<td>126440</td>
<td>640</td>
</tr>
<tr>
<td>8A0</td>
<td>2</td>
<td>255540</td>
<td>1.853</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
</tbody>
</table>

Table 5.2: Results of Group A Instances by the heuristic algorithm and by the tabu search algorithm

$NI$: no improvement

| Instance | $|V(G)|$ | $|E(G)|$ | $|R|$ | $S$ | $K$ | $Q$ |
|----------|--------|--------|------|-----|-----|-----|
| 1A       | 10     | 13     | 5    | 10  | 2   | 15  |
| 2A       | 20     | 27     | 10   | 10  | 2   | 40  |
| 3A       | 50     | 70     | 25   | 10  | 3   | 50  |
| 4A       | 100    | 150    | 59   | 10  | 3   | 120 |
| 5A       | 231    | 331    | 121  | 10  | 4   | 120 |
| 6A       | 257    | 362    | 191  | 10  | 5   | 150 |
| 7A       | 307    | 439    | 260  | 10  | 7   | 150 |
| 8A       | 400    | 600    | 350  | 10  | 7   | 150 |

Table 5.3: A second set of different families of the robust CARP problem instances - Group A

<table>
<thead>
<tr>
<th>Instance</th>
<th>$WS_H$</th>
<th>$Cost_H$</th>
<th>$CPU_H(s)$</th>
<th>$WS_T$</th>
<th>$Cost_T$</th>
<th>$CPU_T(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>10</td>
<td>1295</td>
<td>0.001</td>
<td>10</td>
<td>1059</td>
<td>0.5</td>
</tr>
<tr>
<td>2A</td>
<td>1</td>
<td>3104</td>
<td>0.008</td>
<td>1</td>
<td>2644</td>
<td>1.52</td>
</tr>
<tr>
<td>3A</td>
<td>6</td>
<td>10169</td>
<td>0.012</td>
<td>6</td>
<td>5658</td>
<td>8.326</td>
</tr>
<tr>
<td>4A</td>
<td>2</td>
<td>27370</td>
<td>0.056</td>
<td>10</td>
<td>17735</td>
<td>40.96</td>
</tr>
<tr>
<td>5A</td>
<td>3</td>
<td>46461</td>
<td>0.236</td>
<td>3</td>
<td>31074</td>
<td>124</td>
</tr>
<tr>
<td>6A</td>
<td>1</td>
<td>84875</td>
<td>0.392</td>
<td>5</td>
<td>50972</td>
<td>208</td>
</tr>
<tr>
<td>7A</td>
<td>5</td>
<td>124241</td>
<td>0.664</td>
<td>6</td>
<td>76389</td>
<td>357</td>
</tr>
<tr>
<td>8A</td>
<td>5</td>
<td>118039</td>
<td>1.168</td>
<td>5</td>
<td>71498</td>
<td>450</td>
</tr>
</tbody>
</table>

Table 5.4: Results of Group A Instances by the heuristic algorithm and by the tabu search algorithm
## 5.5. COMPUTATIONAL EXPERIMENTS

| Instance | $|V(G)|$ | $|E(G)|$ | $|R|$ | $S$ | $K$ | $Q$ |
|----------|--------|--------|------|-----|-----|-----|
| 1B0      | 10     | 13     | 5    | 40  | 2   | 30  |
| 2B0      | 20     | 27     | 10   | 40  | 3   | 40  |
| 3B0      | 50     | 70     | 25   | 40  | 5   | 60  |
| 4B0      | 100    | 150    | 59   | 40  | 5   | 120 |
| 5B0      | 231    | 331    | 121  | 40  | 10  | 130 |
| 6B0      | 257    | 362    | 191  | 40  | 12  | 190 |
| 7B0      | 307    | 439    | 260  | 40  | 12  | 225 |
| 8B0      | 400    | 600    | 350  | 40  | 15  | 240 |

Table 5.5: A third set of different families of the robust CARP problem instances - Group B

<table>
<thead>
<tr>
<th>Instance</th>
<th>$WS_H$</th>
<th>$Cost_H$</th>
<th>$CPU_H$</th>
<th>$WS_T$</th>
<th>$Cost_T$</th>
<th>$CPU_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1B0</td>
<td>10</td>
<td>1273</td>
<td>0.001</td>
<td>10</td>
<td>1273</td>
<td>0.001</td>
</tr>
<tr>
<td>2B0</td>
<td>6</td>
<td>2650</td>
<td>0.0016</td>
<td>38</td>
<td>1927</td>
<td>6</td>
</tr>
<tr>
<td>3B0</td>
<td>30</td>
<td>9807</td>
<td>0.052</td>
<td>24</td>
<td>8206</td>
<td>33</td>
</tr>
<tr>
<td>4B0</td>
<td>30</td>
<td>28178</td>
<td>0.204</td>
<td>30</td>
<td>15802</td>
<td>287</td>
</tr>
<tr>
<td>5B0</td>
<td>8</td>
<td>66736</td>
<td>0.936</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>6B0</td>
<td>10</td>
<td>118841</td>
<td>1.668</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>7B0</td>
<td>23</td>
<td>178188</td>
<td>2.668</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>8B0</td>
<td>12</td>
<td>242994</td>
<td>4.752</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
</tbody>
</table>

