Exploring Combinatorial Aspects of the Stop Number Problem

Rafael Colares

THÈSE

Présentée par

Rafael COLARES

pour obtenir le diplôme de

Docteur d'université

Spécialité : Informatique

Exploring Combinatorial Aspects of the Stop Number Problem

Soutenue publiquement le 28 octobre 2019 devant le jury

M. ou Mme	Walid	BEN-AMEUR	Rapporteur et examinateur
	Ali Ridha	MAHJOUB	Rapporteur et examinateur
	Aurélie	LAGOUTTE	Examinatrice
	Gautier	STAUFFER	Examinateur
	Hande	YAMAN	Examinatrice
	Mourad	BAÏOU	Directeur de Thèse
	Hervé	KERIVIN	Directeur de Thèse

École Doctorale Sciences Pour l'Ingénieur
Université Clermont Auvergne

Acknowledgements

PhD students are traditionally undergraduate students who used to excel in some specific field they learned to appreciate during their education. However, many of these students never really needed to face the unknown before starting a PhD thesis. Whenever a question was proposed, the answer – as hard to understand as it could be – could be found in a book and/or explained by the teacher. Carrying on a research project is a whole different story. During the development of significant research, one is continuously faced with questions for which the answers are unknown and trying to solve such questions usually involve failing multiple times. Dealing with constant failure together with the pressure for meaningful discoveries can be sometimes overwhelming, which makes the conception of a PhD thesis a pretty rough task. I dedicate this section to all the ones that directly or indirectly contributed to the achievement of this thesis.

First and foremost I would like to express my admiration for all the professors that are able to manage their efforts between the development of scientific research and the challenging duties of teaching and inspiring undergraduate students. In this regard I would like to highlight professors Carlos Vênancio and Philippe Mahey. Without them the BRAFITEC exchange program between the Industrial Engineering Department from UFMG (Univerdade Federal de Minas Gerais) and the computer science engineering school ISIMA (Institut Supérieur d'Informatique, de Modélisation et de leurs Applications) would never have come to be the success it is today and I would never had the chance of pursuing my studies in France. Furthermore, I would like to stress my gratitude towards professor Martín Ravetti who first presented me to the operations research and optimization fields.

Secondly, I would like to thank my PhD advisors Mourad Baïou and Hervé Kerivin. I am truly honored and grateful for being accepted as the PhD student of such great researchers. I am glad to say today that these two form a fairly complementary team and that I was fortunate enough to have them by my side during these years. On the one hand, I was fascinated by the ability of Mourad to think outside the box and put together concepts that seemed to be unrelated

to me. In the most natural way, Mourad was capable of showing me how each scientific research field is nothing but a frame of a much bigger picture. Among many other things, he was also able of teaching me how failure should be accepted as an important part of the road to success and hence how relevant research should be faced. On the other hand, the countless and fruitful discussions with Hervé have helped me improving many skills, including scientific writing, methodology of research and various technical aspects of combinatorial optimization. Furthermore, his step-by-step guidance was illuminating when tackling the numerous complex problems faced throughout this thesis. All in all, I am proud of the relationship we have developed over these years.

Next I would like to thank my committee members Professors Walid Ben-Ameur, Aurélie Lagoutte, Ali Ridha Mahjoub, Gautier Stauffer and Hande Yaman for the time and effort invested in the reviewing phase of this manuscript. Their pertinent questions and remarks definitely contributed to the improvement of the thesis quality. Special thanks are due to Professor José Correa and all his research team not only for hosting me at Universidad de Chile during an enjoyable one-month visit but also for introducing me to the charming field of approximation algorithms and sharing their thoughtful insights with me. I also would like to thank the Laboratory of Excellence IMobS3 (Innovative Mobility: Smart and Sustainable Solutions) for founding my PhD, and LIMOS (Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes) for hosting me and providing all the physical conditions for the research development.

The past three and a half years at LIMOS have been an incredible experience, not only professionally but also personally. I thank my peer colleagues Benjamin Bergougnoux, Benjamin Dalmas, Matthieu Gondran, Rui Shibasaki – and many others – for their friendship and the memorable moments spent together (*e.g.*, the PhD "Olympic" Games, Matthieu's bachelor party and marriage, the countless beers and board game parties shared). I also would like to thank the many friends I have made outside the LIMOS who have contributed to make this time joyful and fun.

I thank all my family for the endless support and love. I also thank all my childhood friends back in Brazil for their lifetime sincere friendship and trust. Special thanks are due to my BRAFITEC friends Ana Alice Barros, Guilherme Martino, Rui Shibasaki and Fábio Silva – among others – for the good time we have shared during my first years in France.

Last but definitely not least, I would like to thank my loving and supportive partner Pauline Huau for all the moments shared along these years. For all the dedication, patience, love and tenderness, thank you.

All truths are easy to understand
once they are discovered;
the point is to discover them.

Galileo Galilei

Abstract

The Stop Number Problem arises in the management of a dial-a-ride system with small autonomous electric vehicles. In such a system, a fleet of identical capacitated vehicles travels along a predefined circuit with fixed stations in order to serve clients requesting for a ride from an origin station to a destination station. Notice that multiple clients may share the same origin and/or destination stations. The Stop Number Problem consists of assigning each client request to a vehicle such that no vehicle gets overloaded. The goal is to minimize the number of times the fleet of vehicles stops for picking up or delivering clients. When every client requests for exactly one seat in a vehicle, Stop Number Problem is called Unit Stop Number Problem. In this thesis, Unit Stop Number Problem is addressed as a combinatorial-optimization problem.

First, we investigate the complexity of such problem. On the one hand, we study some properties of optimal solutions and derive a series of particular cases that are shown to be solvable in polynomial time. On the other hand, we show that Unit Stop Number Problem is $\mathcal{NP}$-Hard even when restricted to case where each vehicle can take at most two clients at once and the graph induced by the client requests is planar bipartite. Such result – which positively answers a conjecture of Pimenta et al. [151] – is then extended to other related problems such as the k-Edge Partitioning and the Traffic Grooming problem, improving their respective state-of-the-art complexity standards.

In a second part, we consider an integer-programming formulation known in the literature for solving the Unit Stop Number Problem. A preliminary analysis is conducted in order to prove the weakness of such formulation. Afterwards, such formulation is reinforced through a polyhedral approach. We provide a facial study of the polytope associated with the solutions of this problem. New valid inequalities are introduced and necessary and sufficient conditions for which they are facet-defining are given.

Finally, based on the discussed polyhedral results, we derive a new efficient branch-and-cut algorithm. Performance boosting features such as symmetry break-

ing methods and variable elimination/relaxation are also investigated and implemented into the developed framework. Results convincingly demonstrate the strength of the reinforcing valid inequalities and therefore of our branch-and-cut algorithm.

Key words: dial-a-ride, stop number problem, combinatorial optimization, complexity, NP-Hardness, polyhedral combinatorics, branch-and-cut algorithm.

Contents

List of Figures

List of Tables

Introduction

> "Begin at the beginning, the King said gravely, and go on till you come to the end: then stop."
>
> — Lewis Carroll, Alice in Wonderland

Combinatorial optimization is a significant topic in computer science combining applied mathematics, algorithms and computation theory. A combinatorial optimization problem is typically stated as choosing the best element among a finite set of possibilities and generally arises – but not exclusively – from practical situations. Main applications areas include logistics and supply chains, manufacturing, transport, team and portfolio management, telecommunications, finance and many others.

More formally, given a finite set of elements $N = \{1, \ldots, n\}$, a family $\mathcal{F}$ of subsets $F \subseteq N$ and a cost function $c : N \to \mathbb{R}$, a combinatorial optimization problem consists on determining an element $F \in \mathcal{F}$ such that $c(F)$ is maximum (or minimum) over $\mathcal{F}$. Such problems may appear easy to solve at first sight. For instance, an straightforward approach would be trying to enumerate all elements in $\mathcal{F}$ – recall that $\mathcal{F}$ is finite! – and simply choosing the best one according to the cost function c. Nevertheless, more often than not such family is composed of a very large (*e.g.*, exponential in the size of N) number of elements, which prevents the problem's resolution through such approach within a life-time period using a limited resourced computer.

This observation has led scientists to investigate and develop cleverer and fancier methods to solve this kind of problems. Linear and integer programming as well as algorithms based on dynamic programming and/or min-max relationships between combinatorial structures are examples of such methods. However, even with all the knowledge developed throughout the history of computer science, some combinatorial problems are still hard to solve. This means that no algorithm with running time bounded by a polynomial function in the size of its input is known.

Among these hard problems, a certain class of problems stands out: the so-called

$\mathcal{NP}$-Hard problems. Informally, a problem belongs to the $\mathcal{NP}$-Hard class if it is at least as hard as any other problem in the $\mathcal{NP}$ class, and a problem is in $\mathcal{NP}$ if it can be formulated as a yes-no question such that the instances where the answer is "yes" have efficiently verifiable proofs. With respect to combinatorial problems, this amounts to say that a combinatorial problem is in $\mathcal{NP}$ if it can be formulated through the question of whether or not there exists an element $F \in \mathcal{F}$ such that $c(F)$ is at least (or at most) a given value $B \in \mathbb{R}$, and $c(F)$ can be computed in polynomial time.

For $\mathcal{NP}$-Hard problems, it is known that if an efficient algorithm exists for a single $\mathcal{NP}$-Hard problem, then all $\mathcal{NP}$-Hard problems can also be solved in polynomial time. Such property guided researchers to impose a dichotomy between efficiently verifiable problems (class $\mathcal{NP}$) and efficiently solvable problems (class $\mathcal{P}$). Whether or not these two classes are equivalent has never been proved and remains one of the golden open questions in computer science. However, it is widely accepted that $\mathcal{P} \neq \mathcal{NP}$.

The needs for efficient algorithms to solve real-world problems has never ceased to increase since the construction of the first computers in the 40's. In this scenario, whenever a new combinatorial problem is proposed, one of the first questions one asks himself is whether such problem is in $\mathcal{P}$ or in $\mathcal{NP}$-Hard. If one is capable of proving that the problem belongs to $\mathcal{NP}$-Hard, then an efficient algorithm for solving this problem is unlikely (to say the least) to be conceivable. On the other hand, $\mathcal{NP}$-Hard combinatorial problems have plenty of applications in practice, and therefore it is unwise and counter-productive to just ignore them. Instead, for such problems one must aim at solving it not through a polynomial time algorithm but with a suitable algorithm, that is, as efficiently as possible.

The polyhedral approach, developed by Jack Edmonds in the 60's (see Edmonds [59]) over the maximum matching problem, combines the ideas behind linear and integer programming together with the concepts of discrete geometry in order to solve combinatorial problems. It consists on formulating a combinatorial problem as a linear program describing the polyhedron corresponding to the convex hull of its solutions.

Nonetheless, a complete description of such polyhedron might be tricky to obtain and often enough an exponential number of inequalities is needed. Luckily, when solving a combinatorial problem, a partial description of the convex hull of its solutions might be sufficient for solving it in polynomial time. A typical strategy is hence to develop a Branch-and-Cut framework capable of adding inequalities from such description on demand. In this sense, the work of Grötschel et al. [82] plays an

important role by stating that a combinatorial problem can be solved in polynomial time if and only if there exists a polynomial time algorithm capable of solving the associated separation problem, that is, given a point x, decide whether or not x belongs to the polyhedron and, if not, provide an inequality that separates x from the polyhedron. Accordingly, for $\mathcal{NP}$-Hard combinatorial problems, one should expect the associated separation problem to be also $\mathcal{NP}$-Hard. In such cases, one can still benefit from polyhedral approaches by trying to solve the associated separation problem heuristically. Such approach can hence be exploited to yield valid inequalities capable of enhancing the linear relaxation of a given formulation.

Polyhedral approaches have been given great attention in the past fifty years, from both theoretical and practical perspectives, and have proved to be powerful for optimally solving hard combinatorial problems. Indeed, many important state-of-the-art algorithms for solving such problems take advantage of the polyhedral approach. Among these, one of the most famous is the Concorde TSP Solver, a Branch-and-Cut based algorithm treating the classic Travelling Salesman Problem (see Applegate et al. [4, 6]).

In this thesis, we face a new combinatorial problem called the Stop Number Minimization Problem arising from the management of a fleet of autonomous vehicles and first introduced in Pimenta et al. [151]. In such problem, a fleet of identical capacitated vehicles travels through a predefined circuit with fixed stations, while customers demand for a ride between stations of their choice. Assigning a customer demand to a vehicle imposes such vehicle to stop for picking-up the customer at its departure station, carry it until its destination, and stop again for dropping it off at its destination station. The problem consists then in assigning each customer demand to a vehicle such that its capacity is respected all along the circuit and the total number of pick-up and drop-off operations is minimized.

In Chapter 1, an important and introductory background on the concepts exploited throughout this dissertation is provided. In Chapter 2, the context on which Stop Number problem arises is described in detail and the state-of-the-art concerning such problem is presented. Chapter 3 focus on the investigation of the problem's complexity, answering an open conjecture due to Pimenta et al. [151] and giving rise to new results that can be extended to other related problems. Once the problem is shown to belong to the class of $\mathcal{NP}$-Hard problems, a polyhedral study is conducted in Chapter 4, yielding new facet-defining valid inequalities. In Chapter 5, the practical point of view is investigated. The symmetry hidden behind solutions of the Stop Number Minimization Problem is investigated and showed to slow down standard Mixed Integer Linear Programming performances. In order to counter such property, symmetry-breaking methods are developed and implemented for boosting

the computational results. Finally, a Branch-and-Cut framework is described and implemented where the valid inequalities previously presented are introduced to the formulation on demand. The implementation is made under C++ language using the standard Mixed Integer Linear Program solver `IBM ILOG CPLEX 12.8` coupled with `CPLEX Concert Technology`. For the implementation of common data structures such as graphs the `COIN-OR LEMON` C++ library is exploited.

Chapter 1

Preliminaries and notation

> "It is worth noting that the notation facilitates discovery. This, in a most wonderful way, reduces the mind's labour."
>
> — Gottfried W Leibniz

This chapter introduces some preliminary definitions, notation and background knowledge that will ease the reader's effort throughout this dissertation.

1.1 Graph Theory

The terminology employed is standard and follows the one applied in Diestel [54].

Graphs, vertices and edges. A *simple graph* is an ordered pair $G = (V, E)$, where V is a finite set of objects whose elements are called *vertices*, and E is a finite set of pairs of vertices in V called *edges*. Given a graph G, its vertex set is denoted by $V(G)$, and its edge set is denoted by $E(G)$. If set E is composed of unordered pairs of vertices, the graph is referred to as *undirected*. In contrast, if E is composed of ordered pairs of vertices, the graph is referred to as *directed*.

Endpoints, incidence and degree. For simplicity, an edge $\{u, v\} \in E$ can also be denoted by uv. Moreover, sets composed by a single element x can be simply represented by x instead of $\{x\}$. Given an edge $e = uv$, vertices u and v are called the *endpoints* of e. For a given graph $G = (V, E)$, we say that an edge $e \in E$ is *incident* to $v \in V$ if v is one of the endpoints of e. The set of edges incident to $v \in V$ is denoted by $\delta_G(v)$, or simply $\delta(v)$ if graph G is clear from context. The *degree* of a vertex $v \in V$ (*i.e.,* the number of edges incident to $v \in V$), is denoted by $\deg_G(v)$, or simply $\deg(v)$ if graph G is clear from context.

Parallel edges and multigraphs. Two or more edges with the same endpoints are called *multiple* (or *parallel*) edges. In this case, E is a multiset. If $G = (V, E)$ admits multiple edges, the graph is referred to as a *multigraph*. An edge $e \in E$ such that $e = uu$ with $u \in V$ is called a *loop*.

Subgraphs and inductions. Given a graph $G = (V, E)$, the graph G' is said to be a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. For a subset of vertices $S \subseteq V(G)$, the set of edges in E having both endpoints in S, that is $\{uv \in E : u, v \in S\}$, is denoted by $E[S]$. We say that $E[S]$ is the set of edges *induced* (or *spanned*) by S. Moreover, we denote $G[S] = (S, E[S])$ the subgraph of G induced by the vertex set S. Analogously, for a subset of edges $S \subseteq E(G)$, the set of vertices *induced* (or *spanned*) by S is denoted by $V[S]$, and the subgraph of G induced by the edge set S is denoted by $G[S] = (V[S], S)$. A vertex-induced subgraph $G[S]$ is *maximal* (*minimal*) with respect to a given property if $G[S]$ satisfies such property but $G[S \cup v]$ ($G[S \setminus v]$) does not, for any given $v \in V(G) \setminus S$.

Paths and cycles. A *path* is a non-empty graph P where $V(P) = \{v_0, v_1, \ldots, v_n\}$and $E(P) = \{v_0v_1,\ v_1v_2,\ \ldots,\ v_{n-1}v_n\}$ such that vertices v_i are all distinct for $i = 0, \ldots, n$. Vertices v_0 and v_n are called the *ends* of path P. A path P with ends $s \in V(P)$ and $t \in V(P)$ is called an *st-path*. The *length* of a path is given by the number of its edges. Given a path P, the graph $C = (V(P), E(P) \cup v_nv_0)$ is called a *cycle*.

Partitions and connectivity. An *edge-partition* of a graph G is a set $\Pi = \{S_1, \ldots, S_k\}$ of subsets of $E(G)$ where subsets in Π are pairwise disjoint and their union gives $E(G)$. In other words, $\Pi = \{S_1, \ldots, S_k\}$ is an edge-partition of G if $\bigcup_{i=1}^{k} S_i = E(G)$ and $S_i \cap S_j = \emptyset$ for $i \neq j$. Analogously, a *vertex-partition* of a graph G is a set $\Pi = \{S_1, \ldots, S_k\}$ of subsets of $V(G)$ where subsets in Π are pairwise disjoint and their union gives $V(G)$. That is to say, $\Pi = \{S_1, \ldots, S_k\}$ is a vertex-partition of G if $\bigcup_{i=1}^{k} S_i = V(G)$ and $S_i \cap S_j = \emptyset$ for $i \neq j$. A graph G is said to be *connected* if there exists an *st*-path between any two vertices $s \in V(G)$ and $t \in V(G)$. A *connected component* of G is a maximal connected vertex-induced subgraph of G. A graph can be partitioned into its connected components (see Figure 1.1).

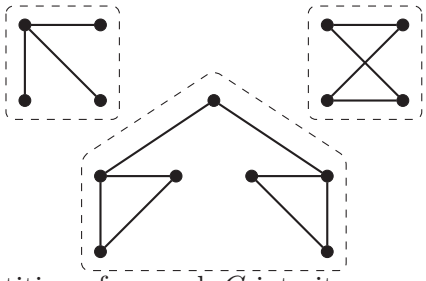
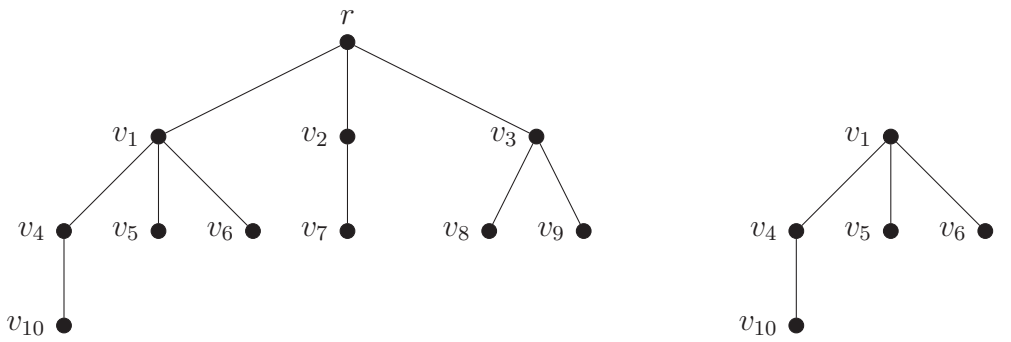


Figure 1.1: Partition of a graph G into its connected components

Trees and forests. If a graph $T = (V, E)$ is acyclic, then it is called a *forest*. If T is also connected, then it is called a *tree*. A vertex $v \in V$ is called a *leaf* if $\deg_T(v) = 1$. Analogously, an edge $uv \in E$ is called a leaf-edge if either u or v is a leaf. In contrast, a vertex $v \in V$ is called an *internal vertex* if $\deg_T(v) \geq 2$. A *rooted tree* is a tree with a distinguished vertex r called the *root*. For a rooted tree T, we say that a node $u \prec v$ if u lies in the unique path from r to v. For a given edge $uv \in E(T)$, if $u \prec v$, we say that u is the parent of v and v is a child of u. The set of the children of v on a rooted tree T is denoted by $\mathrm{ch}_T(v)$ (or simply $\mathrm{ch}(v)$,if the tree in question is implied by the context). If v is a leaf (and v is not the root), then $\mathrm{ch}_T(v) = \emptyset$. Given a rooted tree T, the rooted subtree of T that is rooted in a node $v \in V(T)$ is denoted by T_v. Notice that node $v' \in V(T)$ belongs to T_v if and only if v lies in the unique path from r to v (see Figure 1.2). By definition, $T_r = T$.



(a) A tree T **(b)** Rooted subtree T_{v_1}

Figure 1.2: Rooted Trees

Complements and cuts. Given a subset of vertices $S \subseteq V$ within a graph $G = (V, E)$, the *complement* of S, denoted by $\overline{S}$, is the set of vertices in $V \setminus S$. The *cut* induced by S, denoted by $\delta_G(S)$, is the set of edges having exactly one endpoint in S and one in $\overline{S}$. Precisely, $\delta_G(S) = \{uv \in E(G) : u \in S, v \in \overline{S}\}$.