Table 5.6: Results of Group B0 Instances by the heuristic algorithm and by the tabu search algorithm

$NI$: no improvement

| Instance | $|V(G)|$ | $|E(G)|$ | $|R|$ | $S$ | $K$ | $Q$ |
|----------|--------|--------|------|-----|-----|-----|
| 1B        | 10     | 13     | 5    | 40  | 2   | 15  |
| 2B        | 20     | 27     | 10   | 40  | 2   | 40  |
| 3B        | 50     | 70     | 25   | 40  | 3   | 50  |
| 4B        | 100    | 150    | 59   | 40  | 3   | 120 |
| 5B        | 231    | 331    | 121  | 40  | 4   | 120 |
| 6B        | 257    | 362    | 191  | 40  | 5   | 150 |
| 7B        | 307    | 439    | 260  | 40  | 7   | 150 |
| 8B        | 400    | 600    | 350  | 40  | 7   | 150 |

Table 5.7: A fourth set of different families of the robust CARP problem instances - Group B
CHAPTER 5. ROBUST SPARSE CARP WITH UNCERTAIN TRAVEL COSTS

<table>
<thead>
<tr>
<th>Instance</th>
<th>$WS_H$</th>
<th>$Cost_H$</th>
<th>$CPU_H$</th>
<th>$WS_T$</th>
<th>$Cost_T$</th>
<th>$CPU_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1B</td>
<td>20</td>
<td>1607</td>
<td>0.002</td>
<td>18</td>
<td>1529</td>
<td>2.216</td>
</tr>
<tr>
<td>2B</td>
<td>25</td>
<td>3594</td>
<td>0.052</td>
<td>15</td>
<td>1969</td>
<td>5.944</td>
</tr>
<tr>
<td>3B</td>
<td>1</td>
<td>9989</td>
<td>0.06</td>
<td>33</td>
<td>5439</td>
<td>35.152</td>
</tr>
<tr>
<td>4B</td>
<td>39</td>
<td>28120</td>
<td>0.305</td>
<td>37</td>
<td>16389</td>
<td>169</td>
</tr>
<tr>
<td>5B</td>
<td>22</td>
<td>43603</td>
<td>0.852</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>6B</td>
<td>21</td>
<td>87199</td>
<td>1.723</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>7B</td>
<td>8</td>
<td>119613</td>
<td>2.683</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>8B</td>
<td>31</td>
<td>152818</td>
<td>2.8</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
</tbody>
</table>

Table 5.8: Results of Group B Instances by the heuristic algorithm and by the tabu search algorithm

$NI$: no improvement

$|V(G)|$ | $|E(G)|$ | $R$ | $S$ | $K$ | $Q$

<table>
<thead>
<tr>
<th>Instance</th>
<th></th>
<th></th>
<th>5</th>
<th>100</th>
<th>2</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>1C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>10</td>
<td>13</td>
<td>5</td>
<td>100</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>2C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>20</td>
<td>27</td>
<td>10</td>
<td>100</td>
<td>3</td>
<td>40</td>
</tr>
<tr>
<td>3C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>50</td>
<td>70</td>
<td>25</td>
<td>100</td>
<td>5</td>
<td>60</td>
</tr>
<tr>
<td>4C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>100</td>
<td>150</td>
<td>59</td>
<td>100</td>
<td>5</td>
<td>120</td>
</tr>
<tr>
<td>5C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>231</td>
<td>331</td>
<td>121</td>
<td>100</td>
<td>10</td>
<td>130</td>
</tr>
<tr>
<td>6C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>257</td>
<td>362</td>
<td>191</td>
<td>100</td>
<td>12</td>
<td>190</td>
</tr>
<tr>
<td>7C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>307</td>
<td>439</td>
<td>260</td>
<td>100</td>
<td>12</td>
<td>225</td>
</tr>
<tr>
<td>8C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>400</td>
<td>600</td>
<td>350</td>
<td>100</td>
<td>15</td>
<td>240</td>
</tr>
</tbody>
</table>

Table 5.9: A fifth set of different families of the robust CARP problem instances - Group $C_0$

<table>
<thead>
<tr>
<th>Instance</th>
<th>$WS_H$</th>
<th>$Cost_H$</th>
<th>$CPU_H$</th>
<th>$WS_T$</th>
<th>$Cost_T$</th>
<th>$CPU_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>62</td>
<td>1250</td>
<td>0.016</td>
<td>62</td>
<td>1250</td>
<td>0.016</td>
</tr>
<tr>
<td>2C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>89</td>
<td>2746</td>
<td>0.028</td>
<td>89</td>
<td>1767</td>
<td>28</td>
</tr>
<tr>
<td>3C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>73</td>
<td>12762</td>
<td>0.144</td>
<td>29</td>
<td>6389</td>
<td>153</td>
</tr>
<tr>
<td>4C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>81</td>
<td>27257</td>
<td>0.472</td>
<td>29</td>
<td>15634</td>
<td>731</td>
</tr>
<tr>
<td>5C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>6</td>
<td>72386</td>
<td>2.264</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>6C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>2</td>
<td>112454</td>
<td>3.94</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>7C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>95</td>
<td>189989</td>
<td>6.556</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>8C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>10</td>
<td>296212</td>
<td>12.196</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
</tbody>
</table>

Table 5.10: Results of Group $C_0$ Instances by the heuristic algorithm and by the tabu search algorithm

$NI$: no improvement
# 5.5. COMPUTATIONAL EXPERIMENTS

| Instance | $|V(G)|$ | $|E(G)|$ | $|R|$ | $S$ | $K$ | $Q$ |
|----------|--------|--------|--------|-----|-----|-----|
| Instance 1C | 10 | 13 | 5 | 100 | 2 | 15 |
| Instance 2C | 20 | 27 | 10 | 100 | 2 | 40 |
| Instance 3C | 50 | 70 | 25 | 100 | 5 | 60 |
| Instance 4C | 100 | 150 | 59 | 100 | 5 | 120 |
| Instance 5C | 231 | 331 | 121 | 100 | 7 | 120 |
| Instance 6C | 257 | 362 | 191 | 100 | 12 | 190 |
| Instance 7C | 307 | 439 | 260 | 100 | 12 | 225 |
| Instance 8C | 400 | 600 | 350 | 100 | 15 | 240 |

Table 5.11: A sixth set of different families of the robust CARP problem instances - Group C

<table>
<thead>
<tr>
<th>Instance</th>
<th>$WS_H$</th>
<th>$Cost_H$</th>
<th>$CPU_H$</th>
<th>$WS_T$</th>
<th>$Cost_T$</th>
<th>$CPU_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1C</td>
<td>48</td>
<td>1548</td>
<td>0.007</td>
<td>48</td>
<td>1216</td>
<td>4.872</td>
</tr>
<tr>
<td>2C</td>
<td>49</td>
<td>3655</td>
<td>0.27</td>
<td>49</td>
<td>2520</td>
<td>15.556</td>
</tr>
<tr>
<td>3C</td>
<td>32</td>
<td>9924</td>
<td>0.241</td>
<td>79</td>
<td>5689</td>
<td>101.084</td>
</tr>
<tr>
<td>4C</td>
<td>42</td>
<td>27870</td>
<td>0.452</td>
<td>54</td>
<td>18180</td>
<td>441.576</td>
</tr>
<tr>
<td>5C</td>
<td>66</td>
<td>45412</td>
<td>1.624</td>
<td>$NI$</td>
<td>$NI$</td>
<td>$NI$</td>
</tr>
<tr>
<td>6C</td>
<td>44</td>
<td>91616</td>
<td>3.81</td>
<td>$NI$</td>
<td>$NI$</td>
<td>$NI$</td>
</tr>
<tr>
<td>7C</td>
<td>13</td>
<td>149114</td>
<td>5.318</td>
<td>$NI$</td>
<td>$NI$</td>
<td>$NI$</td>
</tr>
<tr>
<td>8C</td>
<td>79</td>
<td>108750</td>
<td>8.375</td>
<td>$NI$</td>
<td>$NI$</td>
<td>$NI$</td>
</tr>
</tbody>
</table>

Table 5.12: Results of Group C Instances by the heuristic algorithm and by the tabu search algorithm

$NI$: no improvement
sparse graph is generated randomly but respecting that $|E(G)| = |V(G)| + \alpha$ with $1 \leq \alpha \leq \frac{|V(G)|}{2}$ and that the maximum degree in this network is 3. The costs over the scenarios are all generated randomly too.

The numerical instances are divided into 3 groups and each group into two parts: Groups $A_0$ and $A$ with 10 scenarios (Tables 5.1, 5.3), Groups $B_0$ and $B$ with 40 scenarios (Tables 5.5, 5.7) and Groups $C_0$ and $C$ with 100 scenarios (Tables 5.9, 5.11). The reason beyond this decomposition of instances comes after we have noticed that not all the available vehicles are used in the solution, thus we generated instances with a smaller number of vehicles to observe whether this factor may affect the solution.

Consider the instances of Groups $A_0$ and $A$ where we have a relatively small number of scenarios (10 scenarios). It is obvious that both algorithms perform well whatever the size of the instance is i.e. Instances 1$A_0$, 1$A$, 2$A_0$, 2$A$, 3$A_0$ and 3$A$ which are considered as small size instances are solved rapidly (Tables 5.2, 5.4). The gap between the solution determined by the heuristic and that determined by the tabu is relatively high (between 40% and 80%), and this shows that the corresponding tabu search algorithm is efficient and it can ameliorate the solution very well. We have to recall here that we aim at determining a robust solution for the problem despite of the high quality of this solution. Medium size instances (Instances 4$A_0$ and 4$A$, Tables 5.2, 5.4) need a small CPU consuming time to be solved by the heuristic, while they require more time to be solved by the tabu. Big size instances (Instances 5$A_0$, 5$A$, 6$A_0$, 6$A$, 7$A_0$, 7$A$, 8$A_0$ and 8$A$, Tables 5.2, 5.4) are solved rapidly by the heuristic just like the other instances, while it is not the case for the tabu search algorithm that needs more time to perform. As a brief conclusion for the first group of instances, the greedy heuristic algorithms behaves almost in the same way for all the instances where it determines a solution of the problem within a very small consuming time. However, as the size of the studied problem instance increases, the consuming time needed by the tabu search algorithm increases too, though it ameliorates the quality of the solution obviously. Concerning the effect of the number of the available vehicles $K$ on the solution, we observe that for some big size instances, as $K$ decreases, it becomes easier to ameliorate the initial solution as seen for instances 6$A_0$, 8$A_0$, 6$A$ and 8$A$. Moreover, the CPU consuming time of the tabu algorithm decreases with the decrease of $K$. 