Complete, bipartite and complete bipartite graphs. A graph $G = (V, E)$ is said to be *complete* if there exists an edge between every pair of vertices (*i.e.,* $uv \in E$ for any $u \in V$ and $v \in V$). The complete graph with n vertices is denoted by K_n. A graph $G = (V, E)$ is said to be *bipartite* if there exists some subset $S \subseteq V$, for which $\delta_G(S) = E$. Equivalently, a graph G is bipartite if and only if it does not contain an odd cycle (*i.e.*, a cycle with an odd number of edges). A graph $G = (V, E)$ is said to be *complete bipartite* if there exists some subset $S \subseteq V$, for which $\delta_G(S) = E$ and $uv \in E$ for any $u \in S$ and $v \in \overline{S}$. The complete bipartite graph with $|S| = n$ and $|\overline{S}| = m$ is denoted by $K_{n,m}$. The complete bipartite graph $K_{1,n}$ is called a *star*. A star can also be seen as a tree with n leaves and one internal vertex. A star with exactly 3 edges is called a *claw*.

Deletions, contractions and subdivisions. Given a graph $G = (V, E)$, the graph obtained by *deleting* some edge $e \in E$ is $G' = (V, E \setminus e)$. On the other hand, the graph obtained by *deleting* some vertex $v \in V$ is $G' = (V \setminus v, E \setminus \delta_G(v))$. The *contraction* of an edge $uv \in E$ is done by deleting edge uv and merging vertices u and v into a new vertex v' (see Figure 1.3). More formally, the graph obtained by contracting an edge $e \in E$ is $G' = (V', E')$, with the vertex set

$$V' := V \setminus \{u, v\} \cup v',$$

where v' is the new "merged" vertex (*i.e.*, $v' \notin V$), and the edge set

$$E' := \{E \setminus \{\delta_G(u) \cup \delta_G(v)\}\} \cup \{wv' : uw \in E \setminus uv \text{ or } vw \in E \setminus uv\}.$$

The *subdivision* of an edge $uv \in E$ corresponds to the replacement of uv by two edges uw and wv along with a new vertex w (see Figure 1.3). The graph obtained from the subdivision of edge $uv \in E$ is therefore

$$G' = (V \cup w, E \setminus uv \cup \{uw, wv\}).$$

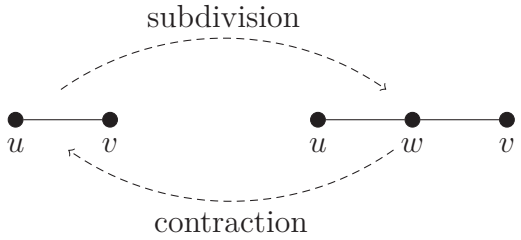


Figure 1.3: Contraction of node w and subdivision of edge uv

Minors and planar graphs. A graph H is called a *minor* of graph G if H can be obtained from G by deleting edges and vertices and by contracting edges. A graph $G = (V, E)$ is *planar* if it can be embedded in the plane without crossing edges. Equivalently, a graph is planar if and only if it does not contain either K_5 or $K_{3,3}$ as a minor.

Throughout this dissertation, all assessed graphs are considered to be loopless. Moreover, unless explicitly mentioned, all graphs are taken to be undirected multigraphs.

1.2 Algorithms and Complexity

Before addressing algorithms and computational complexity theory, the notion of what a *problem* is – at least from an algorithmic point of view – should be clear to any reader. According to Garey and Johnson [74, p. 4], a problem can be defined as a general question, normally containing a number of *parameters* (also called *input*). In this sense, a problem is fully described by providing a generic description of all its parameters and a specification of which properties the *answer* (also called *output*) should possess. An *instance* of a given problem is a set of well-specified values for the problem's input.

Example 1.1 gives the description of a well-studied problem called Facility Location Problem (FLP), on which a group of customers must be served by not yet built facilities. A decision maker must then choose among a set of possibilities where to install (open) such facilities so that the total cost of serving customers and installing facilities is the minimum possible.

Example 1.1 - Facility Location Problem
Input: A set $C = \{c_1, \ldots, c_n\}$ of customers, a set $F = \{f_1, \ldots, f_m\}$ of candidate facility sites, a cost $c_j \in \mathbb{R}^+$ for opening facility $j \in F$ and a cost $d_{ij} \in \mathbb{R}^+$ for serving client $i \in C$ from facility $j \in F$.

Output: A subset $F' \subseteq F$ of opened facilities that minimizes the sum of service costs from each customer to its nearest facility, plus the sum of opening costs of facilities in F'. □

A *decision problem* is a problem whose output is either *"yes"* or *"no"*. An instance whose output is "yes" is called a *yes-instance* or a *no-instance*, otherwise. An *optimization problem* is a problem that aims at maximizing or minimizing a given objective. The problem described in Example 1.1 is classified as an optimization problem. Notice however, that every optimization problem can be translated into a decision problem, giving rise to the so-called *associated decision problem*. Example 1.2 describes the decision problem associated to FLP.

Example 1.2 - Decision Facility Location Problem
Input: A set $C = \{c_1, \ldots, c_n\}$ of customers, a set $F = \{f_1, \ldots, f_m\}$ of candidate facility sites, a cost $c_j \in \mathbb{R}^+$ for opening facility $j \in F$, a cost $d_{ij} \in \mathbb{R}^+$ for serving client $i \in C$ from facility $j \in F$, and a budget $B \in \mathbb{R}^+$.

Output: Is there a subset $F' \subseteq F$ of opened facilities such that the sum of service costs from each customer to its nearest facility, plus the sum of opening costs of facilities in F' is at most B? □

Algorithms are step-by-step procedures conceived for solving problems. An algorithm is said to *solve* a problem $\mathcal{P}$, if for any given instance $\mathcal{I}$ of $\mathcal{P}$, it produces a correct answer for that instance $\mathcal{I}$. *Computer programs* are the concrete product of the implementation of an algorithm.

Like in real life, a problem may be approached from various angles and some might perform more or less efficiently than others. This means that for a single problem $\mathcal{P}$, there exists a great variety of algorithms solving $\mathcal{P}$, each being more or less efficient. Take for example, the problem of finding the greatest common divisor of two given integers a and b. A naive approach would be writing down a list of all divisors for both integers a and b, and then choosing the greatest one appearing in both lists. However, a much more efficient algorithm is known for solving such problem. Indeed, the greatest common divisor can be found in just a few steps through the well-known Euclid's Algorithm (see Stark [169, p. 16]).

By efficiency of an algorithm we refer to the amount of computational resources – such as time or memory – a computer needs to execute it. In fact, time requirements are often enough a dominant factor. Whether or not an algorithm is useful in practice is considerably determined by the computational time it requires. It is funny to imagine that if the algorithm used to find a driver's best route on a GPS took 20 minutes to be executed instead of some fraction of seconds, people would still be using physical paper maps.

It should be clear that the computational time needed to run an algorithm depends on the instance's size. It is entirely rational to imagine that sorting a vector with 1,000 entries should take longer than one with 10 entries. In this regard, a standard way of measuring the time complexity of an algorithm is by providing an upper bound on the running time of its worst-case scenario, that is, the longest running time for any input of size n.

The *big O notation* is the most common metric for analyzing the worst-case running time of an algorithm. By relating the time complexity of an algorithm to the number of steps required to complete it, the big O notation describes the asymptotic behaviour of the algorithm's running time as its input size grows.

Definition 1.1 - Big O notation
Given two function $f, g : \mathbb{R} \to \mathbb{R}$, $g(n)$ is said to belong to $\mathcal{O}(f(n))$, that is, $g(n) \in \mathcal{O}(f(n))$ if there exists positive constants $c, n_0 \in \mathbb{R}^+$ such that $g(n) \leq c\ f(n)$ for all $n \geq n_0$.

A *polynomial time algorithm* is an algorithm whose running time is bounded above by a function in $\mathcal{O}(p(n))$ where p is a polynomial function and n is the input size. Conversely, an algorithm whose running time cannot be bounded above by a

function in $\mathcal{O}(p(n))$ is called an *superpolynomial algorithm*. An algorithm is called *exponential time algorithm* if its running time is bounded above by a function in $\mathcal{O}(2^{n^k})$, for some constant k.

We have seen above that different algorithms may have different time complexity, but what about problems? Once again, just like in real life, there exists problems that seems to be harder to solve than others. The complexity of a problem is taken to be the complexity of the best known algorithm solving such problem. It should be intuitive for any reader that finding all the divisors of an integer seems to be a much more complicated task than finding the greatest common divisor of two given integers. Such intuition is not entirely wrong. Indeed, if one knows how to find all the divisors of an integer, the problem of finding the greatest common divisor of two given integers becomes trivial. For this reason, one may remark that the first problem is at least as hard as the latter one. However, even if one is able to come up with an efficient (say polynomial time) algorithm for the latter problem, this remark does not give any clue on how to solve the first problem.

Indeed, showing that no polynomial time algorithm exists for some given problem might be just as hard as designing a polynomial time algorithm that does solve the problem. As a matter of fact, there exists many interesting problems for which no one was able to find an efficient algorithm or to rule out the existence of such an algorithm. In contrast, what has been largely done is to classify the hardness of a problem by comparing it to another, just as we did above for the greatest common divisor problem.

The theory of $\mathcal{NP}$-Completeness introduced independently by Cook [36] and Levin [119] in the early seventies, remains up to now the most famous and applied classification system of a problem's complexity. A problem belongs to $\mathcal{P}$ (*Polynomial*) if it can be solved through a polynomial time algorithm. Examples of problems in $\mathcal{P}$ are the Minimum Spanning Tree problem (Kruskal [111], Prim [152]) and the Matching problem (Edmonds [60]). Recently, the problem of determining whether or not a given integer is prime was shown to be in $\mathcal{P}$ (Agrawal et al. [2]).

A problem belongs to $\mathcal{NP}$ (*Nondeterministic Polynomial*) if a solution of this problem can be checked to meet the required properties in polynomial time. Notice, that obviously $\mathcal{P} \subseteq \mathcal{NP}$. Informally, the set of $\mathcal{NP}$-Hard problems refers to problems that are at least *as hard as* the hardest problems in $\mathcal{NP}$.

A decision problem $\mathcal{A}$ is said to be *polynomially reducible* to another decision problem $\mathcal{B}$, denoted by $\mathcal{A} \propto \mathcal{B}$, if there exists a polynomial time algorithm that transforms

i. any "yes"-instance of $\mathcal{A}$ into a "yes"-instance of $\mathcal{B}$, and

ii. any "no"-instance of $\mathcal{A}$ into a "no"-instance of $\mathcal{B}$.

Proving that a problem is at least as hard as a problem can be done through *polynomial reductions*. In fact, given two problems $\mathcal{A}$ and $\mathcal{B}$, if $\mathcal{A} \propto \mathcal{B}$, then $\mathcal{B} \in \mathcal{P}$ implies that $\mathcal{A} \in \mathcal{P}$. Equivalently, if $\mathcal{A} \propto \mathcal{B}$, then $\mathcal{A} \notin \mathcal{P}$ implies that $\mathcal{B} \notin \mathcal{P}$. Therefore, if $\mathcal{A} \propto \mathcal{B}$, then $\mathcal{B}$ is at least as hard as a problem $\mathcal{A}$.

Taking the notion of *reducibility* into account, the class of $\mathcal{NP}$-Hard problems can now be formally defined. A problem $\mathcal{A}$ is $\mathcal{NP}$-Hard if $\mathcal{B} \propto \mathcal{A}$ for any problem $\mathcal{B} \in \mathcal{NP}$. A problem is said to be $\mathcal{NP}$-Complete if it belongs to $\mathcal{NP}$ and $\mathcal{NP}$-Hard. The SATISFIABILITY problem was the first problem showed to be $\mathcal{NP}$-Hard through the following theorem in the famous paper by Cook [36].

Theorem 1.1 - Cook's Theorem
If $\mathcal{A} \in \mathcal{NP}$, then $\mathcal{A} \propto$ SATISFIABILITY. ■

After this first step, many other problems were proved to belong in the class of $\mathcal{NP}$-Hard problems as, for this time, it would suffice showing SATISFIABILITY $\propto \mathcal{A}$ in order to prove $\mathcal{A}$ is $\mathcal{NP}$-Hard. It is worth mentioning here the admirable work of Karp [103], where 21 fundamental problems were proved to be $\mathcal{NP}$-Complete. Figure 1.4 illustrates the arrangement of the mentioned complexity classes.

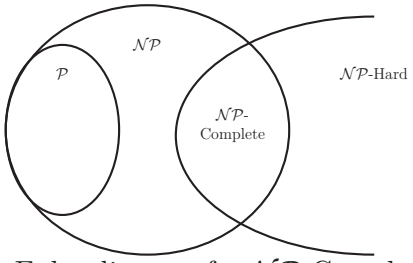


Figure 1.4: Euler diagram for $\mathcal{NP}$-Completeness theory

Whether or not there exists some problem in $\mathcal{NP}$ that is neither in $\mathcal{P}$ nor $\mathcal{NP}$-Complete is a million dollar question and remains a mystery to this day (see Ladner [113]). Indeed, proving whether $\mathcal{P} = \mathcal{NP}$ or $\mathcal{P} \neq \mathcal{NP}$ is one of the greatest unsolved problems of mathematics (see Carlson et al. [28], Clay Mathematics Institute [34]).

The theory of $\mathcal{NP}$-Completeness addresses the issue of classifying a problem according to its complexity. Suppose however that an optimization problem P has been shown to be $\mathcal{NP}$-Hard. The question of how to approach such problem remains unanswered. All we know is that a polynomial time algorithm capable of solving the problem P is not known and that one is unlikely to find such algorithm, unless $\mathcal{P} = \mathcal{NP}$. In this scenario, two main strategies arise. First, if the input sizes are small, a "good" exponential time algorithm might be suitable for solving P.

Second, one may be satisfied with obtaining *near-optimal* solutions in polynomial time instead of looking for the very best solution.

In practice, near-optimal solutions are often good enough. But how to define what is a near-optimal solution? To answer such question, one needs to know how far the solution under consideration is from the actual optimal solution. A polynomial time algorithm that provides a guarantee on the solution quality is called an *approximation algorithm*. In other words, an *approximation algorithm* is an algorithm capable of providing, in polynomial time, solutions that will never differ from the optimal solution by more than a given percentage.

Given an instance of input size n, a $f(n)$-approximation algorithm is an algorithm returning a solution that is at most $f(n)$ times worse than the optimal solution. If *opt* denotes the optimal solution cost of a given optimization problem P and y denotes the cost of the solution provided by the $f(n)$-approximation algorithm, then

$$\max\left\{\frac{opt}{y}, \frac{y}{opt}\right\} \leq f(n).$$

This definition applies for both maximization and minimization problems and $f(n)$ is called the *approximation factor*. If $f(n) = 1$ then the algorithm under consideration is actually an exact algorithm.

Just like with the theory of $\mathcal{NP}$-Completeness, one can classify problems according to their approximation complexity. Problems admitting a constant-factor approximation algorithm running in polynomial time form the class of $\mathcal{APX}$ problems. Problems in $\mathcal{APX}$ class include Vertex Cover, Max Cut and Travelling Salesman Problem with metric distances (*cf.* Vazirani [177], Papadimitriou and Yannakakis [144], Rosenkrantz et al. [161]).

A *polynomial-time approximation scheme* is an approximation algorithm that takes as input an instance of the problem under analysis and a real number $\epsilon \in \mathbb{R}$ such that for any fixed value of $\epsilon > 0$, the algorithm yields, in polynomial time, a solution that is at most $1 + \epsilon$ times worse than the optimal solution. Problems admitting a polynomial-time approximation scheme form the class of $\mathcal{PTAS}$.

Clearly, $\mathcal{PTAS} \subseteq \mathcal{APX}$. An optimization problem is said be $\mathcal{APX}$-Complete if it can be approximated within a constant factor but does not admit an approximation factor of $1 + \epsilon$ unless $\mathcal{P} = \mathcal{NP}$.

1.3 Polyhedra

This section presents some fundamental concepts and results from polyhedral theory. The terminology employed is standard and follows the one applied in Schrijver [164]. For further references on the topic, the reader is referred to Schrijver [163], Bazaraa et al. [12].

The euclidean space of dimension n, denoted by $\mathbb{E}^n$, is the set of all vectors of dimension n. Unless explicitly specified, a vector $x \in \mathbb{E}^n$ will be seen as a column vector, that is, a column array of n numbers. A vector with all its components equal to zero is called the *zero vector* and is denoted by $\mathbf{0}$. Conversely, a vector with all its components equal to one is denoted by $\mathbf{1}$. The *i-th unit vector*, denoted by $\mathbf{e}_i$, is a vector with all its components equal to zero, except for the component in the i-th position, which equals 1. An identity matrix, denoted by I, is a square matrix with ones on the main diagonal and zeros elsewhere, that is

$$I = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}.$$

Given $n \in \mathbb{N}$, $\mathbb{R}^n \subseteq \mathbb{E}^n$ ($\mathbb{Z}^n \subseteq \mathbb{E}^n$, respectively) stands for the set of all vectors of dimension n such that each of its components are real numbers (integer numbers, respectively). Analogously, given a set S, $\mathbb{R}^S$ ($\mathbb{Z}^S$, respectively) is also employed to refer to the set of all vectors of dimension $|S|$ with real (integer, respectively) components, each being associated to one specific element in S. A vector $x \in \mathbb{R}^n$ is said to be *integral* if each of its components are integer values. In other words, $x \in \mathbb{Z}^n$. If at least one of its components is a fraction, it is said to be a *fractional* vector.

A *hyperplane* $H \subset \mathbb{R}^n$ is the set of vectors $x \in \mathbb{R}^n$ defined as $H = \{x : a^T x = b\}$, where a is a nonzero vector in $\mathbb{R}^n$, and b is a real number. In other words, a hyperplane consists of all real vectors x satisfying the equation

$$\sum_{i=1}^{n} a_i x_i = b.$$

A hyperplane can be seen as the generalization of the straight line in $\mathbb{R}^2$, or the plane in $\mathbb{R}^3$. It divides the euclidean space $\mathbb{E}^n$ into two regions, called *halfspaces* denoted by $H^{\geq} = \{x : a^T x \geq b\}$ and $H^{\leq} = \{x : a^T x \leq b\}$. Figure 1.5 illustrates a hyperplane in $\mathbb{R}^2$.

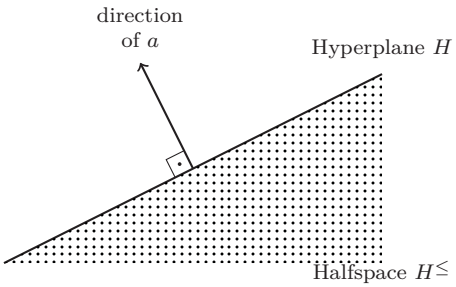


Figure 1.5: Hyperplane and halfspaces

A vector $x \in \mathbb{R}^n$ is said to be a *linear combination* of vectors $a_1, a_2, \ldots, a_k$ in $\mathbb{R}^n$, if

$$x = \sum_{i=1}^{k} \lambda_i a_i,$$

where coefficients $\lambda_1, \lambda_2, \ldots, \lambda_k$ are real numbers. A *conical combination* is a linear combination with non-negative coefficients. A *convex combination* has the additional conditions that $\sum_{i=1}^{k} \lambda_i = 1$ and $0 \leq \lambda_i \leq 1$.

A finite collection of vectors is said to be *linearly independent* if and only if $\lambda_i = 0$ for any $i \in \{1, \ldots, k\}$ is the only solution to

$$\sum_{i=1}^{k} \lambda_i a_i = \mathbf{0}.$$

Analogously, a finite collection of vectors $a_1, a_2, \ldots, a_k$ in $\mathbb{R}^n$ is said to be *affinely independent* if and only if the only solution to the system

$$\begin{cases} \sum_{i=1}^{k} \lambda_i a_i = \mathbf{0}, \\ \sum_{i=1}^{k} \lambda_i = 0. \end{cases}$$

is $\lambda_i = 0$ for any $i \in \{1, \ldots, k\}$. Linear independence clearly implies affine independence, but the opposite might not hold.

A set $S \subseteq E^n$ is said to be *convex* if for any two given points x_a and x_b in X, then

$$\lambda x_a + (1 - \lambda) x_b \in X \qquad \forall \lambda \in [0, 1].$$

In other words, S is a convex set if and only if all points lying on the segment of line between any $x_a \in S$ and $x_b \in S$ belong to S (see Figure 1.6). Notice that a halfspace is, by definition, a convex set. Moreover, the intersection of convex sets is also a convex set.

A *polyhedron* is the intersection of finitely many halfspaces. As a result, a polyhedron is convex. Since each halfspace can be defined through an inequality $a^T x \leq b$,

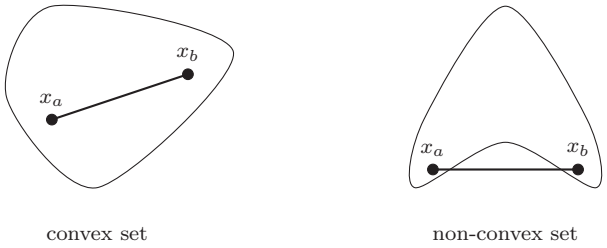


Figure 1.6: Examples of convex and non-convex sets

a polyhedron P can be represented by the system $Ax \leq \mathbf{b}$, where A is the matrix composed by vectors a^T and $\mathbf{b}$ is the vector composed by the independent terms b. In other words, $P = \{x \in \mathbb{R}^n : Ax \leq \mathbf{b}\}$. A *polytope* P is a polyhedron bounded in all directions. That is, there exists a number $k \in \mathbb{R}$ such that $\|x\| \leq k$ for any $x \in P$.

An inequality $a^T x \leq b$ is called an *implicit equality* from polyhedron $P \subseteq \mathbb{R}^n$ if $P \subseteq \{x \in \mathbb{R}^n : a^T x = b\}$. The *dimension* of a polyhedron $P \subseteq \mathbb{R}^n$, denoted by $\dim(P)$, is the maximum number of affinely independent vectors in P minus one. A polyhedron is called *full-dimensional* if one can find $n + 1$ affinely independent vectors in P, that is $\dim(P) = n$.