CHAPTER 5. ROBUST SPARSE CARP WITH UNCERTAIN TRAVEL COSTS

126
For a medium number of scenarios (40 scenarios) represented by Groups $B_0$ and $B$ instances (Tables 5.5, 5.7), we observe that the performance of the heuristic algorithm is almost the same for all the instances (Tables 5.6, 5.8). The performance of the tabu search algorithm differs according to the size of the instance i.e. as the number of the vertices of the network increases, the CPU consuming time of this algorithm increases too. Though there is a high improvement of the quality of the solution. However, we see that there is no rapid improvement for big size instances (Tables 5.6, 5.8).

Concerning the last group of instances; Groups $C_0$ and $C$ with a big number of scenarios (100 scenarios, Tables 5.9, 5.11), the heuristic algorithm performs rapidly and determines a solution within a very short time for all the instances, while the performance of the tabu search algorithm is affected by the size of the instance and it needs more time and memory to improve the solution found by the heuristic (Tables 5.10, 5.12). Furthermore, as the number of available vehicles decreases, the tabu algorithm performs faster for the small size instances, whereas it fails to improve for the medium and big size instances.

A general conclusion is drawn out. On one hand, the heuristic algorithm is able to determine an initial solution for any problem instance and for any number of scenarios within a very short CPU consuming time. On the other hand, the performance of the developed tabu search algorithm is related to the number of scenarios and to the size of the studied problem instance. In other terms, as the number of scenarios increases and as the size of the studied problem instance increases, the CPU consuming time of the tabu search algorithm increases too. However, this algorithm is able to improve very well the solution which is determined by the heuristic whenever it is able to improve.
Conclusion

In this chapter we studied the robust sparse capacitated arc routing problem under travel costs uncertainty where the uncertainty is represented by a set of scenarios. We study this problem over a sparse network whose maximum degree held by its vertices is 3. We presented a mathematical formulation of this problem, and we developed two algorithms to solve the problem. The first constructed heuristic algorithm shows its effectiveness by determining a feasible solution for all the studied instances within a very short CPU consuming time whatever the number of scenarios is. The performance of the developed adapted tabu search algorithm is related to two factors: (1) the size of the studied instance i.e. the number of vertices of the network over which the problem is defined and (2) the total number of scenarios. This algorithm starts by an initial solution which is determined by the heuristic algorithm and attempts to improve it. As seen in the previous section, the CPU consuming time needed by the tabu search algorithm extends as the number of the vertices of the network increases and as the number of scenarios augments. The latter algorithm did not succeed to present an improvement of the initial solution for the big size instances under a medium and a big number of scenarios, but at the end we have a robust solution determined by the heuristic even if it is not of a very good quality.
Conclusions

In this dissertation, our main investigation concerns the capacitated arc routing problem over underlying sparse graphs which is an NP-hard problem in the complexity theory. We are the first who study this problem over such class of graphs with a density reaching to 0.001. As there exist several studies of the capacitated vehicle routing problem over sparse underlying graphs in the literature, we aim at developing a transformation of the CARP into CVRP to exploit the various results of solving the latter by different researchers in order to solve the initial CARP. In addition, we build adapted and robust algorithms in order to solve the problem, and we study the robust sparse capacitated arc routing problem under travel costs uncertainty represented by a set of scenarios.

First we define our studied problem; the sparse capacitated arc routing problem which is defined to be the CARP with underlying sparse graphs. We represent the sparsity of the graph by measuring its density which is given by Remark 3.5.1. We work with graphs of very low density i.e. graphs with high sparsity by imposing two conditions. The first condition is attained by a strong relation between the number of edges and the number of vertices of the graph, and the second is given by imposing that the maximum degree held by the vertices of the graph is 3 and cannot exceed 3. Thanks to these two conditions, we represent the sparsity of the graphs which we are working with. To avoid any confusion between a sparse formulation of the CARP and a formulation of Sparse CARP, we distinguish between them as follows. A sparse formulation is a formulation whose number of variables is proportional to the number of edges of the graph in which the problem is considered, while a formulation of a Sparse CARP is a formulation that models the problem which is defined over underlying sparse graph i.e. it means that the underlying graph of our problem is sparse of a very small
density. Two mathematical formulations of the Sparse CARP are presented in which we introduce a new variable called edge capacity. This is due to the fact that our problem is special as it is concerned with exceptional cases as wars and crisis where it is not allowed to traverse an edge any number of times. For this, we modeled the problem mathematically considering two cases: (1) when it is not allowed to traverse an edge more than one time and in this case the edge capacity is equal to 1, and (2) when it is allowed to traverse the edge a constant number of times which should not be exceeded. The concept of edge capacity is not represented before by any work in the literature, however, an aggregated variable that counts the number of times of traversing an edge has been introduced before. The novelty of our variable is that it approaches more the realistic life cases and it is not considered as a decision variable of the modeling.

A new transformation technique of the CARP into CVRP is then addressed. As there is a lot of research about the vehicle routing problems and the sparse vehicle routing problem in particular, we intend to build a new transformation technique that transforms our Sparse CARP into an equivalent Sparse CVRP to invest the results which are obtained for the latter in order to solve the initial Sparse CARP. The contribution of our technique is that it conserves the structure of the graph with its special sparsity property and conserves the structure of the problem itself.

In a second step, we consider the problem obtained by the proceeded transformation and we solve it by exact methods and by three different approaches as well. Small size instances and medium size ones are solved by the exact method which give promising numerical results. For big size instances, three various approaches are developed in order to be tackled. A constructive heuristic algorithm is built first to obtain an initial feasible solution. Another randomized heuristic algorithm is then presented in order to improve the solution obtained by the constructive algorithm. Finally, an adapted tabu search algorithm is proposed. All these algorithms are developed to be used under dynamic graphs. Our contribution continues at generating a new benchmark of instances as we are the first who work with such classification of CARP problems. The obtained numerical results show the high effectiveness of our built algorithms that in many cases give either optimal solutions or highly improved ones.
The last question we answer in this dissertation is the study of the robust sparse capacitated arc routing problem under travel costs uncertainty. As real life applications may face a high degree of uncertainty which stands against the feasibility or the optimality of the solution, it is important to find a robust solution whatever the situation is. We choose the robust optimization to address the travel costs uncertainty and we represent this uncertainty by a set of scenarios. We solve the problem by a min-max approach that it by minimizing the solution under the worst scenario. A mathematical modeling of the robust problem is given and two algorithms are developed. A greedy heuristic algorithm is constructed first to determine an initial feasible solution of the problem, and then an adapted tabu search algorithm is developed. The tabu search algorithm starts with the initial solution determined by the greedy heuristic and tries to improve the quality of this solution. One more time we generate our own benchmark of instances respecting the structure of the graphs which we are working over. The computational experiments show the high effectiveness and robustness of the heuristic which is able to find a feasible initial solution whatever the size of the instance is in a very small CPU consuming time. The tabu algorithm improves the quality of the solution obtained by the heuristic significantly for the small and medium size instances, whereas it does not succeed to ameliorate that of the big size instances. However, the main objective is to determine a robust solution with best expected quality under several possible conditions and not to find a single global optimal solution.

Eventually, this is a first step to study this type of arc routing problems. We present an attempt to cover the Sparse CARP over different cases; a deterministic case and a robust one. Our work in this thesis may open the door to other to investigate more about this problem.
CHAPTER 5. ROBUST SPARSE CARP WITH UNCERTAIN TRAVEL COSTS

Perspectives

In this thesis, we have already started working with the CARP over a special criteria of sparse graphs that is subjected to some specific conditions. One may be interested in generalizing this study for any sparse graph. As there is no strict clear definition to what a sparse graph is, a general notion of such graph could help in concluding general results for the CARP defined over it.

Solving the sparse CARP can be attained by two directions: (1) either by developing a transformation from sparse CARP into a sparse CVRP as we have done in this research, or (2)solving it directly by building the adapted and the effective methods and algorithms. We follow the first direction using a technique of transformation applied over specific class of sparse graphs as our main goal is to conserve the sparsity of the original graph, thus as a future work, it would be interesting to either develop a new technique that could be applied to all genres of graphs, or to adapt our proposed technique so that it can be applied generally. A key idea for adapting this technique may be to decompose the graph over which the CARP is defined into subgraphs achieving the desired imposed conditions, and then do the necessary modifications in order to test whether the same results can be attained. Concerning the second direction, a provocative issue is to solve the Sparse CARP directly, apply the developed algorithms over the same benchmark of instances we have generated and then compare the obtained results. In what follows, we suggest a mathematical model inspired by the one done in [11], however, we bring major modifications to it.

Let $G = (V, E)$ be a connected undirected graph where $V$ is the set of vertices and $E$ is the set representing the edges or the arcs. Let $R \subseteq E$ be the set of required arcs. An ARP defined over $G$ is to serve the demands of the edges of $R$ with a minimal global cost. For each edge $e \in R$, let $dem(e)$ be the demand of the edge $e$, $r_e$ denotes the routing cost of $e$, $t_e$ denotes the task cost and $p_e$ denotes the penalty cost which is paid upon the failure of doing the task. Let $K = \{1, \ldots, k\}$ be the set of identical vehicles placed at a single depot where each has a capacity $Q$. Denote by $q^k_e$ the remaining quantity held by the vehicle $k$ when it arrives the edge $e$. Note that any required edge can be traversed more than once, but serving takes place during one and only one traversal of the edge. The main objective is to service all the required edges in the graph at a least cost with feasible routes.
Mathematical formulation

We define the set of data variables and parameters of the problem as given below:

- \( x^k_e \): a non-negative integer equal to the number of times the edge \( e \) is traversed by the vehicle \( k \).