A vector x in a polyhedron $P = \{x : Ax \leq \mathbf{b}, x \geq 0\}$ is called an *extreme point* of P, if x cannot be written as a strict convex combination of two distinct vectors x_a and x_b in P. That is, if

$$x = \lambda x_a + (1 - \lambda)x_b \in P,$$

with $0 < \lambda < 1$ and $x_a, x_b \in P$, then $x = x_a = x_b$.

Notice that if P is a non-empty polytope, then the set of extreme points is not empty and finite. As a result, any vector $x \in P$ can be written as a convex combination of its extreme points. Figure 1.7 exhibits some examples of extreme and non-extreme points of a non-empty polytope : x_1 is an extreme point whilst x_2 and x_3 are not.

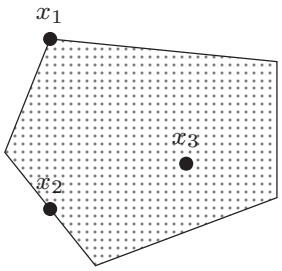


Figure 1.7: Examples of extreme and non-extreme points

A polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is said to be an *integral polyhedron* if and

only if all its extreme points are integral vectors. A matrix A is said to be *totally unimodular* if each non-singular square sub-matrix of A has determinant 0, +1 or −1. A noteworthy link between totally unimodularity and integral polyhedrons is given by the following theorem due to Hoffman and Kruskal [92]:

Theorem 1.2 - Hoffman and Kruskal's theorem
Let A be an integral matrix. Then A is totally unimodular if and only if for each integral vector b, the polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b, x \geq \mathbf{0}\}$ is integral. ■

An alternative characterization of totally unimodular matrices due to Ghouila-Houri [77] is presented in the following theorem:

Theorem 1.3 - Ghouila-Houri's characterization of total unimodularity
Let A be an integral matrix. Then A is totally unimodular if and only if each collection of columns of A can be split into two parts so that the sum of the columns in the first part minus the sum of the columns in the second part is a vector with entries 0, +1, and −1. ■

The following theorem is a result developed in many works (*cf.* Heller and Tompkins [89], Hoffman and Kruskal [92], Hoffman and Kuhn [93], Motzkin [135]) and can easily be deduced from Ghouila-Houri's characterization in Theorem 1.3. It will be particularly useful in Section 4.2.

Theorem 1.4 - Bipartite graphs and total unimodularity
Let $G = (V, E)$ be a graph and let A be the $V \times E$-incidence matrix of G. In other words, A is the $\{0,1\}$-matrix with rows and columns indexed by the vertices and edges of G, respectively, such that $A_{ve} = 1$ if and only if $v \in V$ is an endpoint of $e \in E$. Then A is totally unimodular if and only if G is bipartite. ■

1.4 Linear optimization

A *linear optimization problem* (LP) consists of finding the vector $x \in \mathbb{R}^n$ that maximizes (or minimizes) a linear function $f(x)$ over a finite set of linear inequalities $Ax \leq b$, where A is a $n \times m$ matrix and $b \in \mathbb{R}^m$. Therefore, a linear optimization problem can be stated as

$$\max \left\{ f(x) = c^T x : Ax \leq b \right\}. \tag{1.1}$$

Since a polyhedron P can be represented by a system of linear inequalities (*i.e.* $P = \{x : Ax \leq b\}$), the optimization problem can be equivalently represented by

$$\max\left\{c^T x : x \in P\right\}.$$

A vector $x \in \mathbb{R}^n$ is a *feasible solution* for the system of linear inequalities $Ax \leq b$ if it satisfies all inequalities in $Ax \leq b$. If the system of inequalities does not admit any feasible solution, that is $P = \emptyset$, the problem is said to be *infeasible*.

The linear function $c^T x$ is called the *objective function*, and vector $c \in \mathbb{R}^n$ is also known as the *gradient* since it gives the direction on which the objective function grows. A feasible solution x that maximizes (or minimizes) the objective function is called an *optimal solution*. If a problem has a feasible solution but does not have an optimal solution (*i.e.*, the associated polyhedron is not bounded in the direction of the gradient), the problem is said to be *unbounded*.

Theorem 1.5 gives a relation between optimal solutions of $\max\left\{c^T x : x \in P\right\}$ and the extreme points of P. We consider here only the case where P is a non-empty polytope, since the problems treated in this dissertation do not admit an unbounded polyhedron. A more general result, considering the case where P is any polyhedron, can be found in Bazaraa et al. [12, p. 91].

Theorem 1.5 - Optimal solutions vs. Extreme points
Given a non-empty polytope $P = \{x : Ax \leq b\}$, the optimal solution of the linear optimization problem $\max\left\{c^T x : x \in P\right\}$ occurs in at least one extreme point of P.

Proof. Let $\bar{x_1}, \bar{x_2}, \ldots, \bar{x_k}$ denote the extreme points of P. Recall that any point in P can be written as the convex combination of its extreme points. Therefore, the optimization problem can be rewritten as:

$$\begin{array}{lllll} \text{maximize} & \sum_{i=1}^{k} (c^T \bar{x_i})\lambda_i & & & \\ \text{subject to} & \sum_{i=1}^{k} \lambda_i & = & 1 & \\ & \lambda_i & \geq & 0 & \forall i = 1, \ldots, k \end{array}$$

To solve this problem, one may simply search for the smallest $c^T \bar{x_i}$, say $\bar{x_p}$, and assign $\lambda_p = 1$ while all other λ_i's get a value of zero. The extreme point $\bar{x_p}$ is thus an optimal solution of $\max\left\{c^T x : x \in P\right\}$. ■

Based on this strong property of linear optimization problems, George B. Dantzig developed in 1947 the famous *simplex* method (see Dantzig [44]), which looks for a starting extreme point and walks through the edges of the polyhedron, jumping between adjacent extreme points, according to the problem's gradient. Figure 1.8 illustrate the procedure of simplex algorithm over a polyhedron. For further details

on the simplex method and how it was conceived, the reader is referred to Dantzig [45, p. 94-119] and Dantzig [46].

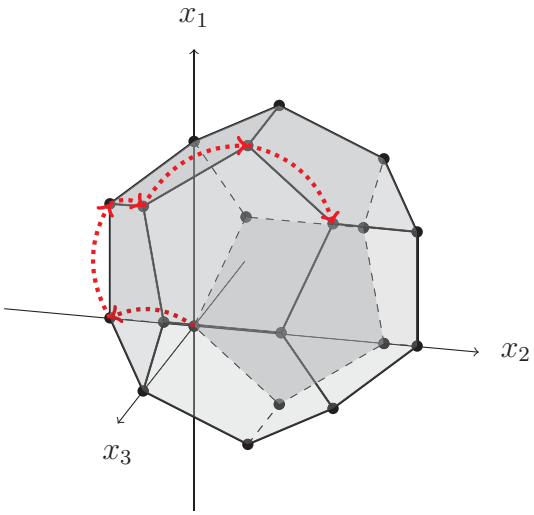


Figure 1.8: Simplex method overview

Simplex method, although quite efficient in practice, was shown to have exponential behaviour in some well-designed instances by Klee and Minty [107]. It was not until 1979 that solving a linear optimization problem was shown to be possible in polynomial time with the development of *ellipsoid* and *interior-point* methods, due to Khachiyan [104] and Karmarkar [102], respectively.

If a linear optimization problem is stated as indicated in (1.1), then its *dual* problem is

$$\min \left\{ g(y) = y^T b : y^T A \geq c \right\}. \tag{1.2}$$

Conversely, problem (1.1) is called the *primal* problem. Notice that the dual of (1.2) is exactly the problem (1.1).

Duality is a key concept in linear optimization that develops the relationship between two specific linear optimization problems – the primal and dual problems. We restate here two of the most important results concerning duality attributed to Von Neumann [178] and Gale et al. [71], that shall be helpful for some of the results proposed in this dissertation. For further material about duality theory, the reader is referred to Dantzig [45, p. 120-144].

Theorem 1.6 - Weak Duality Theorem
If x and y are feasible solutions for (1.1) *and* (1.2)*, respectively, then*

$$y^T b \geq c^T x. \qquad \blacksquare$$

As a result, any feasible solution of problem (1.2) provides an upper bound to problem (1.1). Conversely, any feasible solution of problem (1.1) provides a lower bound to problem (1.1). Moreover, if the primal is unbounded, then the dual is infeasible (and vice-versa).

Theorem 1.7 - Strong Duality Theorem
If x and y are feasible solutions of equal objective function value for (1.1) *and* (1.2)*, respectively, then x and y are optimal solutions of the respective problems.* ■

1.5 Integer and combinatorial optimization

An *integer linear optimization problem* (ILP) is a linear optimization problem with the additional restriction that each variable admits only integer values. The *formulation* of an ILP can thus be stated as

$$\max\left\{f(x) = c^T x : Ax \leq b, x \in \mathbb{Z}^n\right\},$$

or equivalently as

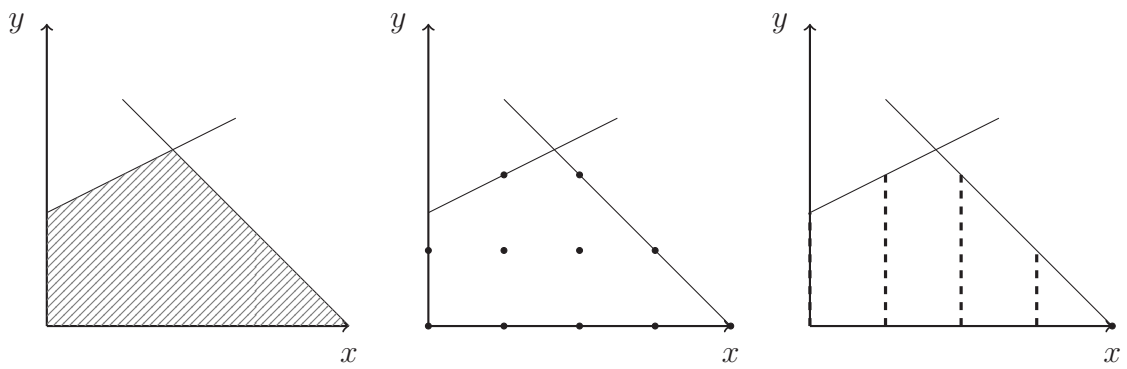
$$\max\left\{f(x) = c^T x : x \in P, x \in \mathbb{Z}^n\right\},$$

where P is a polyhedron.

If only a subset of the variables has this additional restriction, the problem called a *mixed-integer linear optimization problem* (MILP), and can be stated as

$$\max\left\{f(x,y) = c^T x + h^T y : Ax + Dy \leq b, x \in \mathbb{Z}^n, y \in \mathbb{R}^p\right\}.$$

Figure 1.9 illustrates the difference of the feasible solution sets of a LP, an ILP and a MILP over the polytope $P = \{(x,y) : 2y - x \leq 3, y + x \leq 4, x \geq 0, y \geq 0\}$.



(a) LP: $x \in \mathbb{R}, y \in \mathbb{R}$ **(b)** ILP: $x \in \mathbb{Z}, y \in \mathbb{Z}$ **(c)** MILP: $x \in \mathbb{Z}, y \in \mathbb{R}$

Figure 1.9: Illustration of feasible solution sets

Given a finite subset $S \subseteq \mathbb{R}^n$, the *convex hull* of S, denoted by $\text{conv}(S)$, is the smallest convex set containing S. In other words, $\text{conv}(S)$ is the set of all vectors in $\mathbb{R}^n$ that can be written as the convex combination of vectors in S.

The following fundamental theorem of integer programming, due to Meyer [132], states that given a polyhedron $P \subseteq \mathbb{R}^n$, the convex hull $\text{conv}(P \cap \mathbb{Z}^n)$ is also a rational polyhedron. Figure 1.10 illustrates such fact.

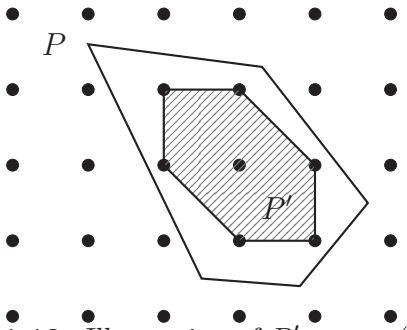


Figure 1.10: Illustration of $P' = \text{conv}(P \cap \mathbb{Z}^2)$

Theorem 1.8 - Fundamental Theorem of Integer Programming
Let $P = \{(x, y) : Ax + Dy \leq b\}$ and $S = \{(x, y) \in P : x \in \mathbb{Z}^n\}$, where A and D are rational matrices and b is a rational vector. Then, there exists rational matrices A', D' and a rational vector b' such that

$$conv(S) = \{(x, y) : A'x + D'y \leq b'\}. \qquad \blacksquare$$

To its fullest extent, Theorem 1.8 precises that solving the LP $\max\{c^T x : x \in \text{conv}(P \cap \mathbb{Z}^n)\}$, yields an optimal solution for the ILP $\max\left\{c^T x : x \in P, x \in \mathbb{Z}^n)\right\}$. Moreover, from Figure 1.10 it is easy to see that one single ILP may have an infinite number of formulations. Indeed, two polyhedrons P_1 and P_2 induce valid formulations for an ILP if and only if $P_1 \cap \mathbb{Z}^n = P_2 \cap \mathbb{Z}^n$.

Given a finite set $N = \{1, \ldots, n\}$ and weights $c_i \in \mathbb{R}$ for each $i \in N$, let $\mathcal{F}$ be a family of feasible subsets $F \subseteq N$ (feasible solutions). Then, finding a maximum (or minimum) weight feasible subset is called a *combinatorial optimization problem*. In other words, a combinatorial optimization problem can be stated as

$$\max\left\{\sum_{i \in F} c_i : F \in \mathcal{F}\right\}. \tag{1.3}$$

For a given set $F \in \mathcal{F}$, the *incidence vector* $x^F \in \{0, 1\}^n$ is defined as follows

$$x_i^F = \begin{cases} 0, & \text{if } i \in F \\ 1, & \text{otherwise.} \end{cases}$$

It follows directly that the combinatorial optimization problem (1.3) may be rewritten as

$$\max\left\{\sum_{i\in N} c_i x_i^F : x^F \in S\right\},$$

where $S \subseteq \{0,1\}^n$ is the set of incident vectors of $\mathcal{F}$.

Finally, from Theorem 1.5 and Theorem 1.8, it is easy to see that

$$\max\left\{\sum_{i\in N} c_i x_i^F : x^F \in S\right\} = \max\left\{\sum_{i\in N} c_i x_i^F : x^F \in \text{conv}(S)\right\}.$$

In other words, the combinatorial problem (1.3) reduces to a linear problem over the convex hull of its feasible solutions. For an example of application of such theory, the reader is referred to the beautiful paper of Edmonds [59].

Gathering these results together with the fact that linear optimization problems can be solved in polynomial time (through ellipsoid and interior point methods), one might imagine that ILP's can be solved in polynomial time, once the convex hull of its feasible solutions, $\text{conv}(S)$, is known. Unfortunately, obtaining a complete linear description of $\text{conv}(S)$ is usually a difficult task. It turns out that unlike for LP's, no polynomial-time algorithm for solving a general ILP is known. Indeed, the problem of solving an ILP has been shown to be $\mathcal{NP}$-Complete. With a polynomial reduction from Satisfiability problem, Karp [103] showed its membership to the class of $\mathcal{NP}$-Hard problems and its membership in $\mathcal{NP}$ was proved in Borosh and Treybig [18].

Two of the most famous algorithms for solving ILP's (and MILP's) are based on the concept of *linear relaxation*. The following statements will be given for ILP's but can obviously be extended for MILP's. Given an ILP $\max\{f(x) = c^T x : Ax \leq b, x \in \mathbb{Z}^n\}$, its *linear relaxation* is the LP $\max\{f(x) = c^T x : Ax \leq b, x \in \mathbb{R}^n\}$ obtained by dropping the integrality constraints.

The *Branch-and-Bound* algorithm proposed by Land and Doig [115] has been largely applied for solving $\mathcal{NP}$-Hard optimization problems. Given a maximization ILP $\max\{c^T x : x \in P, x \in \mathbb{Z}^n\}$, the Branch-and-Bound algorithm consists of subdividing P into increasingly smaller polyhedrons through the *branching procedure*. For this, one has to solve the ILP's linear relaxation. If the optimal solution found x^* is integer, then it is also an optimal solution for the ILP. Otherwise, given a fractional component x_i^* of x^*, one constructs two new polyhedrons $P_1 = \{x \in P, x_i \geq \lceil x_i^* \rceil\}$ and $P_2 = \{x \in P, x_i \leq \lfloor x_i^* \rfloor\}$. Such procedure is repeated, constructing the so-called Branch-and-Bound tree, until every leaf of the tree yields an integer solution or no solution at all (characterizing infeasibility). Then, the optimal solution of the ILP is

the best solution among all such integer solutions. In order to shorten the branching procedure, branches on which there are proofs that one cannot find an optimal solution inside are pruned. Such proofs are gathered through the continuous updates on the upper and lower bounds and will be discussed more extensively later. For supplementary information on how Branch-and-Bound works the reader is referred to Wolsey [181, p.91].

The algorithm's performance relies considerably on obtaining "good" bounds. Notice that the linear relaxation of an ILP yields an upper bound for a given maximization ILP, that is

$$\max\left\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\right\} \leq \max\left\{c^T x : Ax \leq b, x \in \mathbb{R}^n\right\}.$$

Similarly, the linear relaxation yields a lower bound for a minimization ILP. For this reason, given two formulations P_1 and P_2 for the same ILP, P_1 is said to be stronger than P_2 if $P_1 \subset P_2$. Alternatively, P_1 is said to be stronger than P_2 if the linear relaxation of P_1 provides a better bound than the linear relaxation of P_2 for any given objective function. That is,

$$\max\left\{c^T x : x \in P_1\right\} \leq \max\left\{c^T x : x \in P_2\right\} \qquad \forall c \in \mathbb{R}^n.$$

Let $\text{OPT} = \max\left\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\right\}$ denote the optimal solution value of the ILP, and $\text{LR} = \max\left\{c^T x : Ax \leq b, x \in \mathbb{R}^n\right\}$ denote the optimal solution value of its linear relaxation. A standard indicator of the strength of an ILP formulation is the *integrality gap* which is defined as the percentage difference between OPT and LR, that is,

$$\text{Gap} = \frac{\text{LR} - \text{OPT}}{\text{OPT}}.$$

For a minimization problem, $\text{Gap} = \frac{\text{OPT}-\text{LR}}{\text{OPT}}$.

Given $\alpha \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$, an inequality $\alpha^T x \leq \beta$ is said to be a *valid inequality* for a set P if $\alpha^T x \leq \beta$ holds for any vector $x \in P$. With regard to integer programming, searching for valid inequalities is a major field of research. An emblematic example of application in which such theory can be found is over the Uncapacitated Facility Location (UFL) problem (see Jakob and Pruzan [96], Cornuéjols et al. [41]). The first attempts of using the principle of valid inequalities in order to tighten the integrality gap of a given "natural" formulation of UFL problem are due to Balinski [11], Manne [124], Kuehn and Hamburger [112].

Indeed, cleverly introducing valid inequalities to an ILP formulation $\max\{c^T x : x \in P, x \in \mathbb{Z}^n\}$ is a typical approach for reinforcing it. For this, the introduced

inequalities should be capable of cutting off solutions in $P \backslash \text{conv}(P \cap \mathbb{Z}^n)$. Figure 1.11 gives an illustration on the subject where the dashed area represents the solutions that are cut off by the valid inequality.

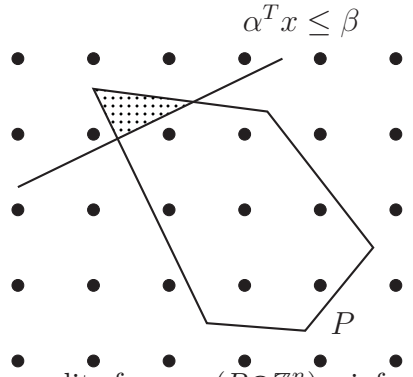


Figure 1.11: A valid inequality for $\text{conv}(P \cap \mathbb{Z}^n)$ reinforcing the ILP formulation

For some valid inequality $\alpha^T x \leq \beta$ of a polyhedron P, $F = \{x \in P : \alpha^T x = \beta\}$ is called a *face* of P. If a face F of P has $\dim(F) = \dim(P) - 1$, it is called a *facet* of P. In other words, F is called a *facet* of P if F is a maximal face of P. The valid inequality inducing a facet is called *facet-defining* (see Figure 1.12).

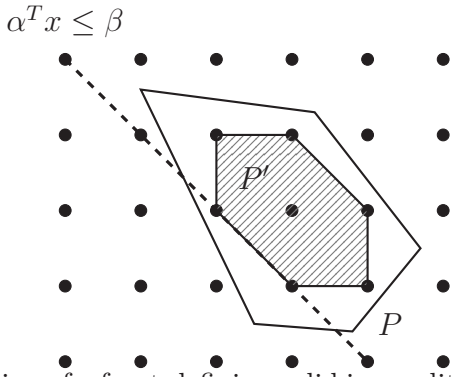


Figure 1.12: Illustration of a facet-defining valid inequality of $P' = \text{conv}(P \cap \mathbb{Z}^2)$

Another method for solving ILPs, based on the continuous strengthening of the ILP formulation, is the (60 years old!) iconic and well-studied *cutting plane* algorithm due to Gomory et al. [80]. Unlike the Branch-and-Bound algorithm, the cutting plane principle does not consists on dividing the original polyhedron P but on carving polyhedron P, through the successively insertion of wisely chosen valid inequalities (cutting planes) into P. For this, one starts by (again) solving the ILP's linear relaxation. If the optimal solution yielded is not integer, one searches for an inequality that holds for any integer solution in P but is capable of cutting off the current fractional solution found. The insertion of such inequality generates a new polyhedron P' such that $\text{conv}(P \cap \mathbb{Z}^n) \subseteq P' \subset P$. This procedure is repeated until an integer solution is found. For supplementary information on how cutting plane

algorithm works the reader is referred to Conforti et al. [35] and Wolsey [181, p.113].