- \( y^k_e \): a binary variable such that:
  \[
  y^k_e = \begin{cases}
  1, & \text{if the edge } e \text{ is serviced by the vehicle } k; \\
  0, & \text{otherwise}.
  \end{cases}
  \]

- \( z_e \): a binary variable such that:
  \[
  z_e = \begin{cases}
  1, & \text{if } q^k_e < \text{dem}(e) \text{ for some } k \text{ serving } e; \\
  0, & \text{otherwise}.
  \end{cases}
  \]

- \( S^k_{e,f} \): a binary variable such that:
  \[
  S^k_{e,f} = \begin{cases}
  1, & \text{if the edge } f \text{ is serviced after the edge } e \text{ by the same vehicle } k; \\
  0, & \text{otherwise}.
  \end{cases}
  \]

Let \( x^k(F) = \sum_{e \in F} x^k_e \), and \( y^k(R') = \sum_{e \in R'} y^k_e \), where \( k \in K \), \( R' \subseteq R \) and \( F \subseteq E \). Let \( \delta(\Omega), \Omega \subseteq V \) be the set of the edges having one extremity in \( \Omega \), and the another one in \( V \setminus \Omega \). On the other hand, \( E(\Omega) \) is the set of edges with two extremities in \( \Omega \). \( \delta_R(\Omega) \) and \( E_R(\Omega) \) are respectively two sets of required edges in \( \delta(\Omega) \) and \( E(\Omega) \).

Hence the model:
\[
\min Z(x,y,z) = \sum_{k \in K} \sum_{e \in E} r_e x^k_e + \sum_{k \in K} \sum_{e \in R} t_e y^k_e + \sum_{e \in R} p_e z_e \quad (5.19)
\]

subject to:
\[
\sum_{k \in K} y^k_e \leq 1 \quad \forall e \in R \quad (5.20)
\]
The objective function (5.30) aims to minimize the total cost (routing, task and penalty costs). Constraints (5.31) mean that every required edge is serviced at most once and by one and only one vehicle. Constraints (5.32) are capacity constraints i.e. the capacity of the vehicles must not be exceeded. Connectivity constraints are represented in (5.33) where it ensures that if vehicle $k$ services edge $e \in R$, then its route connects this edge to the depot. Constraints (5.34) are parity constraints specifying that each route induces an even graph. Constraints (5.35) and (5.36) are related to the cases where disruptions happen once the remaining of the capacity of the vehicle does not satisfy the demand of the edge for which there will be a penalty cost to be paid. Concerning constraints (5.37) and (5.38), they are constraints that show the succession of the term $S^k_{e,f}$ and its transitivity. Constraints (5.40) are those of decision variables.
5.5. COMPUTATIONAL EXPERIMENTS

Discussion

Contrary to the model by [11], our proposed model separates the cost into three different costs: routing, task and penalty costs. The penalty cost is applied if the services is not delivered. For instance, the constraints (5.31) are inequalities which allow to miss the corresponding task. Constraints (5.32), (5.33), and (5.34) are similar to those of [11]. However, Constraints (5.35), (5.36), (5.37) and (5.38) represent the novelty of this suggestion since they impose that the capacity as well as succession and the transitivity between the services must be respected. This formulation is not tested numerically to decide whether it is better or worse than that presented in [11] and to determine its effectiveness and its drawbacks.

Furthermore, considering the robust sparse CARP, studying the sparse CARP under demands uncertainty or even under travel costs uncertainty combined with demands uncertainty for example could be an impressive subject to work on. Moreover, choosing other resolution methods rather than the tabu search algorithm, as genetic algorithm for instance, could be worth to see if it is more effective for the robust case and if it could achieve more promising numerical results.
Publications

1. Transforming sparse arc routing problems into node routing problems with a sparse feasible graph conservation
   S. FAILI, H. DKHIL, A. SBIHI and A. YASSINE
   submitted to INFOR: Information Systems and Operational Research

2. Efficient algorithms to solve the Arc Routing Problem with feasible sparse graph
   S. FAILI, H. DKHIL, A. SBIHI and A. YASSINE
   RAIRO-OR with minor corrections

3. Transforming the sparse arc routing problem into a node routing problem with the sparse feasible graph conservation
   S. FAILI, A. SBIHI and A. YASSINE
   submitted to Annals of Operations Research

4. Robust Sparse CARP under travel costs uncertainty
   S. FAILI, A. SBIHI, A. YASSINE and I. DIARASSOUBA
   submitted to the Journal of Operational Research Society
CHAPTER 5. ROBUST SPARSE CARP WITH UNCERTAIN TRAVEL COSTS
Conclusions

Cette thèse s’inscrit dans le domaine de l’optimisation et la recherche opérationnelle. La recherche qui en découle se veut essentiellement une contribution dans le domaine des problèmes de tournées (routing problems) en particulier et dans le vaste domaine de l’optimisation combinatoire en général. Notre travail de recherche a porté sur l’étude d’un problème particulier de tournées de véhicules à savoir le sparse CARP. Nous avons commencé par rappeler l’état de l’art sur ce type de problèmes. D’une part, nous avons donné une définition de la notion de “graphes creux” que nous avons adaptée à notre problème. D’autre part, nous avons étudié le problème sparse CARP sur la classe de graphes creux avec une densité atteignant 0,001. Ce problème fait partie de la classe des problèmes NP-difficiles dans la théorie de la complexité.

Notre démarche scientifique s’est basée sur l’étude du problème de tournées de véhicules (sur nœuds) (CVRP) avec graphe creux tel qu’il est défini dans la littérature. Partant des résultats existants et de la riche littérature sur les problèmes CVRP, nous avons développé une transformation du sparse CARP en un sparse CVRP. Ceci nous a permis de tirer profit des approches développées pour la résolution du problème sparse CARP initial. En outre, nous avons construit plusieurs modèles adaptés à la fois déterministes et robustes et avons proposé des algorithmes efficaces de résolution.