Given a polyhedron $P \subseteq \mathbb{R}^n$ and a vector $x \in \mathbb{R}^n$, the *separation problem* for P is to decide whether or not x belongs to P and, if not, provide an inequality $\alpha^T x \leq \beta$ that is valid for P but violated by x. The equivalence between optimization and separation was showed through Theorem 1.9 due to Grötschel et al. [82].

Theorem 1.9 - Optimization ≡ Separation
The linear optimization over $\max\{c^T x : x \in P\}$ *is polynomially solvable if and only if the separation problem for* P *is polynomially solvable.* ■

This means that an ILP $\max\{c^T x : x \in P, x \in \mathbb{Z}^n\}$ can be solved in polynomial time if and only if the separation problem for $\text{conv}(P \cap \mathbb{Z}^n)$ can be solved in polynomial time, unless $\mathcal{P} = \mathcal{NP}$.

In the context of integer and combinatorial optimization, the two presented methods - Branch-and-Bound and cutting plane - are often combined into the so-called *Branch-and-Cut* algorithm. In this case, cutting planes are generated in each node of the Branch-and-Bound tree with the intention of tightening the current bounds and thus providing a stronger pruning phase.

Further references on linear and integer optimization are Chvatal et al. [33], Schrijver [163], Wolsey [181].

Chapter 2

The stop number problem

> "The greatest challenge to any thinker is stating the problem in a way that will allow a solution."
>
> — Bertrand Russell

In this chapter the context on which the Stop Number problem arises is introduced. The problem is then formally defined and the main related works are presented. Finally, we present the IP formulation on which the integrality of our study is based on.

2.1 Context

In modern societies, mobility plays a central role in economic and social activities such as commuting, manufacturing, distributing goods, or supplying energy (see Rodrigue et al. [158]). Mobility is supported and driven by transport systems allowing interactions among individuals, institutions and nations. In Delbosc [49], the role of transport systems in the development of the well-being and life satisfaction of a community is highlighted.

It is undeniable that transport systems have constantly progressed throughout human civilization's history. Nonetheless, the demand for faster, cheaper and more convenient forms of mobility has correspondingly evolved. Today, the world is changing at a faster pace than ever. According to the United Nations Department of Economic and Social Affairs [175], 55% of the world's population is concentrated today in urban areas and such proportion is expected to reach 68% by 2050. At the same time, the global demand for passenger mobility in urbanized areas – in terms of passengers-kilometers per year – is expected to double within thirty years (see Little [120]). The scenario for the transport of goods is even worse due to the

significant expansion of the e-commerce sector and the consequent boom in demand for last-mile delivery. As a matter of fact, e-commerce represented a 2.3 trillion US$ market in 2017 and an expressive growth is expected for the years to come (see Statista [170]). Figure 2.1 illustrates such progression. Furthermore, the number of packages and parcels delivered annually in the United States is expected to expand from 11 billion in 2018 to 16 billion by 2020 (see Laseter et al. [116]). One of the major challenges facing the world today is therefore to implement an extensive reform of traditional mobility systems, particularly in urban areas.

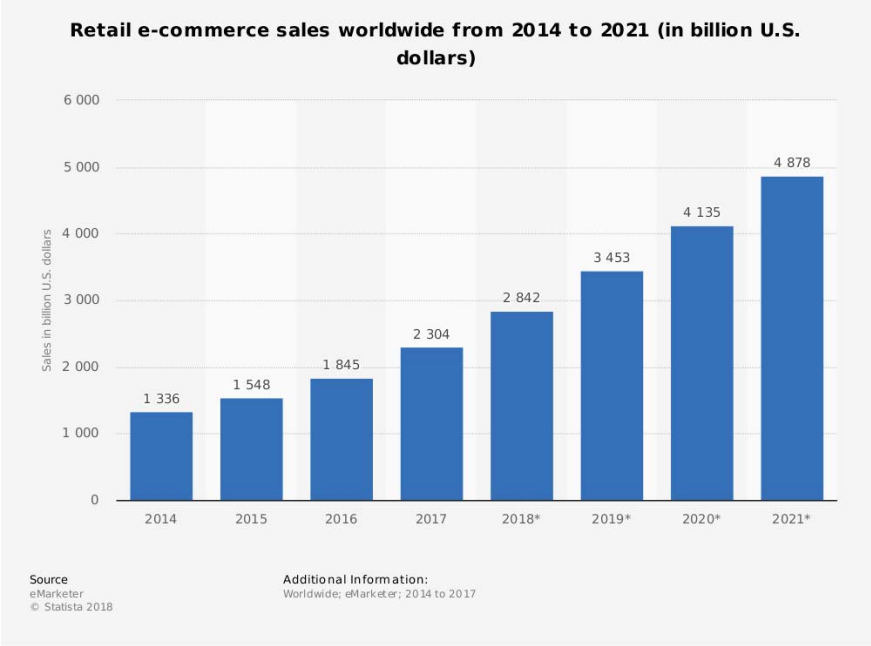


Figure 2.1: Retail e-commerce sales worldwide from 2014 to 2021 (in billion U.S. dollars)

As stated by Gao et al. [72], Ehlers [64], the task of rethinking mobility is currently being guided by four major technology-driven trends in transportation: connectivity, autonomous driving, diverse mobility services and electrification. Each of these trends has the power to provide important innovations, but the combination of them gives rise to the so-called intelligent mobility which has the potential to deliver a true revolution to the sector. All in all, the intelligent mobility global market is forecast to be worth more than 1 trillion US$ by 2030.

Connectivity. In Merlin [131], the perfect transport system is characterized by its low cost (ideally free of charges), instantaneous travel time and unlimited capacity and availability. In this sense, the transport of data has recently achieved impressive standards. Indeed, by having access to the internet, one is able to exchange messages on demand between peers of a network incredibly fast (if not instantly), at a negligible price. It is worth noting that the amount of connected sensors and data generators is growing at over 30% each year and the Internet of Things (IoT)

is predicted to contain over 24 billion interconnected devices by 2020, as reported in Gubbi et al. [88]. In this scenario, by 2030 all new European vehicles are expected to be part of such network, being capable of communicating with each other and with other infrastructures (see European Automobile Manufacturers Association [65]). Such level of connectivity can allow an optimised traffic flow management through a better understanding and control of time and space in roads and urban areas.

Autonomous driving. Autonomous Vehicles (AVs) refer to vehicles with the capacity of navigating without human intervention. Several studies can be found in the literature linking the development of AVs to improvements on traffic mitigation and safety (*e.g.*, Talebpour and Mahmassani [172], Fagnant and Kockelman [66], Kim et al. [105], Ozguner et al. [140]). Indeed, at least 90% of motor vehicle crashes are caused by human error, as reported in Singh [168]. In addition, the deployment of fully automated vehicles promises a revolution on the driving experience as users would be able to focus on other activities such as working, relaxing, or accessing entertainment. Finally, AVs have the potential to allow social benefits by providing independent mobility to excluded people such as younger, elderly or disabled passengers (see KPMG International [109]).

Electric vehicles. The concept of sustainability has received increasing attention in the recent decades (World Commission on Environment and Development [182]). The current form of individual motorized transportation represents a meaningful environmental threat. Recently, French government has announced a draft law to put an end to the sale of petrol and diesel cars by 2040 (see De Rugy and Borne [48]), even if these still represent today 95% of sales. In this scenario, electric vehicles represent flourishing market. While in 2017 only 3 million vehicles had electric engines, perspectives are that, due to supportive government policies and cost reductions, such number will raise to 125 million by 2030 (see Bunsen et al. [27, p.11]).

Diverse mobility. Tightening CO_2 regulations and people's growing consciousness over environmental issues together with the struggle of urban centers to accommodate the ever growing traffic flow have transformed the image of traditional motorized vehicles in the collective mindset and made it synonymous with pollution, inconvenience, noise and stress. Alongside with this, the current model for individual motorized mobility is incredibly inefficient as an average private vehicle stands idle (*i.e.*, parked at home or elsewhere) for 96% of the time (RAC Foundation [155, p.23]). An executive survey (KPMG International [108]) reveals that the majority of today's urban drivers will not want to own a car by 2025, even if demand for mobility is only increasing within urban areas. Such mentality change is giving rise to new transport systems proposing a more flexible, reactive and responsible way of

dealing with customers mobility demands. Among these systems one may refer to on-demand private ride hiring (*e.g.*, Uber, Lyft) and vehicle sharing (*e.g.*, Moov'in, car2go, Lime). Such systems can either perform in a free-floating network where vehicles can be picked up and left parked anywhere, or within hub/depot structures on which vehicles must be left in predefined parking stations.

New transport systems are striving to find their place between the established fully individual transportation and the standard collective mobility solutions (*e.g.*, buses, subways and trams). A common key component for the successful implementation of such new systems is the capability of providing a complete (or almost complete) coverage of destinations for its users. This can be done by either by ensuring that clients and/or goods are picked up and delivered directly at their origin and destination points, or by providing a multi-modal integration of synchronized systems. VIPAFLEET project arises from this background by focusing on contributing to sustainable intelligent mobility through the development of models and algorithms for managing fleets of specific autonomous vehicles named VIPA, a French acronym[1] for Autonomous Individual Passenger Vehicle. VIPA is an electrical vehicle, developed by Ligier[2] and Easymile[3], and designed to operate in fully autonomous manner (*i.e.*, without any driver assistance), notably in semi-closed and closed sites like medical complexes, commercial or industrial areas and campuses. The most recent VIPA version named EZ10 is illustrated in Figure 2.2.

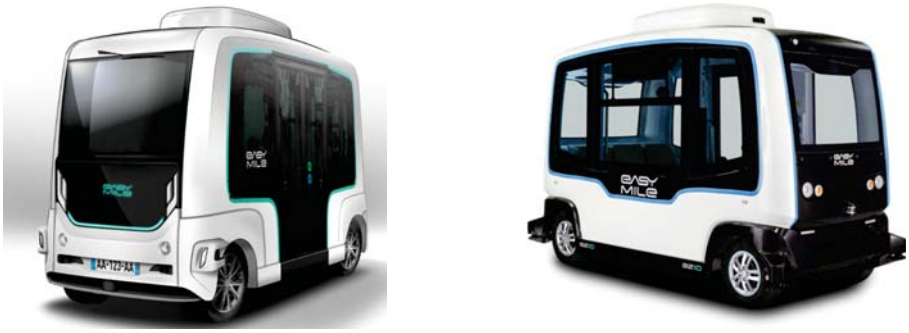


Figure 2.2: Autonomous shuttle EZ10 by Easymile and Ligier

The VIPA shuttles are designed to operate in different circulation modes:

- **Tram mode:** Shuttles travel around a predefined circuit always in the same direction and stop at a station upon request.
- **Elevator mode:** Shuttles perform as an horizontal elevator, travelling on a predefined line, reacting to users demands and therefore changing its driving direction accordingly.

[1] *Véhicule Individuel Public Autonome*, in French.
[2] `https://www.ligier.fr/nos-gammes/ez10.html`, accessed on 07-06-2019
[3] `https://easymile.com/`, accessed on 07-06-2019

- **Taxi mode:** Shuttles serve transport requests (defined by an origin station and a destination station) smartly choosing its path through a connected network.

It is worth noting that a VIPA shuttle can transport more than one passenger at the same time. Its latest version, EZ10, is designed to have a capacity of up to 15 passengers. This leads to a Pickup-and-Delivery (PDP) problem on which a fleet of capacitated vehicles is responsible for transporting clients or goods that must be moved from certain pickup locations to other delivery locations on a given network. A state of the art of the PDP and its variants is given in Section 2.3.

In practice, the transport system must be reactive and the fleet of vehicles should respond dynamically to the on-going flow of user demands through online algorithms. However, in order to better evaluate such reactive procedures, the static case (also called offline case) where demands are known in advance should be understood and mastered. Conversely, a good understanding of the properties and difficulties of the static case is essential to the development of better suited online algorithms. This thesis targets systems operating in tram mode and treats its offline case. For a study on other modes in the online case, the reader is referred to Bsaybes [25].

VIPAFLEET project is a collaboration of numerous partners in order to guarantee the reliability of such innovative transport system. Apart from the VIPA manufacturers Ligier and Easymile, other industrial partners, namely Michelin[4] and Exotic Systems[5], also supports the project. The research and development function is funded by the laboratory of excellence (LabEx) Innovative Mobility: Smart and Sustainable Solutions[6] (IMobS3) and conducted by the research unities Laboratory of Computing, Modelling and Optimization of the Systems[7] (LIMOS) and Institut Pascal[8].

2.2 Problem definition

2.2.1 General presentation

The Stop Number Problem (SNP), first introduced in Pimenta et al. [151], arises from the management of a fleet of VIPA shuttles performing in tram mode. In this mode of operation, a circuit with predefined stations is fixed. A special station is

[4] `https://www.michelin.fr/`, accessed on 07-06-2019.
[5] `https://www.exotic-systems.com/`, accessed on 07-06-2019.
[6] `http://www.imobs3.uca.fr/index.php/en/`, accessed on 07-06-2019.
[7] `https://limos.fr/presentation`, accessed on 07-06-2019.
[8] `http://www.institutpascal.uca.fr/index.php/en/`, accessed on 07-06-2019.

denoted the depot. Customers use their smartphones or a 'call terminal' to request for a ride from an origin station to some destination station of their choice. For its part, the fleet of identical capacitated vehicles travels around the circuit (always in the same direction) and stops at a station upon request.

Due to infrastructure restrictions, stations usually do not belong to the circuit but are attached to it (see Figure 2.3). This particularity produces a significant impact on the fleet management of such a system. Indeed, in order to respond to a client demand, the vehicle must slow down and make a deviation from its original course. Such deviations increases the travel times of on-board customers as well as the vehicle's battery consumption, a key resource for electrical vehicles. If the deviations lengths are supposed to be approximately the same, then improving the quality of service fairly corresponds to minimizing the total number of stops performed by the fleet of vehicles. Notice that if this is not the case, one may focus on minimizing the total deviation length. In Pimenta et al. [151], it is also pointed out that minimizing the total number of stops is a good way of improving the system's reliability by ensuring a steady flow of the vehicles.

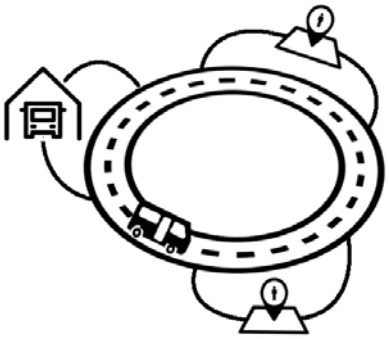


Figure 2.3: Circuit Scheme

The *Stop Number Problem* (SNP) consists of assigning each client demand to a vehicle such that no vehicle gets overloaded, and the total number of vehicles' stops is minimized. For this, one may use as many vehicles as desired. Notice that, in the search for a better solution, a vehicle is allowed to make several tours before serving a demand. Tours performed before serving a demand are called *waiting tours*. In order to ensure the quality of service and deal with customers time windows, the maximum number of waiting tours is bounded by a given parameter $H \geq 0 \in \mathbb{N}$. Moreover, once a customer is picked up, it cannot stay on the vehicle for a full tour, that is, once loaded it has to be unloaded as soon as the vehicle reaches its destination. Finally, a customer demand may request for more than one seat on a single vehicle. In this sense, a demand is specified by an origin station, a destination station and a load that stands for the number of seats requested.

In order to investigate the core complexity of SNP, this thesis focus on a constrained version of SNP where no waiting tour is allowed (*i.e.*, $H = 0$) and each demand can only request for one seat at a vehicle. It is important to mention that even if each demand has an unity load, multiple demands having the same origin and destination are allowed. Such constrained variant, is hereafter denoted Unit Stop Number Problem (U-SNP). Next, the U-SNP is formally described.

2.2.2 The Unit Stop Number Problem

Let $V = \{1, \ldots, n\}$ be the set of predefined stations numbered as they appear along the circuit network. Notice that the depot does not belong to V. For any two subsets $S_1 \subseteq V$ and $S_2 \subseteq V$, we say $S_1 \prec S_2$ ($S_1 \succ S_2$), if all stations in S_1 appear before (after) all stations in S_2 on the circuit. In other words, $S_1 \prec S_2$ if $i < j$ for any $i \in S_1$ and any $j \in S_2$. Let E be the set of m unit-load dial-a-ride demands, where each demand e is specified by a pick-up (origin) station $o_e \in V$ and a drop-off (destination) station $d_e \in V$ with $o_e < d_e$, that is, $e = (o_e, d_e)$. Without loss of generality[9], we assume that each station of V appears as an endpoint of at least one demand of E.

To serve these m demands, we are given a fleet of p identical vehicles, each of them having the same capacity $C \in \mathbb{Z}^+$. Let $K = \{1, \ldots, p\}$ denote this set of available vehicles. Throughout this dissertation we make the assumption that the number of available vehicles is not a crucial resource. In other words, the decision maker has the choice to use as many vehicles as desired in order to reduce the total number of stops. With this in mind, unless explicitly specified, the number of available vehicles p is set to a trivial upper bound m (*i.e.*, $p = m$).

Notice however that reducing the number of stops prevents the use of an *unreasonable* number of vehicles (this interdependence is further explained in sections 3.1 and 5.2). Moreover, unless explicitly specified, the results presented along this thesis can easily be extended to the case where the number of available vehicles p is part of the instance.

An instance of U-SNP is then defined by the network's stations V, the client demands E and the capacity C of the available vehicles. With any U-SNP instance $\mathcal{I} = (V, E, C)$, a graph $G_{\mathcal{I}} = (V, E)$ can be associated[10] and henceforth, stations and demands may be referred to as nodes and edges, respectively. Notice that since $o_e < d_e$ for any $e \in E$, the associated graph $G_{\mathcal{I}}$ has a natural orientation.

[9] If it is not clear to the reader why the statement is true, Property 1 explicitly exposes the reasons.

[10] When instance $\mathcal{I}$ is clear from the context, we may omit the subscript $\mathcal{I}$.

Nonetheless, we usually represent it as an undirected graph for simplicity. A set of five demands represented as intervals over four stations and the associated graph are shown in Figure 2.4.

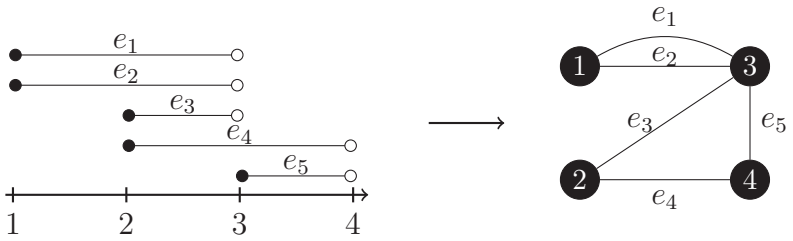


Figure 2.4: Construction of the associated graph $G_{\mathcal{I}}$ from instance $\mathcal{I}$.

Demands sharing the same extremities (*e.g.*, e_1 and e_2 in Figure 2.4) are referred to as parallel demands and correspond to multiple edges in $G_{\mathcal{I}}$. Without loss of generality, we suppose that $G_{\mathcal{I}}$ is a (loopless) connected multigraph for otherwise solving U-SNP on $G_{\mathcal{I}}$ would reduce to independently solving as many U-SNPs as $G_{\mathcal{I}}$ has connected components.

For any subgraph H of $G_{\mathcal{I}}$, and any station $v \in V(H)$, let

$$\delta^-_H(v) = \{e \in E(H) : d_e = v\} \quad \text{and} \quad \delta^+_H(v) = \{e \in E(H) : o_e = v\}$$

denote the sets of demands in $E(H)$ that have v as their destination and origin station, respectively. For the instance depicted in Figure 2.4, $\delta^-_{G_{\mathcal{I}}}(3) = \{e_1, e_2, e_3\}$ and $\delta^+_{G_{\mathcal{I}}}(3) = \{e_5\}$.

For any subset $F \subseteq E$ and any station $v \in V$, let

$$\Delta_F(v) = \{e \in F : o_e \leq v \leq d_e - 1\}$$

denote the set of demands in F that *cross* or *starts* at station v. Notice that demands having v as their destination station do not belong to $\Delta_F(v)$. All demands belonging to $\Delta_E(v)$ are said to *intersect* station v. On the associated graph G, the set of demands intersecting station v, that is $\Delta_E(v)$, is defined by the cut-set $\delta_G(\{1, \ldots, v\})$. Figure 2.5 illustrates $\Delta_E(v)$, for each $v \in V$, in the instance described in Figure 2.4.

Every station v whose set $\Delta_E(v)$ is inclusion-wise maximal is referred to as a *maximal-intersection station*. In other words, a station $v \in V$ is said to be a *maximal-intersection station* if there exists no station $v' \in V$ for which $\Delta_E(v) \subset \Delta_E(v')$. For the example depicted in Figure 2.4, both stations 2 and 3 are maximal-intersection, but stations 1 and 4 are not.

Any feasible solution to U-SNP is a partition of E into p subsets $E_1, \ldots, E_p$ that

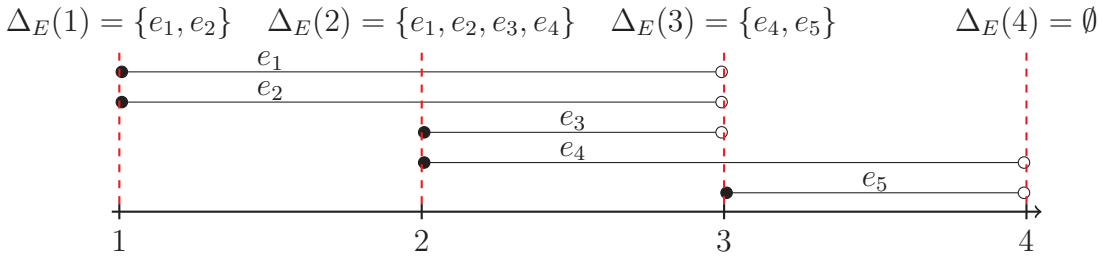


Figure 2.5: Illustration of $\Delta_E(v)$.

satisfy $|\Delta_{E_i}(v)| \leq C$ for $i = 1, \ldots, p$ and all $v \in V$. U-SNP thus consists of finding a partition $\{E_1, \ldots, E_p\}$ that minimizes the cost function

$$c(\{E_1, \ldots, E_p\}) = \sum_{i=1}^{p} |V[E_i]|$$

where $V[E_i]$ represents the set of stations vehicle $i \in \{1, \ldots, p\}$ must stop at. To fix ideas, consider again the example depicted in Figure 2.4 with $C = 2$. Then a feasible solution with five stops (*i.e.*, $c(\{E_1, \ldots, E_p\}) = 5$) is $E_1 = \{e_1, e_2\}$, $E_2 = \{e_3, e_4, e_5\}$, and $E_i = \emptyset$ for $i \in \{3, 4, 5\}$.

A minor relaxation to the problem is established below. From now on, we assume that a vehicle is allowed to naively stop at a station even when not requested (*i.e.*, no pick-up or drop-off operation is expected). Such relaxation offers a greater freedom for future analysis of the solutions of U-SNP without impacting the structure of the optimal solutions as ensured by Property 1,

Property 1 - Trivial remark
Let $\mathcal{I}$ be an instance of U-SNP. Then, in an optimal solution of $\mathcal{I}$, a vehicle only stops at the endpoint stations of the clients assigned to it.

Proof. Consider a solution where some vehicle stops at a station without picking up nor delivering any client. The solution obtained by removing this stop is feasible and strictly better than the former one. ■

2.2.3 The intersection case

An interesting particular case, hereafter denoted Intersection U-SNP, arises when there exists some station $v' \in V$ wherein all demands intersect, that is, $\Delta_E(v') = E$. In such case, each vehicle $i \in K$ can serve at most C demands, that is, each subset E_i must have at most C edges.

Notice that the structure of the associated graph $G_{\mathcal{I}}$ is also affected if $\mathcal{I} = (V, E, C)$ is an instance of Intersection U-SNP. Indeed, if this is the case, $G_{\mathcal{I}}$ is, by

definition, bipartite (see Property 2). The opposite, however, may not hold. That is, if the associated graph $G_{\mathcal{I}}$ is bipartite, $\mathcal{I}$ may not be an instance of Intersection U-SNP. Figure 2.6 provides an example where $\mathcal{I}$ is not an instance of Intersection U-SNP even if $G_{\mathcal{I}}$ is bipartite.

Property 2 - Bipartiteness of Intersection U-SNP
Given an instance $\mathcal{I} = (V, E, C)$ of Intersection U-SNP, $G_{\mathcal{I}}$ is a bipartite graph.

Proof. Recall that $\Delta_E(v) = \delta_{G_{\mathcal{I}}}(\{1, \ldots, v\})$. Since $\mathcal{I}$ is an instance of Intersection U-SNP, there exists some station $v' \in V$ for which $\Delta_E(v') = E$. Therefore, $\Delta_E(v') = \delta_{G_{\mathcal{I}}}(\{1, \ldots, v'\}) = E$, and hence $G_{\mathcal{I}}$ is bipartite. ■

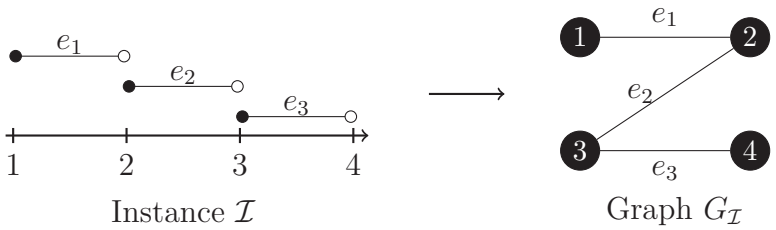


Figure 2.6: An example where $\mathcal{I}$ is not an instance of Intersection U-SNP but $G_{\mathcal{I}}$ is bipartite.

It follows directly from Property 2 that one can easily construct an instance of Intersection U-SNP from any bipartite graph $G = (U, V, E)$, by numbering the nodes (stations) in U from 1 to $|U|$ and the nodes in V from $|U| + 1$ to $|U| + |V|$.

To illustrate such particular case, consider again instance $\mathcal{I}$ depicted in Figure 2.4. By definition, $\mathcal{I}$ is not an instance of Intersection U-SNP. However, if $\mathcal{I}'$ is defined by removing e_5 from $\mathcal{I}$, then every demand intersects station 2 (*i.e.*, $\Delta_E(2) = E$), and hence $\mathcal{I}'$ is an instance of Intersection U-SNP.

2.3 State of the art

2.3.1 Transport related problems

The Stop Number Problem consists of transporting clients and/or goods from pickup points to delivery points, which characterizes the problem as a Vehicle Routing Problem with Pickups and Deliveries (VRPPD). In Parragh et al. [148, 147], VRPPD is shown to be composed of two major problem sub-classes. The first one concerns unpaired pickups and deliveries and is often referred to as Pickup and Delivery Travelling Salesman Problem (PDTSP) or Pickup and Delivery Vehicle Routing Problem (PDVRP). Such problems typically arises when the transported goods are identical

(*e.g.*, money) and hence no traceability is required. In other words, the goods delivered at some delivery point are allowed to come from any pickup location. The second subclass accounts for the transportation of paired pickups and deliveries, that is, each transport demand is associated with an origin and a destination. In such situation, the vehicle serving a demand must stop at both its origin and destination points. This subclass comprises the classical Pickup and Delivery Problem (PDP) and the Dial-A-Ride Problem (DARP), where PDP deals with the transportation of goods and DARP deals with people transportation. In practice, the difference between PDP and DARP is related to additional constraints and/or objectives taking into account the user convenience for improving the service quality. The SNP is clearly inserted in this latter subclass. Figure 2.7, adapted from Parragh et al. [148], provides a scheme depicting the relations and inheritances of such family of problems.

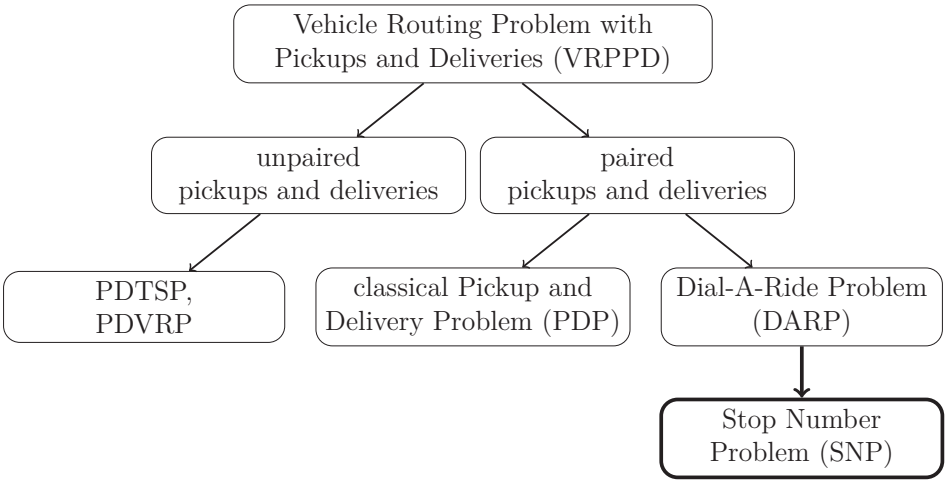


Figure 2.7: Pickup and delivery problems. Adapted scheme from Parragh et al. [148].

Each of the mentioned problem classes has received considerable attention in the literature. Since SNP can be viewed as a DARP, we focus on providing only its state of the art, instead of a global overview of transport problems. For a more extensive presentation of recent research on vehicle routing problems, the reader is referred to Braekers et al. [21], where 277 articles published between 2009 and 2015 are analyzed and classified according to their contributions to the field. Furthermore, a taxonomy for the literature devoted to variants of vehicle routing problems is provided in Lahyani et al. [114]. Further reviews of transport related scientific literature can be found in Berbeglia et al. [15], Toth and Vigo [174, 173], Bodin [17].

Dial-A-Ride Problems

DARPs are commonly defined within the following generic framework. Let n denote the number of requests to be served and $G = (N, A)$ be a complete directed graph where $N = P \cup D \cup \{s, t\}$, $P = \{p_1, \ldots, p_n\}$ and $D = \{d_1, \ldots, d_n\}$. Nodes s and t represent the initial and final depots, while subsets P and D refer to the sets of pickup and delivery locations, respectively. Hence, with each request $i \in \{1, \ldots, n\}$, an origin p_i and a destination d_i is associated. Let K denote the set of available vehicles, each with capacity Q_k, for $k \in K$. Finally, with each arc $(i, j) \in A$, a travel time t_{ij} and a routing cost c_{ij} are associated. Other features may be incorporated to this general framework, according to the application on which the problem is based on. The most common features include:

- A maximum trip duration T_k for each $k \in K$, is typically considered when driver's working hours must be taken into account or when fuel is a critical resource;

- If requests are not homogeneous, a load q_i may be associated to each request $i \in \{1, \ldots, n\}$. This means that the vehicle serving request i has to pickup q_i units at p_i and to, identically, drop-off q_i units at d_i. Moreover, a service duration h_i can also be associated to each request $i \in \{1, \ldots, n\}$, forcing the vehicle serving request i to hold still for h_i time units at p_i for the pickup service to be concluded.

- Usually when transporting people, the quality of service is a key factor. For handling the customers waiting time, a time window $[e_i, l_i]$ can be associated with each request $i \in \{1, \ldots, n\}$, where e_i and l_i represent the earliest and latest time at which a service should begin at p_i. Moreover, a maximum riding time L_i may be associated to each request $i \in \{1, \ldots, n\}$ to avoid users remaining in the vehicle for too long.

Thereby, DARPs consist of assigning requests and constructing routes (*i.e.*, deciding the order on which requests are picked up and delivered) for each vehicle in K such that (i.) each request is assigned to some vehicle; (ii.) the load of a vehicle never exceeds its capacity; (iii.) users and/or vehicle time restrictions are respected; and (iv.) the total route costs are minimized.

Most of the exact methods developed for treating DARPs found in the literature are based within a Branch-and-Cut (BC), Branch-and-Price (BP), or Branch-and-Price-and-Cut (BPC) framework.

Branch-and-Cut. The first attempt to optimally solve DARP through a BC algorithm is due to Cordeau [37], where a three-index MIP formulation for solving DARP is introduced. The families of valid inequalities explored are derived from well-known inequalities for the Vehicle Routing Problem and the Traveling Salesman Problem such as subtour elimination constraints, precedence constraints and order constraints. Instances with up to 32 requests were solved. In Ropke et al. [160] a tighter two-index formulation is presented and the valid inequalities from Cordeau [37] are adapted to fit such formulation. Moreover, two new families of valid inequalities are introduced based on the idea of incompatible user time windows. The same instances as in Cordeau [37] were solved. Experiments show that the both formulations (the two-indexed and the three-indexed) are competitive, even if the more compact one has a minor advantage.

In Parragh [146], a BC framework is proposed for dealing with a variant of standard DARP taking into account heterogeneous vehicles and users (H-DARP). A three-index formulation and a two-index formulation based on the works from Cordeau [37] and Ropke et al. [160] are proposed, and the valid inequalities proposed in such studies are adapted to fit H-DARP. Instances with up to 40 requests are solved to optimality and results show that the two-index formulation outperforms the three-index one. Also inspired by the works from Cordeau [37] and Ropke et al. [160], a BC algorithm for solving H-DARP with multiple depots (MD-H-DARP) is proposed in Braekers et al. [20].

Other BC algorithms have been developed in the literature for treating problem-specific features of the DARP. For instance, in Liu et al. [121] driver's lunch breaks are taken into account and hence a request cannot be served during this period of time. Moreover, due to a strict limit on the trip duration, vehicles are allowed (and forced) to make multiple trips per day. For treating these particularities, two families of valid inequalities are proposed. In Braekers and Kovacs [19], being assigned to a familiar driver is taken as an important aspect of service quality. For this reason, driver's consistency is considered in a multi-period DARP by limiting the maximum number of different drivers serving the same customer over a period of time. In Cortés et al. [42], a BC framework based on Benders Decomposition (Benders [14]) is proposed for solving a variant of DARP where passengers are allowed to switch vehicles on certain hub nodes.

A common picture in all such BC frameworks is the search for valid inequalities strengthening the formulation under study and hence improving its performances. However, it appears to exist a lack of theoretical research that is capable of truly evaluating the strength of such inequalities (*e.g.*, by providing the conditions under which such cuts are facet-defining). Such criticism is also shared by other authors

(*e.g.*, the literature review conducted by Ho et al. [91, p.403]). With this in mind, we take extra care on addressing this subject in Chapter 4 for the U-SNP.

Branch-and-Price.

Another common approach when dealing with DARP is to develop a BP framework. Just as BC algorithms, BPs are embedded in a Branch-and-Bound framework, but instead of generating cuts to reinforce the linear relaxation of a given formulation, BPs focus on column generation. Typically, BP requires a reformulation of the problem into a MIP having a better LP relaxation and a much larger number of variables. A subset of variables is then removed from such formulation giving rise to the so-called *restricted master problem*. Such action allows a faster resolution but, in return, it may prevent optimality since not all variables are considered. In order to handle this drawback, at each node of the Branch-and-Bound tree, a column generation procedure is applied by solving a *pricing problem*. The pricing problem either generates a variable capable of improving the relaxed solution or proves that such variable does not exists. If such variable is found, it is reintroduced to the formulation and the LP is reoptimized. Branchings take place when no improving variable can be found within the pricing problem and the candidate solution does not satisfies the integrality constraints.

Usually, when dealing with DARPs the pricing problem consists on generating feasible vehicle routes, while the master problem focus on finding a subset of the available routes that satisfies all requests and optimizes the problem's objective function (*e.g.*, minimize the total route cost). Such reformulation is commonly referred to as *Set Partitioning* formulation and is showed to provide better LP relaxation bounds (see Bramel and Simchi-Levi [22]). The first BP framework for solving DARPs based on such reformulation is due to Dumas et al. [55]. In Garaix et al. [73], a DARP taking into account vehicles occupancy rate is treated through a BP algorithm, where the pricing problem is solved via dynamic programming. In Feng et al. [67], constraint programming is applied for solving the pricing problem. A column generation approach is also applied in Parragh et al. [150] for solving a HDARP.

Branch-and-Price-and-Cut.

If additional cutting planes are introduced to the restricted master problem in order to reinforce its linear relaxation, the resulting algorithm is called Branch-and-Price-and-Cut. BPC algorithms combines both column generation and cutting planes in order to solve combinatorial problems. In Qu and Bard [154], a BPC framework is developed for the HDARP where subset-row inequalities (due to Jepsen et al. [97]) are dynamically introduced to the master problem, together with adapted

constraints derived from Cordeau [37], Ropke et al. [160], Ropke and Cordeau [159]. In Gschwind and Irnich [85], a BPC algorithm is proposed for solving a DARP where intra-route constraints (*i.e.*, constraints affecting single routes such as maximum trip duration, precedence and dynamic time windows) are considered. Recently, a BPC framework has been proposed in Luo et al. [123] for solving a multi-objective realistic DARP. In such version, requests are not forced to be served like in other typical DARP applications. Instead, the objective function is modified to first maximize the number of satisfied requests and then minimizing the total route costs. Moreover, some additional real-world constraints like driver's lunch breaks are also taken into account.

Heuristics and metaheuristics.

Due to the $\mathcal{NP}$-Hardness of DARPs, only small-sized instances can be optimally solved within reasonable time using exact methods. For this reason, most of scientific research that has been dedicated to DARP focus on the development of computational efficient algorithms yielding effective solutions, even if optimality is not guaranteed. Many different heuristics and metaheuristics have been developed with this goal.

Tabu Search (TS) follows the concept of local search heuristics in the sense that local changes are applied to a given solution in order to obtain a new one. Tabu search enhances the performance of standard local search by constructing a *tabu* list of solutions that should not be revisited. Moreover, non-improving changes are accepted in order to avoid getting trapped in local optima. One of the first TS algorithms proposed for the DARP is due to Cordeau and Laporte [39]. Such algorithm uses simple local search procedures such as reassigning a request from a vehicle to another. Moreover, in order to guide the exploration towards new solutions, frequent changes are penalized and infeasible solutions are temporarily accepted. Many other TS algorithms for dealing with additional constraints and real-life applications are adapted versions of such pioneer TS (see Paquette et al. [145], Ho et al. [90], Kirchler and Calvo [106], Beaudry et al. [13]).

A huge number of other metaheuristics for DARPs can be found in the literature. These include simulated annealing (*e.g.*, Braekers et al. [20], Zidi et al. [187], Reinhardt et al. [157]), genetic algorithms (*e.g.*, Jorgensen et al. [98], Masmoudi et al. [129]) and variable neighborhood search (*e.g.*, Parragh et al. [149]).

Usually heuristics are less effective than metaheuristics. However, when there is an urgent need for fast solutions, heuristics may be applied. Such situations arise typically when dealing with dynamic (online) DARPs or in order to provide a first feasible solution for a more complex method (see Braekers et al. [20]). In

Marković et al. [128] a real-world application requiring the resolution of a dynamic DARP uses a greedy insertion heuristic for obtaining solutions quickly. In Xiang et al. [185], a dynamic and stochastic DARP (*i.e.*, a dynamic DARP dealing with time-dependent stochastic events such as fluctuating travel times, new requests, cancellations, vehicle's mechanical failures, etc) takes advantage of a local search heuristic that is called each time a new event take place. A local search procedure capable of solving instances with up to 2000 requests is also used in Xiang et al. [184] to obtain fast solutions for a large-scale static DARP.

Further extensive and detailed surveys on DARP are provided in Ho et al. [91], Cordeau and Laporte [40], Parragh et al. [147], Cordeau and Laporte [38], Molenbruch et al. [134].

Stop Number Problem

Stop Number Problem is a recent problem that was introduced in Pimenta et al. [151] as a reliability oriented DARP, following the conference paper of Deleplanque et al. [51]. Indeed, SNP can be seen as a Dial-a-Ride Problem where the arc costs $c_{ij} = 0$ if nodes i and j correspond to the same station, and $c_{ij} = 1$ otherwise. Notice however that some features of the DARP are handled indirectly in SNP. For instance, time windows are handled through parameter H representing the maximum number of waiting tours. Moreover, since the circuit network and its stations are fixed and predefined, the routing phase is also much less complex than in a standard DARP. Indeed, once the assignment of demands to vehicles is done, the routing problem in SNP is reduced to choosing on which tour $h \in \{0, \ldots, H\}$ each demand is served. As a consequence, if $H = 0$, the routing problem is trivial.

Such considerations yield the question of whether or not SNP and U-SNP are as hard as DARPs. In Pimenta et al. [151], SNP was proved to be (weakly) NP-Hard through a simple reduction from the classic Partition Problem (see Garey and Johnson [74]). Indeed, if all requests have the same origin and destination stations, SNP is clearly optimized when the number of employed vehicles is minimum. Therefore, if one supposes

$$C = \frac{\sum_{i \in E} l_i}{2},$$

where l_e stands for the load of request i, then SNP fairly corresponds to solving a Partition Problem.

Notice however that such proof is fully based on the different request loads, and hence cannot be directly applied to U-SNP. The following theorem, due to Pimenta et al. [151], shows that U-SNP can be efficiently solved under some specific

conditions.

Theorem 2.1 - Pimenta et al. [151]
U-SNP is solvable in polynomial time if C and p are fixed constants through dynamic programming. ■

Nevertheless, the question of whether or not U-SNP is $\mathcal{NP}$-Hard remains unanswered and the following conjecture is also proposed in Pimenta et al. [151].

Conjecture 1 - Pimenta et al. [151]
U-SNP is $\mathcal{NP}$-Hard. ■

For solving SNP, an integer linear programming formulation is proposed in Pimenta et al. [151]. Such formulation is further detailed in Section 2.4. Moreover, a set partitioning reformulation yielding better bounds but containing an exponential number of variables is also proposed. The associated pricing problem is also shown to be $\mathcal{NP}$-Hard. It is reported that only small-sized instances could be solved to optimality. Finally, a Greedy Randomized Adaptive Search Procedure (GRASP) is proposed for heuristically solving large instances. In Deleplanque et al. [51], SNP is studied within a Branch-and-Price framework based on the same set partitioning reformulation.

2.3.2 Other related problems

It is interesting to remark that some network design problems arising from the telecommunications field are closely related to the Stop Number Problem. To the best of our knowledge, the connections between the problems mentioned in this subsection and the SNP were never studied before. It is therefore natural to transfer and adapt some of the results known in the literature for these problems to the object of our study: the Stop Number Problem. Such transfers will be explained in detail at pertinent and suitable points across this dissertation. Notice however, that the other way around also holds, and some of the results we obtained in the study of U-SNP can and will be extended to these related problems.