Dans un premier lieu, nous avons défini le concept de “non densité” (le caractère creux) ou “sparsity” d’un graphe dont nous mesurons la densité donnée par la formule en Remarque 3.5.1. Tous les graphes sous-jacents creux sur lesquels nous avons basé notre recherche se car-
actérisent par une faible densité. Cette propriété a été établie en imposant deux conditions: (i) la première condition est atteinte par une relation forte entre le nombre de nœuds du graphe et le nombre des arcs de ce graphe, (ii) la seconde est donnée en imposant que le degré maximal des sommets du graphe est égal à 3 et ne peut pas dépasser 3. Grâce à ces deux conditions, nous avons représenté le caractère creux ou “sparse” des graphes. Pour éviter toute confusion entre une formulation sparse du problème CARP et une formulation du problème sparse CARP, nous avons rappelé la différence entre ces deux notions comme suit: une formulation sparse est une formulation dont le nombre de variables est proportionnel au nombre des arêtes du graphe dans lequel le problème est considéré. Cependant, une formulation du problème sparse CARP est une formulation qui modélise le problème qui est défini sur le graphe creux, c'est-à-dire un graphe dont la densité est très faible.

Nous avons proposé deux formulations mathématiques du sparse CARP pour lesquels nous avons introduit une nouvelle variable appelée capacité d’un arc. Cela est dû à la particularité du problème étudié où les arcs requis sont un sous ensemble de l’ensemble des arcs du graphe. Deux modélisations du problème ont été considérées: (i) lorsqu’il n’est pas permis de traverser un arc requis plus d’une fois et dans ce cas la capacité de chaque arc du graphe est égale à 1, et (ii) lorsqu’il est permis de parcourir l’arc requis un nombre de fois (≥ 1) mais borné. Une des nouveautés de notre travail est d’avoir introduit le concept de capacité d’arc, aussi, une variable agrégée qui compte le nombre de fois pour traverser un arc a déjà été introduite dans la littérature. La nouveauté apportée par cette nouvelle variable est assez réaliste.

La particularité de notre technique de transformation permet de conserver la structure du graphe avec sa propriété spéciale de la “sparsity” et conserve la structure du problème lui-même. Plusieurs algorithmes ont été développés pour résoudre le sparse CARP basés sur des métaheuristiques telles que la recherche tabou. Tous ces algorithmes ont été développés sous graphes dynamiques. Les résultats numériques obtenus ont montré l’efficacité des approches développées la fois en terme de la qualité de la solution que celui du temps d’exécution.

Finalement, il s’agit d’une première ouverture pour étudier ce type de problèmes de tournées de véhicules sur arcs. Nous avons présenté une recherche qui a étudié le sparse CARP la fois déterministe et robuste. En effet, le travail proposé de cette thèse a permis de considérer une nouvelle variante des problèmes de tournées aux applications pratiques fort potentiel.

**Perspectives**

Dans cette thèse, nous avons ouvert la voie l’étude du sparse CARP. Une des recherches futures que nous proposons de développer consiste en la généralisation de cette étude à d’autres types de graphe.

Notre recherche nous a permis de résoudre le problème sparse CARP par deux techniques différentes: (i) en développant une transformation du sparse CARP en un sparse CVRP puis en utilisant des approches propres au CVRP, (ii) en résolvant directement le problème sarse CARP en construisant des méthodes adaptées. Toute transformation appliquée doit avoir pour objectif principal de conserver la “sparsity” du graphe original. Ceci constitue une nouvelle orientation des
travaux futures que nous comptons développer afin de généraliser nos approches à d’autres type de graphes (degrés quelconques, …). Une idée clé pour l’adaptation de cette technique peut être de décomposer le graphe sur lequel le CARP est défini en sous-graphes sous certaines conditions qui restent à définir. Concernant la deuxième direction, résoudre directement le sparse CARP est tout aussi un défi en introduisant des relaxations sur certaines contraintes par exemples.

Un autre travail en perspective consiste à considérer le sparse CARP robuste sous différents types de scénarios (incertitudes sur les demandes, ou en combinant les incertitudes sur les coûts de transport avec les incertitudes sur les demandes).

Dans ce qui suit, nous suggérons un modèle mathématique inspiré par le modèle de [11].

Soit $G = (V, E)$ un graphe connecté non-orienté où $V$ est l’ensemble des nœuds et $E$ est l’ensemble des arêtes. Soit $R \subseteq E$ l’ensemble des arêtes requises. Pour chaque arête $e \in R$, soit $dem(e)$ la demande de $e$, $r_e$ le coût de transport par $e$, $t_e$ le coût de service de $e$ et $p_e$ le coût de pénalité payé quand n’est pas fait. Soit $K = \{1, \ldots, k\}$ l’ensemble des véhicules homogènes au dépôt. La capacité de chaque véhicule est $Q$. La paramètre $q_k^e$ donne la capacité restant dans le véhicule $k$ quand elle arrive $e$. Chaque arête non-requis peut être traversée plus qu’une fois, mais le service est fait par un seul traversal de l’arête. L’objectif principal est de servir tous les arêtes requis dans le graphe à coût minimale avec des routes ralisables.