The Intra-Ring Synchronous Optical Network Design Problem

SONET is an American standard of optical fiber transmission technology standing for Synchronous Optical NETworks (also known in Europe as Synchronous Digital Hierarchy, or SDH for short). The Intra-Ring Synchronous Optical Network Design Problem is a network design problem arising from the deployment

of SONETs. In such problem, one is given a set of *telecommunication centers* linked in a circular fashion by a cable of fixed *capacity* optic fibers, called *rings*. Moreover, a set of expected *bandwidth requests*, or *demands*, between pairs of centers is given. For each ring, a collection of demands must be assigned such that the sum of the bandwidth requests served by the same ring fits its capacity. Within each ring, an electronic termination called Add-Drop Multiplexer (ADM) must be placed at each center incident to one of the demands assigned to it. Demands assigned to the same ring and incident to the same center may share the same ADM. Due to the expensive costs of ADMs, the objective is to minimize the number of installed ADMs. For further technical information on SONETs and ADMs the reader is referred to Wu [183].

Formally, the Intra-Ring Synchronous Optical Network Design Problem can be defined as follows. Consider a set of nodes N referring to the telecommunication centers. Let $w_{ij} \in \mathbb{R}^+$ denote the bandwidth requested between nodes $i \in N$ and $j \in N$ and define an edge set $E = \{ij : w_{ij} > 0\}$. The weight on edge $ij \in E$ is given by w_{ij}. Moreover, let M be the set of rings each with capacity k. The problem consists on partitioning E into $|M|$ subsets $E_1, \ldots, E_{|M|}$ that minimizes $\sum_{i=1}^{|M|} |V[E_i]|$, such that $\sum_{e \in E_i} w_e \leq k$, for each $i \in \{1, \ldots, |M|\}$. Notice that if $G = (N, E)$ is a bipartite graph, Intra-Ring Synchronous Optical Network Design Problem is equivalent to Intersection SNP with $H = 0$ and no parallel demands.

In Sutter et al. [171], the number of available rings is taken to be irrelevant as the cost of ADMs are significantly higher than *opening* an extra ring. The authors propose exact methods and heuristics for solving the problem. A MIP formulation (with great similarities with the one proposed in Pimenta et al. [151] for the SNP) is proposed. Nonetheless, the authors report that due to the lack of a good upper bound on the number of rings, a great deal of symmetry is detected slowing down the performance of such formulation. Moreover, the weakness of lower bounds is pointed out as a main reason for which the formulation fails on solving the problem. A Column-Generation approach is also proposed to provide a quality certificate to the results obtained heuristically. However, since the column generation subproblem is also hard to solve, instances with more than 15 nodes could not be solved to optimality.

In Sherali et al. [167], Lee et al. [117], a variant where the number of ADMs within the same ring must respect a given constant limit $R \in \mathbb{N}$ is investigated. The problem is shown to be $\mathcal{NP}$-Complete with a reduction from Partition Problem, using the same principle as in the proof of NP-Hardness given by Pimenta et al. [151] for the SNP. Valid inequalities reinforcing the formulation are introduced and used

to derive a Branch-and-Cut framework. Nonetheless, most of the valid inequalities presented require the presence of the ADM limit R. Furthermore, no facial study is conducted.

The k-Edge-Partitioning Problem

The k-Edge-Partitioning Problem appears in the literature as an *uniform* version of the Intra-Ring Synchronous Optical Network Design Problem. In such problem, all demands are said to request for the same bandwidth and the ring capacity is k times this common demand unit. Once again, the cost of using or *opening* a ring is considered insignificant, and thus one may use as many rings as desired.

The problem is formally defined as follows. Given a simple graph $G = (V, E)$, where V is the set of n centers and E is the set of m demands, and an integer $k \geq 1$, find a partition of E into R subsets $E_1, \ldots, E_R$ that minimizes $\sum_{i=1}^{R} |V[E_i]|$, such that $|E_i| \leq k$ for each $i \in \{1, \ldots, R\}$. Notice that if G is a bipartite graph, k-Edge-Partitioning Problem is equivalent to Intersection U-SNP with no parallel demands.

In Goldschmidt et al. [78], the problem is shown to be solvable in polynomial time for $k = 2$ and $\mathcal{NP}$-Complete for $k \geq 3$. Notice however, that the $\mathcal{NP}$-Completeness proof given uses a reduction from Edge-Partition into Triangles Problem. Since such problem is trivial in bipartite graphs, such proof cannot be immediately extended to show that Intersection U-SNP is also $\mathcal{NP}$-Complete. The authors also propose a linear-time $\mathcal{O}(\sqrt{k})$-approximation algorithm. Such algorithm relies on covering the edges of G with trees of size at least $\left\lceil \frac{k}{2} \right\rceil$.

In Brauner et al. [24], a linear-time $\mathcal{O}(\sqrt{k})$-approximation algorithm based on the search for an Eulerian path is proposed. Such algorithm guarantees a better approximation ratio than the one proposed in Goldschmidt et al. [78], for Eulerian graphs or if $k \leq \frac{m-1}{\sqrt{2m}-1}$. Moreover, a list of particular cases where the problem can be solved in polynomial time is given. More precisely, these cases are: i. G is a grid graph and $k = 3$; ii. G is a tree and $k = 3$; iii. G is a complete bipartite graph graph and $k = 3$; iv. G is the complete bipartite graph $K_{2,n}$ graph and k is even.

In Brauner and Lemaire [23], a $\ln \sqrt{\frac{k}{2}}$-approximation algorithm is proposed based on the set covering greedy algorithm due to Chvátal [32]. However, the proposed algorithm only runs in polynomial time if k is a fixed constant, which characterizes the algorithm as pseudo-polynomial.

The Path Traffic Grooming Problem

With the development of wavelength division multiplexing (WDM) technology, it became possible to split the eligible bandwidth of an optical fiber into several wavelength channels. The number of available wavelength channels is given by $W \in \mathbb{N}$. This means that, given a WDM network $G = (V, E)$, instead of having one SONET ADM placed on every wavelength at each node of the network, it is possible to have an ADM installed only for the wavelengths used at a given node. Moreover, with the deployment of commercial WDM systems, it has become evident that the dominant costs in building optical networks are based on the cost of its components, that is, the number of ADMs installed.

In such networks, traffic demands between nodes must be assigned to a wavelength channel and routed in the network through a path. In order to use bandwidth more efficiently, some optical networks allow multiple traffic demands to share the capacity of a wavelength channel. If each traffic request uses $1/g$ of the bandwidth of a wavelength, g is said to be the *grooming ratio* (or *grooming factor*). As a consequence, at most g traffic demands passing through a network edge $e \in E$ can be assigned to the same wavelength. Traffic Grooming Problem consists on assigning demands to wavelength channels as well as routing them through the network, in such a way that the wavelength channel's capacity is respected on every edge of the network and the number of ADMs required is minimized.

The concept of traffic grooming was introduced in Gerstel et al. [76]. After its appearance, Traffic Grooming has been widely studied within many different network topologies, notably rings (*i.e.*, G is a cycle). See Modiano [133], Dutta and Rouskas [56], Zhu and Mukherjee [186] for surveys. The case where G is a path, and therefore, the routing decision is trivial was first considered in Huang et al. [95]. However, the problem studied in such paper considers that a given traffic demand occupies $1/g$ of the bandwidth of a wavelength channel through the whole path G, which makes the problem equivalent to k-Edge Partitioning Problem described previously.

In Bermond et al. [16], a demand is said to only occupy the bandwidth over the path on which it is routed. We are more interested in this latter version named Path Traffic Grooming that can be formally defined as follows. Consider a path P_n with vertex set $V(P_n) = \{1, \ldots, n\}$ and edge set $E(P_n)$ composed of edges uv where $1 \leq u \leq n-1$ and $v = u+1$. Given path a P_n representing the network, a graph $H = (V(P_n), E)$ representing the set of requests and a grooming factor g, find a partition of E into W subsets $E_1, \ldots, E_W$, such that for each edge $uv \in E(P_n)$, the number of edges $ij \in E_w$ with $i \leq u < j$ is at most g, for each $w \in \{1, \ldots, W\}$.

The goal is to minimize $\sum_{w=1}^{W} |V[E_w]|$. Notice that Path Traffic Grooming is equivalent to U-SNP with no parallel demands.

In Bermond et al. [16], Path Traffic Grooming is shown to be solvable in polynomial time if the set of traffic requests have a uniform all-to-all structure (*i.e.*, H is a complete graph) and $g = 2$. Moreover, the case where $g = 1$, is solved in polynomial time for any set of traffic requests. In Amini et al. [3], the problem is shown to be $\mathcal{APX}$-Complete for any fixed value of $g \geq 2$ and an unbounded number of available wavelengths. In addition, an $\mathcal{O}(n^{\frac{1}{3}} \log^2 n)$-approximation algorithm is provided.

In Shalom et al. [165], the problem is shown to be $\mathcal{NP}$-Complete for any fixed $g \geq 2$ and *bounded* number of wavelengths through a reduction from k-Coloring of Circular Arc Graph. However, such result is contradictory with Theorem 2.1. We are convinced that the proof of $\mathcal{NP}$-Completeness of Path Traffic Grooming presented only works for an *unbounded* number of wavelengths since k-Coloring of Circular Arc Graph can be solved in polynomial time when the number of colors k is fixed (see Garey et al. [75]).

Network design problems involving the deployment of SONETs in WDM networks have been largely studied throughout the literature. Notice however that the aim of this subsection is not to give a complete state-of-the-art of such problems. Instead, we have tried to give a glimpse on the problems that show a particular strong relation with the problem under analysis in this thesis: the Stop Number Problem.

2.4 Problem formulation and associated polytope

A natural IP formulation for the U-SNP is introduced by Pimenta et al. [151]. Such formulation is based on two sets of binary variables $x \in \{0,1\}^{m \times p}$ and $y \in \{0,1\}^{n \times p}$. The variable x_e^i represents the fact that demand $e \in E$ is assigned or not to vehicle $i \in K$, that is,

$$x_e^i = \begin{cases} 1, & \text{if } e \in E_i \\ 0, & \text{otherwise.} \end{cases}$$

The variable y_v^i indicates whether or not vehicle $i \in K$ stops at station $v \in V$, that is,

$$y_v^i = \begin{cases} 1, & \text{if } v \in V[E_i] \\ 0, & \text{otherwise.} \end{cases}$$

Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP, such formulation is hereafter denoted by

$$\text{USNP}_{(V,E,C)} = \{x \in \mathbb{R}^{m\times p}, y \in \mathbb{R}^{n\times p} : (x, y) \text{ verifies } (2.2) - (2.6)\},$$

and is described below.

$$\min \sum_{v\in V} \sum_{i\in K} y_v^i \tag{2.1}$$

subject to

$$\sum_{i\in K} x_e^i = 1 \qquad \forall e \in E, \tag{2.2}$$

$$\sum_{e\in\Delta_E(v)} x_e^i \leq C \qquad \forall v \in V, i \in K, \tag{2.3}$$

$$x_e^i - y_v^i \leq 0 \qquad \forall i \in K, e \in E, v \in \{o_e, d_e\}, \tag{2.4}$$

$$x_e^i \in \{0, 1\} \qquad \forall e \in E, i \in K, \tag{2.5}$$

$$y_v^i \in \{0, 1\} \qquad \forall v \in V, i \in K. \tag{2.6}$$

The objective function (2.1) stands for the total number of stops the fleet of vehicles is supposed to make. The assignment constraints (2.2) ensure that each demand is assigned to exactly one vehicle. The capacity constraints (2.3) guarantee the vehicle's capacity is respected all along the circuit. Stop constraints (2.4) imposes that a vehicle must stop at the pick-up and drop-off stations of a demand assigned to it. Finally, constraints (2.5) and (2.6) settle the domains of variables x and y.

Notice however that, one is not obliged to verify if the capacity is satisfied *all along* the circuit. Instead, it may focus exclusively on verifying if capacity is respected on maximal-intersection stations. This means that for the Intersection U-SNP the following property holds.

Property 3 - Capacity constraints on Intersection U-SNP
If (V, E, C) is an instance of Intersection U-SNP, then capacity constraints (2.3) can be replaced by

$$\sum_{e\in E} x_e^i \leq C \qquad \forall \quad i \in K. \tag{2.7}$$

Given an instance (V, E, C) of U-SNP, let

$$\mathcal{P}_{(V,E,C)} = \text{conv}\left(\text{USNP}_{(V,E,C)}\right)$$

be the convex hull of the solutions of U-SNP. To the best of our knowledge, such polytope has not yet been the subject of substantial research. In this sense, Chapter 4 is dedicated to such study.

2.5 Concluding remarks and useful definitions

Before diving into the main results of this dissertation, some additional remarks and definitions are made necessary for an easier extensive comprehension of the contents to come.

Definition 2.1 - Idleness
*Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP and a feasible solution $\{E_1, \ldots, E_p\}$ of $\mathcal{I}$, vehicle $i \in K$ is said to be **idle** if no demand is assigned to vehicle i, that is, $E_i = \emptyset$. Analogously, for a solution $(\bar{x}, \bar{y}) \in \mathrm{USNP}_{(V,E,C)}$, vehicle $i \in K$ is said to be idle if $\bar{x}_e^i = 0$ for every $e \in E$.*

Definition 2.2 - Full charge
*Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP and a feasible solution $\{E_1, \ldots, E_p\}$ of $\mathcal{I}$, vehicle $i \in K$ is said to be **fully charged** at station $v \in V$ if exactly C demands are present in vehicle i at station v, that is, $|\Delta_{E_i}(v)| = C$. Analogously, for a solution $(\bar{x}, \bar{y}) \in \mathrm{USNP}_{(V,E,C)}$, vehicle $i \in K$ is said to be fully charged at station $v \in V$ if $\sum_{e \in \Delta_E(v)} \bar{x}_e^i = C$.*

Definition 2.3 - Minimum and optimal number of vehicles
Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP and a feasible solution $\pi = \{E_1, \ldots, E_p\}$ of $\mathcal{I}$, let p'_π denote the number of non-idle vehicles in solution π. When solution π is clear from the context, we may omit the subscript π. Moreover, let p_{min} denote the minimum number of vehicles necessary to serve all the demands in E. In other words, if $\Pi_\mathcal{I}$ represents the set of all feasible solutions of $\mathcal{I}$, then

$$p_{min} = \min_{\pi \in \Pi_\mathcal{I}} \{p'_\pi\}.$$

Analogously, let p_{opt} denote the minimum number of non-idle vehicles among the optimal solutions of $\mathcal{I}$. In other words, if $\Pi^*_\mathcal{I}$ represents the set of optimal solutions of $\mathcal{I}$, then

$$p_{opt} = \min_{\pi \in \Pi^*_\mathcal{I}} \{p'_\pi\}.$$

Definition 2.4 - Merge
Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP, a feasible solution $\{E_1, \ldots, E_p\}$ of $\mathcal{I}$, and a pair of vehicles i and j, the solution $\{\bar{E}_1, \ldots, \bar{E}_p\}$ is said to be obtained from $\{E_1, \ldots, E_p\}$ by merging vehicles i and j if

$$\begin{aligned} &\bar{E}_i = E_i \cup E_j, \\ &\bar{E}_j = \emptyset, \\ &\bar{E}_k = E_k \quad \forall k \in K \setminus \{i, j\}. \end{aligned}$$

Analogously, given a solution $(x, y) \in \mathrm{USNP}_{(V,E,C)}$ *, the solution* $(\bar{x}, \bar{y})$ *is said to be obtained from* (x, y) *by merging vehicles* i *and* j *if*

$$\begin{aligned}
&\bar{x}_e^i = x_e^i + x_e^j && \forall e \in E,\\
&\bar{x}_e^j = 0 && \forall e \in E,\\
&\bar{x}_e^k = x_e^k && \forall e \in E, k \in K \setminus \{i, j\},\\
&\bar{y}_v^i = \max\{y_v^i, y_v^j\} && \forall v \in V,\\
&\bar{y}_v^j = 0 && \forall v \in V,\\
&\bar{y}_v^k = y_v^k && \forall v \in V, k \in K \setminus \{i, j\}.
\end{aligned}$$

Notice that merging two vehicles might result in a unfeasible solution. Indeed, the resulting solution $\{\bar{E}_1, \ldots, \bar{E}_p\}$ is only feasible if $|\Delta_{E_i \cup E_j}(v)| \leq C$ for every $v \in V$. Nonetheless, throughout this thesis we are cautious enough to invoke such operation only on suitable situations.

Chapter 3

Properties and complexity

> "Simple things should be simple, complex things should be possible."
>
> — Alan Kay

In this chapter, the complexity of U-SNP is deeply investigated. Firstly, a series of properties and lower bounds for the U-SNP are presented. Based on such results, some particular cases are shown to be solvable in polynomial time. Next, a collection of important observations are made in order to finally derive a proof of NP-Hardness for the U-SNP.

3.1 Properties

For U-SNP, a vehicle does not have to stop in every station of the circuit. Instead, a vehicle must stop at the endpoint stations of all the clients assigned to it. Such observation yields the following remark that must be made explicit to the reader even if it might seem trivial.

Proposition 3.1 - Trivial lower bound
Given an instance (V, E, C) of U-SNP, a trivial lower bound on the total number of stops is the number of stations n.

Proof. Each station $v \in V$ has at least one demand incident to it. Therefore, at least one vehicle must stop at v. If exactly one vehicle stops at each of the n stations, then one has a solution with n stops. ■

Notice however that such lower bound is only met in extremely particular cases. As a matter of fact, this can only be achieved if $\Delta_E(v)$ has at most C edges, for

each $v \in V$. Moreover, if this is the case, a single vehicle can handle all demands and U-SNP can be trivially solved. By taking into account the structure of the graph $G = (V, E)$, one may try to derive better quality lower bounds than the one described in Proposition 3.1.

Proposition 3.2 - Lower bound based on degrees
Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP, a lower bound on the total number of stops is given by

$$\sum_{v \in V} \left\lceil \frac{\max\{|\delta^-_{G_\mathcal{I}}(v)|, |\delta^+_{G_\mathcal{I}}(v)|\}}{C} \right\rceil . \tag{3.1}$$

Proof. A single vehicle can take at most C demands among $\delta^-_{G_\mathcal{I}}(v)$ or $\delta^+_{G_\mathcal{I}}(v)$. Since all demands must be served by some vehicle, each station $v \in V$ must be visited by at least

$$\left\lceil \frac{\max\{|\delta^-_{G_\mathcal{I}}(v)|, |\delta^+_{G_\mathcal{I}}(v)|\}}{C} \right\rceil$$

different vehicles. ■

The next lower bound proposed here requires a further investigation on the minimum number of vehicles needed to serve the m demands in E (*i.e.*, p_{min}), for a given instance (V, E, C) of U-SNP. A trivial lower bound for p_{min} is

$$p_{min} \geq \max_{v \in V} \left\{ \left\lceil \frac{|\Delta_E(v)|}{C} \right\rceil \right\},$$

for otherwise, the fleet is not capable of serving all demands in $\Delta_E(v') \subseteq E$, where

$$v' = \arg\max_{v \in V} \{|\Delta_E(v)|\} .$$

On the other hand, since the demands can be viewed as (half closed - half open) intervals of the real line, a solution using exactly $\max_{v \in V} \left\{ \left\lceil \frac{|\Delta_E(v)|}{C} \right\rceil \right\}$ vehicles, clearly can be computed in polynomial time by a first-fit type algorithm with demands picked out in order of increasing pick-up station. The above results lead to the following proposition.

Proposition 3.3 - The value of p_{min}
Given an instance (V, E, C) of U-SNP,

$$p_{min} = \max_{v \in V} \left\{ \left\lceil \frac{|\Delta_E(v)|}{C} \right\rceil \right\}.$$

Notice that if (V, E, C) is an instance of Intersection U-SNP, $\max_{v \in V} \{|\Delta_E(v)|\} =$

m and therefore,

$$p_{min} = \left\lceil \frac{m}{C} \right\rceil . \qquad \blacksquare$$

Property 4 - Idle vehicles
Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP, with $C \geq 2$, there always exists a solution $\{E_1, \ldots, E_p\}$ of $\mathcal{I}$ where at least one vehicle is idle. That is, there exists some $i \in K$ for which $E_i = \emptyset$.

Proof. Trivial, since $p = m \geq p_{min} + 1$. $\blacksquare$

In graph theory, the girth of a graph G corresponds to the length of the smallest cycle contained in G. In other words, a graph is said to have girth k if it contains no cycle of size smaller than k. Such notion can be explored in order to derive new lower bounds.

Property 5 - Big cycles produce forests
Given an instance (V, E, C) of Intersection U-SNP, such that graph $G = (V, E)$ has girth greater than C, then in any feasible solution, $G(E_i)$ is a forest, for $i \in K$.

Proof. $G(E_i)$ cannot contain any cycle since the smallest cycle in G would violate the vehicle's capacity. Therefore, $G(E_i)$ is a forest. $\blacksquare$

Proposition 3.4 - Lower bound based on girth
Given an instance (V, E, C) of Intersection U-SNP, such that graph $G = (V, E)$ has girth greater than C, then a lower bound on the total number of stops is given by

$$m + \left\lceil \frac{m}{C} \right\rceil .$$

Proof. From Property 5, one knows that in any feasible solution, $G(E_i)$ is a forest, for any $i \in K$. Recall that the cost of a solution is evaluated as

$$\sum_{i \in K} |V[E_i]|.$$

Let p' denote the number of vehicles used in such feasible solution. Without loss of generality, assume $E_i \neq \emptyset$ for $i \leq p'$. By definition, a tree with m edges has $m+1$ vertices. It follows that $|V[E_i]| = |E_i| + t_i$, where $t_i \geq 1$ is the number of trees in the forest $G(E_i)$, for $i \leq p'$. Hence,

$$\sum_{i \in K} |V[E_i]| = \sum_{i=1}^{p'} (|E_i| + t_i) \geq \sum_{i=1}^{p'} (|E_i| + 1) = m + p'.$$

Finally, since $p' \geq p_{min}$,

$$\sum_{i \in K} |V[E_i]| \geq m + \left\lceil \frac{m}{C} \right\rceil . \qquad \blacksquare$$

In Pimenta et al. [151], it is pointed out – empirically – that for any given an instance (V, E, C) of U-SNP, $p_{opt} = p_{min}$. Next, we show that such statement holds if $C = 2$ and (V, E, C) is an instance of Intersection U-SNP. However, for $C \geq 3$, such statement is false. We prove it by showing that even when $G = (V, E)$ is a tree, p_{opt} can be strictly greater than p_{min}.

Proposition 3.5 - On Intersection U-SNP with $C = 2$, $p_{opt} = p_{min}$
Given an instance $(V, E, 2)$ of Intersection U-SNP,

$$p_{opt} = p_{min}.$$

Proof. If, in an optimal solution, 2 or more vehicles are assigned less than 2 clients, then one can merge pairs of such vehicles in order to construct a solution where at most one vehicle is assigned one single client. Such constructed solution induces at most the same number of stops as the former solution. Therefore, the constructed solution is also optimal. Then,

$$p_{opt} = \left\lceil \frac{m}{2} \right\rceil = p_{min}. \qquad \blacksquare$$

Proposition 3.6 - On Intersection U-SNP with $C = 3$, $p_{opt} \neq p_{min}$
On Intersection U-SNP, p_{opt} might be strictly greater than p_{min}.

Proof. The proof is done through a counter-example. Consider the instance of Intersection U-SNP depicted in Figure 3.1 with $C = 3$.

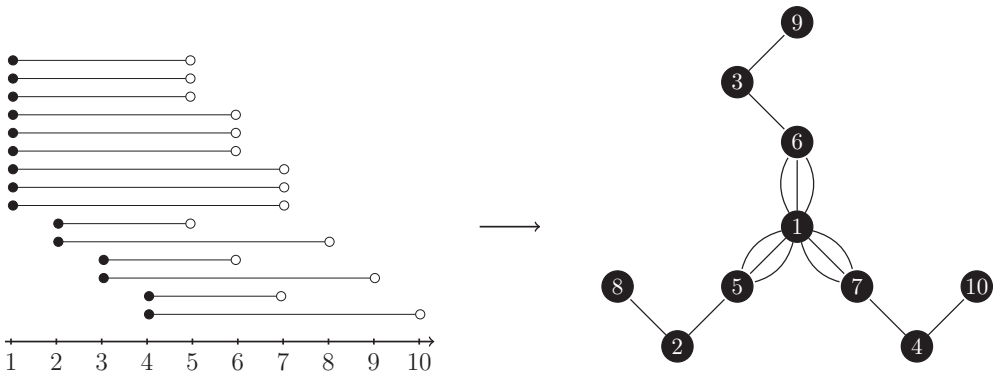


Figure 3.1: Counter-example instance used for proving $p_{min} \neq p_{opt}$.

The instance depicted here is composed of 3 triples of *parallel demands*, all with same origin, and a path of length 2 attached to each of their destinations. A feasible

solution with 15 stops is obtained by assigning each triple of *parallel demands* and each path of length 2 to a different vehicle. Moreover, this is an optimal solution since Proposition 3.2 ensures that

$$\sum_{v \in V} \left\lceil \frac{\max\{|\delta_G^-(v)|, |\delta_G^+(v)|\}}{C} \right\rceil = 15$$

is a lower bound on the number of stops. However, this solution uses 6 vehicles, while

$$p_{min} = \left\lceil \frac{m}{C} \right\rceil = \left\lceil \frac{15}{3} \right\rceil = 5.$$

We show next that one cannot find an optimal solution using exactly p_{min} vehicles. In order to obtain exactly 15 stops, each each station $v \in V$ must be visited by exactly $\left\lceil \frac{\max\{|\delta_G^-(v)|, |\delta_G^+(v)|\}}{C} \right\rceil$ vehicles. Hence, stations $2, 3$ and 4 must be visited by exactly 1 vehicle, which forces the paths of length 2 to be each on a different vehicle. Similarly, stations $5, 6$ and 7 must be visited by exactly 2 vehicles, and station 1 by exactly 3 vehicles. Additionally, using only 5 vehicles imposes each vehicle to carry exactly 3 demands. Thus each vehicle that takes a path of length 2 must also take one of the *parallel demands* incident to it, otherwise one of stations $5, 6$ or 7 must be visited by 3 vehicles. With this in mind, station 1 is already visited by 3 vehicles. However, some of the *parallel demands* have not been assigned yet, forcing station 1 to be visited by more than 3 vehicles, which prevents optimality. ■

To go even further, consider the tree $T = (V, E)$ represented in Figure 3.2 and let $\mathcal{I} = (V, E, 3)$ be an instance of Intersection U-SNP.

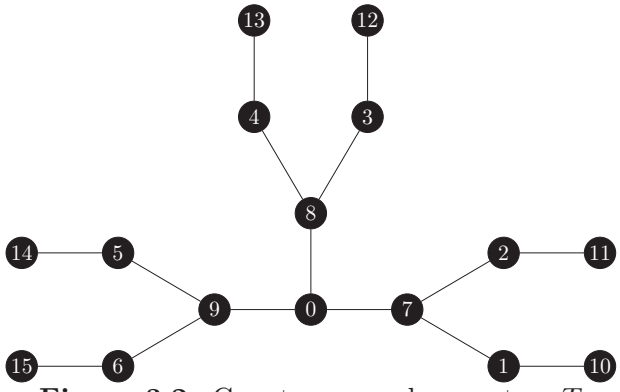


Figure 3.2: Counter-example on a tree T

Instance $\mathcal{I}$ is composed of 15 demands, which means that $p_{min} = 5$. Solving such instance with only 5 vehicles available yields an optimal solution stopping 22 times. On the other hand, if 6 vehicles are provided, an optimal solution with 21 stops can be achieved[1]. This shows that given an instance (V, E, C) of U-SNP, p_{opt} can be

[1]Both optimal solutions mentioned were obtained by solving the MIP formulation proposed by

strictly larger than p_{min} even if $G = (V, E)$ is a tree.

Another intuition one may have when trying to solve U-SNP is to keep *parallel demands* together. Indeed, for the Intersection U-SNP with $C = 2$, this policy does not prevent optimality as indicated by Proposition 3.7.

Proposition 3.7 - Parallel demands on Intersection U-SNP
Given an instance $\mathcal{I} = (V, E, 2)$ of Intersection U-SNP such that E has two parallel demands e and e', there exists an optimal solution where e and e' are assigned to the same vehicle.

Proof. Assume on the contrary that e and e' are served by different vehicles in every optimal solution. Let $\{E_1^*, \ldots, E_p^*\}$ be an optimal solution such that, *w.l.o.g.*, $e \in E_1^*$ and $e' \in E_2^*$. Trivially, $|E_1^*| = 2$, for otherwise $\{E_1^* \cup e', E_2^* \setminus e', E_3^*, \ldots, E_p^*\}$ would be a feasible solution with no more stops than $\{E_1^*, \ldots, E_p^*\}$. Similarly, $|E_2^*| = 2$. Let f denote the *other* demand served by vehicle 1, that is, $f = E_1^* \setminus e$. Likewise, let $f' = E_2^* \setminus e'$. Consider now the feasible solution $\{\{e, e'\}, \{f, f'\}, E_3^*, \ldots, E_p^*\}$ and let $\alpha = |V[\{f, f'\}]|$. Notice that $2 \leq \alpha \leq 4$. Then, one has

$$c(\{\{e, e'\}, \{f, f'\}, E_3^*, \ldots, E_p^*\}) = c(\{E_1^*, \ldots, E_p^*\}) - (|V[E_1^*]| + |V[E_2^*]|) + 2 + \alpha. \quad (3.2)$$

If e and f (or e' and f') are parallel demands, then $|V[E_1^*]| + |V[E_2^*]| = 2 + \alpha$ and by (3.2), $\{\{e, e'\}, \{f, f'\}, E_3^*, \ldots, E_p^*\}$ is optimal. On the other hand, if none of the sets E_1^* and E_2^* are composed of parallel demands, one has $|V[E_1^*]| + |V[E_2^*]| \geq 6 \geq 2 + \alpha$ since $\alpha \leq 4$, and once again $\{\{e, e'\}, \{f, f'\}, E_3^*, \ldots, E_p^*\}$ is optimal. Therefore, there always exists an optimal solution where the parallel demands e and e' are served by the same vehicle. ■

Nonetheless, such strategy might be deceiving for more general cases. Proposition 3.8 provides evidence that adopting this strategy may prevent optimality.

Proposition 3.8 - Parallel demands on U-SNP
Given an instance $\mathcal{I} = (V, E, 2)$ of U-SNP such that E has two parallel demands e and e', there might exist no optimal solution where e and e' are assigned to the same vehicle.

Proof. The proof is done through a counter-example. Consider the instance of U-SNP described in Figure 3.3 with $C = 2$.

A feasible solution with six stops can be obtained by assigning demands a, c and e to vehicle 1 and the others to vehicle 2. Notice that, in this solution the

Pimenta et al. [151]

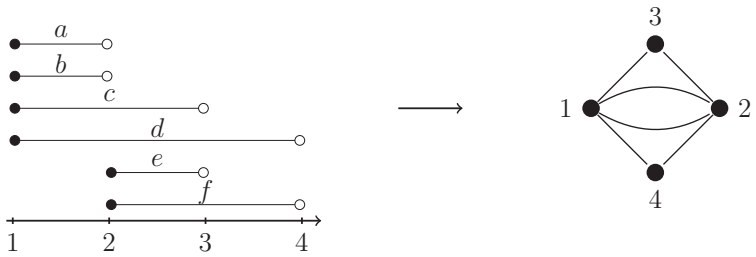


Figure 3.3: Counter-example instance used for proving *parallel demands* might have to be separated

parallel demands a and b are not served by the same vehicle. Next, we show that if one forces these demands to be served by the same vehicle, then one cannot find a solution with less than seven stops. In other words, no optimal solution wherein a and b are served by the same vehicle can be found.

For this, let $\{E_1, \ldots, E_p\}$ be a feasible solution such that, *w.l.o.g.*, $a \in E_1$ and $b \in E_1$. Demands c and d must be served by other vehicles (*i.e.*, $c \notin E_1$ and $d \notin E_1$), for otherwise the capacity of vehicle 1 is violated.

Suppose c and d are served by different vehicles. *W.l.o.g*, assume $c \in E_2$ and $d \in E_3$. Hence, the assignment of demands a, b, c and d already induces six stops (the vehicles 1,2, and 3 have two stops each) and assigning e to any vehicle increases this value.

On the other hand, if c and d are served by the same vehicle (*w.l.o.g.*, say $c \in E_2$ and $d \in E_2$), then the assignment of demands a, b, c and d already induces five stops (vehicle 1 has two stops and vehicle 2 has three stops). Notice that demands e and f cannot be served by vehicle 2 (*i.e.*, $e \notin E_2$ and $f \notin E_2$), for otherwise its capacity is violated. Moreover, none of the remaining demands e and f can be served by other vehicle than vehicle 1 (*i.e.*, $e \in E_1$ and $f \in E_1$), for otherwise a third vehicle would be required, and at least two more stops would be needed. However, the constructed solution $\{\{a, b, e, f\}, \{c, d\}, \emptyset, \ldots, \emptyset\}$ requires at least seven stops. Therefore, no solution with six stops wherein a and b are assigned to the same vehicle can be found. ■

The results presented in this section offer an insight on how difficult it is to characterize the optimal solutions of U-SNP. At the same time, for some specific cases, it was possible to show that by adopting some simple intuitions, optimality remains preserved. The next section takes profit of such results to derive polynomial time algorithms that are capable of solving certain particular cases of U-SNP.

3.2 Polynomial Cases

When instance $\mathcal{I} = (V, E, C)$ meets some particular conditions, U-SNP can be solved in polynomial time. Some trivial cases arise when $C = 1$ and (V, E, C) is an instance of Intersection U-SNP, or when $C \geq \max_{v \in V}\{|\Delta_E(v)|\}$. In the former case, each vehicle is constrained to serve exactly one demand, and therefore no solution with less than $2m$ stops is achievable. In the latter case, all demands fit into one single vehicle since $p_{min} = 1$, and this solution has n stops, which meets the lower bound provided by Proposition 3.1. Next, a series of less trivial cases are shown to be solvable in polynomial time.

Proposition 3.9 - Unit Capacity
Let $\mathcal{I} = (V, E, 1)$ be an instance of U-SNP. Then, U-SNP can be solved in $\mathcal{O}(m)$ time and the optimal solution has cost

$$\sum_{v \in V} \max\left\{|\delta^-_{G_\mathcal{I}}(v)|, |\delta^+_{G_\mathcal{I}}(v)|\right\}.$$

Proof. Notice that since $C = 1$, parallel demands cannot be served by the same vehicle. In Bermond et al. [16], Path Traffic Grooming is shown to be solvable in polynomial time for grooming factor $g = 1$. Since Path Traffic Grooming is equivalent to U-SNP with no parallel demands, we may use such result to derive a polynomial time algorithm capable of solving U-SNP for $C = 1$. Consider the following greedy algorithm that, given an instance $(V, E, 1)$ of U-SNP, sequentially:

i. selects a maximal sequence $(e_1, \ldots, e_q), q \geq 1$, of demands with $d_{e_i} = o_{e_{i+1}}$ for $i = 1, \ldots, q-1$,

ii. assigns the demands of such sequence to an available vehicle, and

iii. removes the q demands of the sequence from E.

Such algorithm may run in $\mathcal{O}(m)$ time and yields a solution with

$$\sum_{v \in V} \max\left\{|\delta^-_{G_\mathcal{I}}(v)|, |\delta^+_{G_\mathcal{I}}(v)|\right\}$$

stops, which meets the lower bound provided by Proposition 3.2. ■

Proposition 3.10 - Complete graphs and $C = 2$
Let $(V, E, 2)$ be an instance of U-SNP such that $G = (V, E)$ is a complete graph K_n. Then U-SNP can be solved in polynomial time.

Proof. In Bermond et al. [16], Path Traffic Grooming is shown to be solvable in polynomial time when the request graph is a complete graph K_n and the grooming factor $g = 2$. Since Path Traffic Grooming is equivalent to U-SNP with no parallel demands, such result also applies for U-SNP. ■

Proposition 3.11 - Stars
Let (V, E, C) be an instance of U-SNP such that $G = (V, E)$ is a star. Then U-SNP can be solved in polynomial time and the optimal solution has cost

$$\sum_{v \in V} \left\lceil \frac{\max\{|\delta_G^-(v)|, |\delta_G^+(v)|\}}{C} \right\rceil .$$

Proof. Construct an optimal solution of $(V, E, 1)$ using the greedy algorithm described in Proposition 3.9. Using the output solution, construct a solution of (V, E, C) by merging groups of C vehicles. The resulting solution stops

$$\sum_{v \in V} \left\lceil \frac{\max\{|\delta_G^-(v)|, |\delta_G^+(v)|\}}{C} \right\rceil$$

times, which meets the lower bound provided by Proposition 3.2. ■

Proposition 3.12 - Paths and cycles for the intersection case
Let (V, E, C) be an instance of Intersection U-SNP such that $G = (V, E)$ is a path (or a cycle). Then U-SNP can be solved in polynomial time and the optimal solution has cost

$$m + \left\lceil \frac{m}{C} \right\rceil .$$

Proof. The proof is done for the case where G is a path, but it also holds if G is a cycle. Let $(e_1, \ldots, e_m)$ be the sequence in which the demands appear on path G (see Figure 3.4).

Figure 3.4: Sequence of demands in G

Consider the greedy algorithm that assigns each set of demands

$$\left\{ e_{iC+1}, \ldots, e_{\max\{m, iC+C\}} \right\}$$

to a different vehicle, for $0 \leq i \leq \left\lceil \frac{m}{C} \right\rceil - 1$. Such algorithm yields a solution with $\left\lceil \frac{m}{C} \right\rceil - 1$ fully charged vehicles, each stopping $C + 1$ times and one vehicle stopping

$$\left[m - \left(\left\lceil \frac{m}{C} \right\rceil - 1 \right) C \right] + 1$$

times. Therefore, the solution obtained has

$$m + \left\lceil \frac{m}{C} \right\rceil$$

stops, which meets the lower bound provided by Proposition 3.4. ■

Proposition 3.13 - Trees, grids and complete bipartite graphs for the intersection case with $C = 3$
Let $(V, E, 3)$ be an instance of Intersection U-SNP such that $G = (V, E)$ is either a tree, a grid or a complete bipartite graph. Then U-SNP can be solved in polynomial time.

Proof. In Lemaire [118], k-Edge-Partitioning Problem is shown to be solvable in polynomial time if G is either a tree, a grid or a complete bipartite graph and $k = 3$. Recall that if G is a bipartite graph, then k-Edge-Partitioning Problem is equivalent to Intersection U-SNP with no parallel demands. Since trees, grids and complete bipartite graphs are all examples of bipartite graphs, such result also applies for U-SNP. ■

So far, the particular cases presented in this section did not have to deal with the presence of parallel demands. For this reason, many results known in the literature for some related problems could be easily extended to the U-SNP. Next, we show that any instance (parallel demands allowed) of the Intersection U-SNP with $C = 2$ can be solved in polynomial time.

Proposition 3.14 - Intersection case with $C = 2$
Intersection U-SNP can be solved in $\mathcal{O}(m)$ time when $C = 2$.

Proof. Consider an instance $\mathcal{I} = (V, E, 2)$ of Intersection U-SNP. By definition, each vehicle can serve at most two demands. We define a partition $\{\hat{E}_1, \ldots, \hat{E}_p\}$ of E as follows. Let each subset $\hat{E}_i$ be composed of two parallel demands for $0 \leq i \leq q$, such that the set of remaining demands

$$\tilde{E} = E \setminus \left\{ \bigcup_{i=0}^{q} \hat{E}_i \right\}$$

does not contain any pair of parallel demands. Notice that this step can be easily done in $\mathcal{O}(m)$ time through a simple greedy algorithm. Moreover, since Proposition 3.7 holds, $\{\hat{E}_1, \ldots, \hat{E}_p\}$ is an optimal solution of $\mathcal{I}$ if and only if $\{\hat{E}^*_{q+1}, \ldots, \hat{E}^*_p\}$ is an optimal solution of $\tilde{\mathcal{I}} = (V, \tilde{E}, 2)$.

For $k = 2$, it has been shown in Goldschmidt et al. [78] that k-Edge Partitioning Problem reduces to the Maximum 2-Chain Packing problem, which

can be solved in $O(m)$ time through an algorithm due to Masuyama and Ibaraki [130]. It follows that such algorithm can be applied to solve Intersection U-SNP for $C = 2$, and therefore, $\{\hat{E}^*_{q+1}, \ldots, \hat{E}^*_p\}$ can be obtained in $O(m)$ time. ■

3.3 $\mathcal{NP}$-Hardness

After showing that Intersection U-SNP can be solved in polynomial time for $C = 2$, the question of whether or not the general U-SNP is $\mathcal{NP}$-Hard for fixed values of $C \geq 2$ arises naturally. In Pimenta et al. [151], SNP is proved to be weakly $\mathcal{NP}$-Hard through a simple reduction from Partition Problem. Indeed, if the demands are not said to be identical in terms of load, it is easy to see that some kind of Knapsack or Bin-Packing problems are hidden behind SNP. On the other hand, when dealing with the unitary variant of SNP (*i.e.,* U-SNP), such problems become straightforward and the reduction proposed for SNP cannot be used. Nevertheless, the authors observe that, in practice, even the unitary variant of SNP remains difficult to solve through standard exact methods (*e.g.*, Branch-and-Bound and Branch-and-Cut). Such observation together with the struggle to come up with a tractable algorithm capable of solving the problem, pushed the authors to conjecture U-SNP to be $\mathcal{NP}$-Hard.

Notice that the simple statement of equivalence between U-SNP and Path Traffic Grooming made in Section 2.3 suffices to positively answer such conjecture. Nonetheless, our study on the complexity of U-SNP goes deeper, yielding stronger complexity results for U-SNP (that can obviously be extended for the related problems described in Section 2.3). More specifically, in this section, U-SNP is shown to be $\mathcal{NP}$-Hard for any fixed capacity $C \geq 2$ even when restricted to the case where the associated graph G is a planar bipartite graph. The proofs presented next were inspired from the works of Dyer and Frieze [57].

The General Case

In Amini et al. [3], Path Traffic Grooming Problem is shown to be $\mathcal{APX}$-Hard (and thus, also $\mathcal{NP}$-Hard), for any fixed grooming factor $g \geq 2$, through a reduction from the $\mathcal{APX}$-Complete problem of finding the maximum number of edge-disjoint triangles (*i.e., cycles of length 3*) in a tripartite graph. Recall that Path Traffic Grooming is equivalent to U-SNP with no parallel demands. Therefore, any result on the complexity of Path Traffic Grooming can be applied to U-SNP, by restriction.

Notice however, that if G is said to be bipartite, the problem of finding the maximum number of edge-disjoint triangles in G is trivial since, by definition, bipartite graphs do not contain any triangle. The proof proposed in Amini et al. [3] is therefore no longer valid, or at least, cannot directly show the $\mathcal{NP}$-Hardness of Path Traffic Grooming (and thus, of U-SNP) on bipartite graphs. Next, we propose a polynomial reduction from 3-Dimensional Matching to prove U-SNP is indeed $\mathcal{NP}$-Hard even when G is restricted to be a planar bipartite graph. For this, let us first recall the 3-Dimensional Matching Problem (3DM).

3-Dimensional Matching Problem – Garey and Johnson [74, p. 221]
Input: Three disjoint sets X, Y, and Z with equal cardinality q, and a set of triples $T \subseteq X \times Y \times Z$.
Output: Decide whether or not there is a subset $M \subseteq T$ such that $|M| = q$ and no two elements of M agree in any coordinate.