**Formulation Mathématiques**

Variables et paramètres:

- $x_k^e$: un entier non négatif égal au nombre de fois où $e$ est traversé par $k$.
- $y_k^e$: une variable binaire telle que:

$$
y_k^e = \begin{cases} 
1, & \text{si } e \text{ est servi par } k; \\
0, & \text{sinon.}
\end{cases}
$$
• $z_e$: une variable binaire telle que:

$$z_e = \begin{cases} 1, & \text{si } q^k_e < \text{dem}(e) \text{ pour un } k \text{ qui serve } e; \\ 0, & \text{sinon.} \end{cases}$$

• $S^k_{e,f}$: une variable binaire telle que:

$$S^k_{e,f} = \begin{cases} 1, & \text{si } f \text{ est servi apr\`es } e \text{ par le m\^eme } k; \\ 0, & \text{sinon.} \end{cases}$$

Soit $x^k(F) = \sum_{e \in F} x^k_e$, and $y^k(R') = \sum_{e \in R'} y^k_e$, $k \in K$, o\`u $R' \subseteq R$ et $F \subseteq E$.

Soit $\delta(\Omega), \Omega \subseteq V$ l’ensemble des ar\`etes avec une extrémit\`e dans $\Omega$, et l’autre est dans $V \setminus \Omega$. Par contre, $E(\Omega)$ est l’ensemble des ar\`etes avec deux extrémit\`es dans $\Omega$. $\delta_R(\Omega)$ et $E_R(\Omega)$ sont deux ensembles des ar\`etes requis dans $\delta(\Omega)$ et $E(\Omega)$.

Alors le mod\`ele est:

$$\min Z(x,y,z) = \sum_{k \in K} \sum_{e \in E} r^k_e x^k_e + \sum_{k \in K} \sum_{e \in R} t^k_e y^k_e + \sum_{e \in R} p^k_e z^k_e$$ (5.30)

sous contraintes:

$$\sum_{k \in K} y^k_e \leq 1 \quad \forall e \in R$$ (5.31)

$$\sum_{e \in R} \text{dem}(e) y^k_e \leq Q \quad \forall k \in K$$ (5.32)

$$x^k(\delta(\Omega)) + y^k(\delta_R(\Omega)) \geq 2y^k_f \quad \forall \Omega \subseteq V \setminus \{\text{dep\'ot}\} e \in E_R(\Omega), k \in K$$ (5.33)

$$x^k(\delta(\Omega)) + y^k(\delta_R(\Omega)) = \text{nombre pair, } \Omega \subseteq V \setminus \{\text{dep\'ot}\}, k \in K$$ (5.34)

$$z^k_e = 1 - \sum_{k \in K} y^k_e \quad \forall e \in R$$ (5.35)
5.5. COMPUTATIONAL EXPERIMENTS

\[ q^k_e = Q - \sum_{f \in R} S^k_{f,e} \text{dem}(f) \geq \text{dem}(e) \quad \forall e \in R, e \neq f, f \in R, k \in k \] (5.36)

\[ \min(y^k_e, y^k_f) = S^k_{e,f} + S^k_{f,e} \quad \forall e, f \in R, e \neq f, k \in k \] (5.37)

\[ \min(S^k_{e,f}, S^k_{f,g}) \leq S^k_{e,g} \quad \forall e, f, g \in R, e \neq f \neq g \] (5.38)

\[ x^k_e \in \mathbb{Z}_+ \forall e \in E, k \in K \] (5.39)

\[ y^k_e, z_e, S^k_{e,f} \in \{0, 1\} \quad \forall e, f \in R, k \in K \] (5.40)

L’objectif (5.30) minimise le coût total. Les contraintes (5.31) signifient que chaque arête requise est servie une seule fois par un seul véhicule. Les contraintes de connectivité sont représentées dans (5.32). Les contraintes (5.33) sont des contraintes de parité qui montrent que chaque route induit un graphe pair. Les contraintes (5.35) et (5.36) sont liés aux cas où des perturbations se produisent une fois que le reste de la capacité du véhicule ne satisfait pas à la demande d’arête et pour lequel il y aura un coût de pénalité à payer. Les contraintes (5.37) et (5.38) montrent la successivité des \( S^k_{e,f} \) et leur transitivité. Les contraintes (5.40) représentent les variables de décision.

Discussion

Contrairement au modèle proposé dans [11], notre modèle sépare la fonction de coût en trois coûts différents: les coûts de routage, de tâche et de pénalité. Le coût de la pénalité est appliqué si les services ne sont pas livrés. Par exemple, les contraintes (5.31) sont des inégalités qui permettent de manquer la tâche correspondante. Les contraintes (5.32), (5.33) et (5.34) sont similaires à celles de [11]. Cependant, les contraintes (5.35), (5.36), (5.37) et (5.38) représentent la nouveauté de
ce modèle puisqu’elles imposent que la capacité ainsi que la successivité et la transitivité entre les services doivent être respectée.

Cette formulation n’est pas encore validée ni testée numériquement pour décider si elle est meilleure ou pire que celle présentée dans [11] et pour déterminer son avantage et son désavantage.
Bibliography


experience, in The vehicle routing problem. Paolo Toth and Daniele Vigo.