It is well known that 3DM problem is $\mathcal{NP}$-Complete (see Karp [103]). In Dyer and Frieze [58], 3DM is shown to be NP-Complete even when the associated bipartite graph $H = (T, S, E')$ is restricted to be planar, where $S = X \cup Y \cup Z$ and

$$E' = \bigcup_{t=(x,y,z)\in T} \{(t,x),(t,y),(t,z)\}.$$

When such restriction is applied, the problem is renamed as Planar 3DM. Figure 3.5 illustrates graph $H = (T, S, E')$.

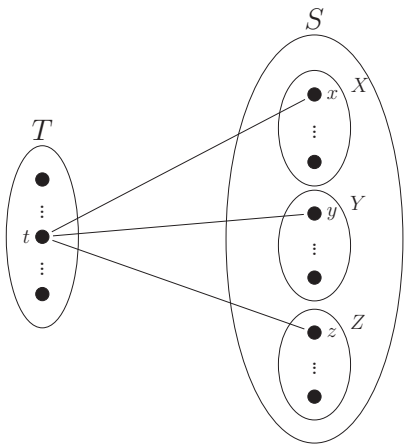


Figure 3.5: 3DM bipartite graph $H = (T, S, E')$

Theorem 3.1 - $\mathcal{NP}$-Hardness of U-SNP for $C = 2$
U-SNP is $\mathcal{NP}$-Hard even when restricted to the case where $C = 2$ and G is a planar bipartite graph.

Proof. Given an instance of Planar 3DM with disjoint sets X, Y, Z of cardinality q and a set of triples T, let $H = (T, S, E')$ be the planar bipartite graph described in Figure 3.5 where $S = X \cup Y \cup Z$ and

$$E' = \bigcup_{t=(x,y,z)\in T} \{(t,x),(t,y),(t,z)\}.$$

Next, we construct an instance $\mathcal{I}$ of U-SNP in polynomial time and claim that the 3DM instance has a matching if and only if $\mathcal{I}$ has a solution with a certain amount of stops. For this, let us construct graph $G = (V, E)$ in the following manner.

Start with an empty graph G. For each element $i \in W = Y \cup Z$, add a vertex w_i. For each element $x \in X$, add a vertex x. For each triple $t = (x, y, z) \in T$, add a vertex x'_t and let X' denote this set of vertices. Finally, for each triple $t = (x, y, z) \in T$ add a vertex t and edges (x'_t, t), (x'_t, x), (t, w_y), and (t, w_z). This construction yields the graph depicted in Figure 3.6 with $2|T| + |S|$ vertices and $4|T|$ edges.

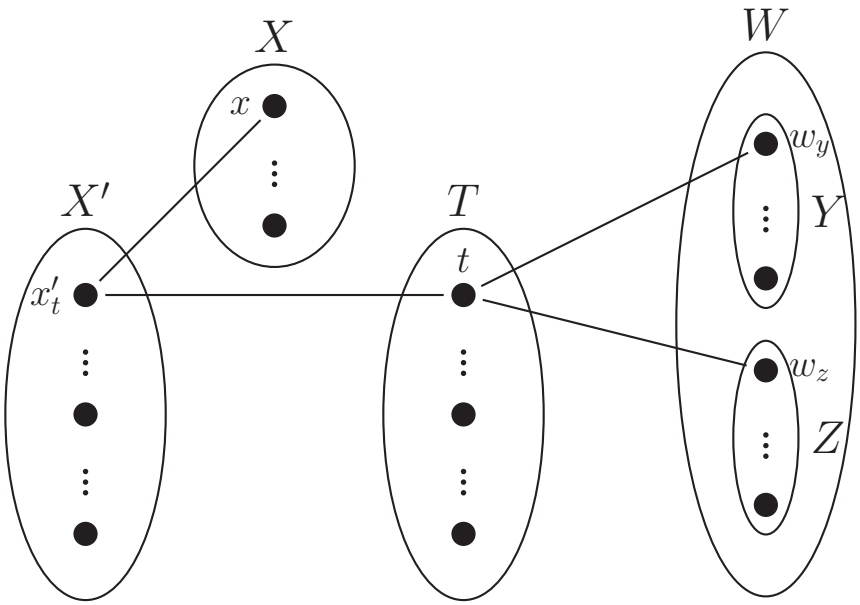


Figure 3.6: Partial construction of graph G

To complete the construction, attach $\deg(w)-1$ disjoint paths of length 3 to each vertex $w \in W$. Denote W_j the set of the j-th vertices of such paths. Additionally, attach $\deg(x) - 1$ disjoint paths of length 2 edges to each vertex $x \in X$. Denote X_j the set of the j-th vertices of such paths. Notice that the constructed graph, illustrated in Figure 3.7, is planar bipartite if and only if graph H is also planar bipartite.

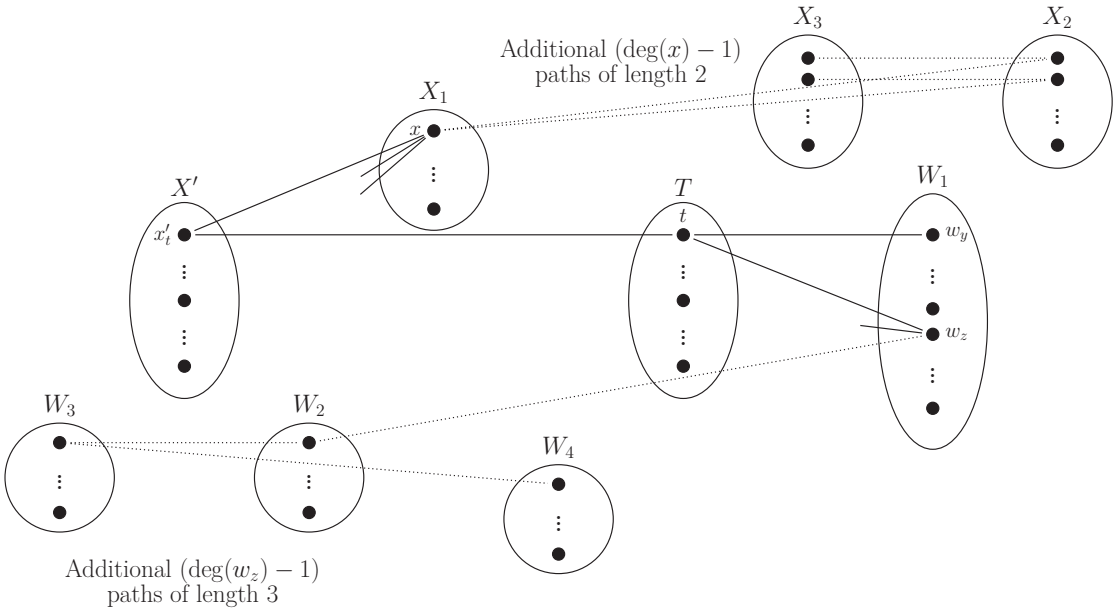


Figure 3.7: Final construction of graph G

Finally, number the vertices of G as they appear in Figure 3.7 from the left to the right, that is, $W_3 \prec X' \prec W_2 \prec X_1 \prec W_4 \prec T \prec X_3 \prec W_1 \prec X_2$. Let $\mathcal{I} = (V(G), E(G), 2)$ be the constructed instance of U-SNP. Such construction can be done in polynomial time since $|V(G)| = 10|T| - \frac{5}{3}|S| = 10|T| - 5q$ and $|E(G)| = 12|T| - \frac{8}{3}|S| = 12|T| - 8q$.

We now claim that the planar 3DM instance has a matching if and only if $\mathcal{I}$ has a solution with $|E(G)| + \frac{|E(G)|}{4}$ stops. Firstly, notice that each vehicle may take at most 4 demands, since capacity constraints must be respected and $\Delta_E(v') \cup \Delta_E(v'') = E$, where v' and v'' are the stations with highest number in X' and X_3, respectively (see Figure 3.8).

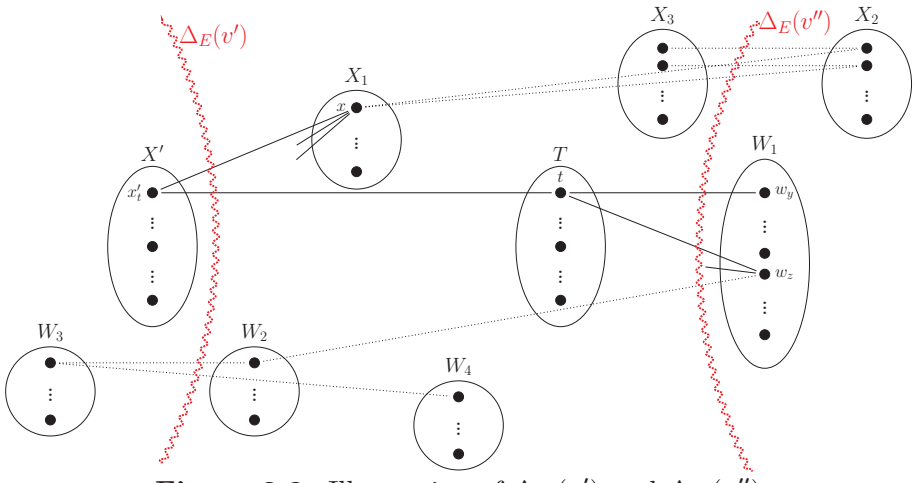


Figure 3.8: Illustration of $\Delta_E(v')$ and $\Delta_E(v'')$

Additionally, for any vehicle i, notice also that $G(E_i)$ cannot contain a cycle

since all cycles in G induce a violation on the vehicle's capacity. Therefore, $G(E_i)$ is a forest and each vehicle i stops $|E_i| + f_i$ times, where f_i is the number of connected components of $G(E_i)$. Given a solution of $\mathcal{I}$, let p' denote the number of non-empty vehicles in such solution. The total number of stops of this solution of $\mathcal{I}$ is therefore given by

$$\sum_{i=1}^{p'} |V[E_i]| = \sum_{i=1}^{p'} (|E_i| + f_i) = |E(G)| + \sum_{i=1}^{p'} f_i.$$

By definition, $\sum_{i=1}^{p'} f_i \geq p' \geq p_{min}$. Moreover, since each vehicle cannot take more than 4 demands, $p_{min} \geq \frac{|E(G)|}{4}$. It follows that $|E(G)| + \frac{|E(G)|}{4}$ is a lower bound on the number of stops of any solution of $\mathcal{I}$. Moreover, if a solution with $|E(G)| + \frac{|E(G)|}{4}$ stops exists, then

$$\sum_{i=1}^{p'} f_i = p' = k_{min} = \frac{|E(G)|}{4}.$$

This means that each non-empty vehicle is assigned exactly 4 connected demands and stops at exactly 5 stations. Therefore, the additional paths of length 3 attached to vertex $w \in W_1$ must be each in a different vehicle. In addition, they must form a connected component of size 4 with some other demand incident to w, coming from T. Removing these components leaves exactly one demand incident to each station in W_1. Analogously, the additional paths of length 2 attached to vertex $x \in X_1$ must be each in a different vehicle. Thus they must form a connected component of size 4 with another path of length 2 linking vertex x to some vertex in T. Removing such components leaves exactly one path incident to each station in X_1. This decomposition leaves G with connected components of size at most 4, each having exactly one vertex in T.

If $\mathcal{I}$ has a solution with $|E(G)| + \frac{|E(G)|}{4}$ stops, the remaining connected components must be trees with one internal vertex in each set T and X', and one leaf in each set X, Y and Z (Figure 3.6 depicts such structure). Therefore, such vehicles induce a 3-dimension matching on graph H. This decomposition can be easily reversed to show that any 3-dimension matching in H induces a solution of $\mathcal{I}$ with $|E(G)| + \frac{|E(G)|}{4}$ stops. ■

The idea used in the proof of Theorem 3.1 can be further exploited to derive a formal proof of $\mathcal{NP}$-Hardness of U-SNP, for other fixed values of C. Indeed, such result can be easily obtained for any fixed capacity $C = 2k$, where $k \in \mathbb{Z}_+^*$.

Theorem 3.2 - $\mathcal{NP}$-Hardness of U-SNP for even capacities
The U-SNP is $\mathcal{NP}$-Hard even when restricted to the case where $C = 2k$, for $k \in \mathbb{Z}_+^*$, and G is a planar bipartite graph.

Proof. Given a planar bipartite graph H associated with an instance of Planar 3DM

(see Figure 3.5), construct graph G depicted in Figure 3.7 in the same manner as described in the proof of Theorem 3.1. Next, construct graph G' by replacing every edge in G by a path of length k. G' can obviously be constructed in polynomial time since k is fixed. Moreover, if $k = 1$ then $G' = G$.

Finally, let $\mathcal{I} = (V(G'), E(G'), 2k)$ be a constructed instance of U-SNP, by numbering the vertices of G' in a way that (i.) every cycle in G' induces a violation on the vehicle's capacity and (ii.) there exist two stations v' and v'' such that

$$\Delta_E(v') \cup \Delta_E(v'') = E, \quad \text{and} \quad |\Delta_E(v')| = |\Delta_E(v'')| = \frac{|E(G')|}{2}.$$

Such numeration can be easily found applying a similar idea as the one used in the proof of Theorem 3.1. Figure 3.9 gives an example (vertices are numbered from the left to the right) of such numeration for $k = 2$ that can be extended for any $k \in \mathbb{Z}_+^*$.

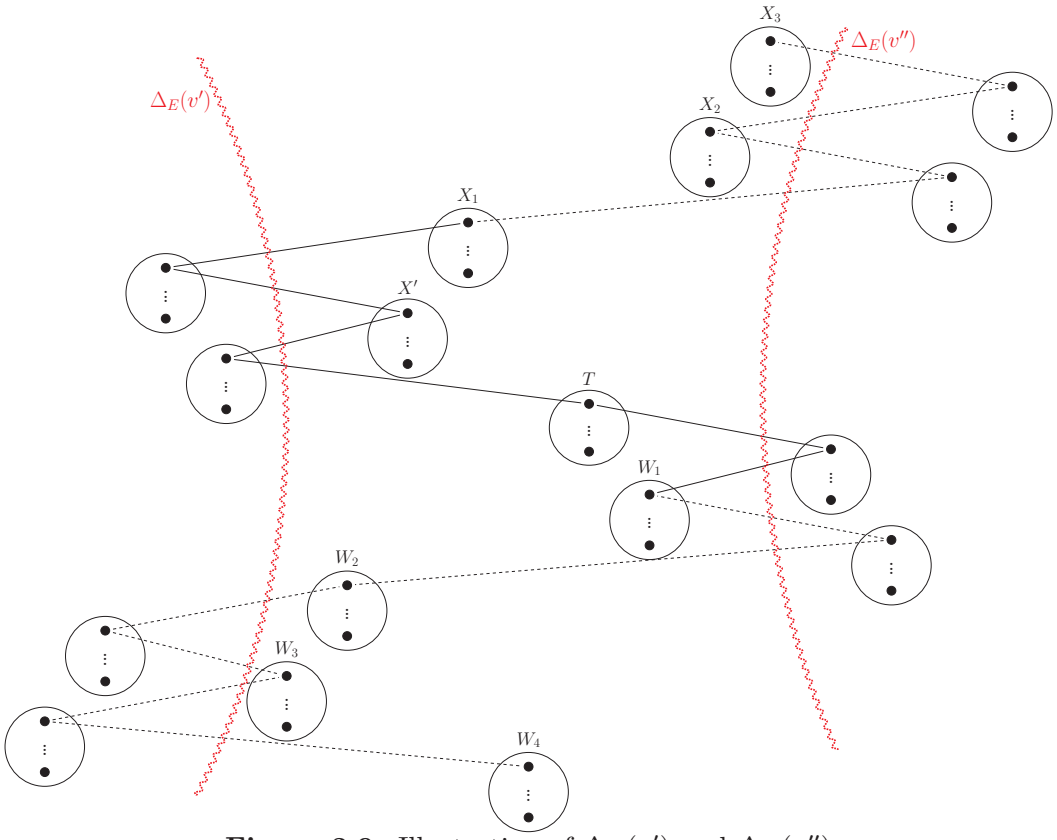


Figure 3.9: Illustration of $\Delta_E(v')$ and $\Delta_E(v'')$

Hence, each vehicle may take at most $2C$ demands and the arguments used in the proof of Theorem 3.1 can now be adjusted and applied here for showing that H has a 3-dimensional matching if and only if $\mathcal{I}$ has a solution with $|E(G)| + \frac{|E(G)|}{2C}$ stops. ■

Recall that Path Traffic Grooming is shown in Amini et al. [3] to be $\mathcal{NP}$-Hard for any fixed grooming factor $g \geq 2$ even when restricted to the case where the graph of requests H is a tripartite graph. Since Path Traffic Grooming is equivalent to U-SNP with no parallel demands and the construction used in the proof of Theorem 3.2 does not include any parallel demand, the complexity result obtained for U-SNP can be extended to the Path Traffic Grooming, reinforcing the known complexity results for even values of grooming factor g.

Corollary 3.1 - Extended result for Path Traffic Grooming
Path Traffic Grooming *is $\mathcal{NP}$-Hard for any fixed grooming factor $g = 2k$, for $k \in \mathbb{Z}_+^*$, even when restricted to the case where the graph of requests H is a planar bipartite graph.* ■

The results presented above can only treat the case where C is even. However, it seems strange that the parity of C plays such an important role on establishing the problem's complexity. It turns out that the $\mathcal{NP}$-Hardness of U-SNP on planar bipartite graphs can indeed be extended to odd values of C. Such result will be achieved through the proof of Theorem 3.4.

After proving that U-SNP is $\mathcal{NP}$-Hard for even values of C and knowing that Intersection U-SNP can be solved in polynomial time for $C = 2$, the question of whether or not Intersection U-SNP is $\mathcal{NP}$-Hard for higher capacities arises immediately. Such question is further investigated below.

The Intersection Case

In Goldschmidt et al. [78], k-Edge-Partitioning Problem is shown to be $\mathcal{NP}$-Hard, for any fixed $k \geq 3$, through a reduction from Edge-Partition into Triangles Problem.

> Edge-Partition into Triangles Problem
> *Input*: An undirected graph $G = (V, E)$.
> *Output*: Decide whether or not there exists a partition of E into sets inducing a subgraph of G isomorphic to K_3.

Recall that if G is a bipartite graph, then k-Edge-Partitioning Problem is equivalent to Intersection U-SNP with no parallel demands. It is therefore intuitive to try applying the same reduction to U-SNP. It turns out, however, that solving Edge-Partition into Triangles Problem on bipartite graphs is trivial as bipartite graphs, by definition, do not contain any odd cycle. Moreover, in Lemaire

[118], k-Edge-Partitioning Problem is also investigated and shown to be solvable in polynomial time for some sub-classes of bipartite graphs and $k = 3$. The authors state however, that for an arbitrary bipartite graph, the problem's complexity is not known. Next, we propose a polynomial reduction from 3-Dimensional Matching to prove Intersection U-SNP is indeed $\mathcal{NP}$-Hard. Such result can then be extended to k-Edge-Partitioning Problem.

Theorem 3.3 - $\mathcal{NP}$-Hardness of Intersection U-SNP for $C = 3$
Intersection U-SNP is $\mathcal{NP}$-Hard even when restricted to the case where $C = 3$ and G is a planar bipartite graph.

Proof. Given an instance of Planar 3DM with disjoint sets X, Y, Z of cardinality q and a set of triples T, let $H = (T, S, E')$ be the planar bipartite graph described in Figure 3.5 where $S = X \cup Y \cup Z$ and

$$E' = \bigcup_{t=(x,y,z)\in T} \{(t,x),(t,y),(t,z)\}.$$

Next, we construct an instance $\mathcal{I}$ of Intersection U-SNP in polynomial time and claim that the 3DM instance has a matching if and only if $\mathcal{I}$ has a solution with a certain amount of stops. For this, let G be a graph obtained from H by attaching $\deg_H(i)-1$ disjoint paths of length $k+1$ to each vertex $i \in S$. Graph G is illustrated in Figure 3.10.

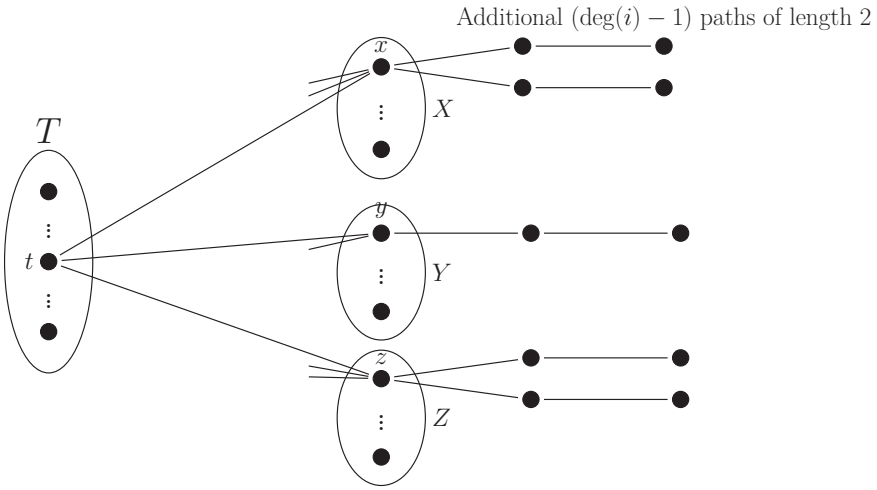


Figure 3.10: Construction of graph G for Theorem 3.3

Notice that if H is planar bipartite, then the constructed graph G is also planar bipartite. Let $\mathcal{I} = (V(G), E(G), 3)$. However, for $\mathcal{I}$ to be an instance of Intersection U-SNP, the stations in $V(G)$ need to be numbered in a way that there exists some station v' wherein all demands intersect. This can be easily done, since graph G

remains bipartite planar. For instance, if W and $\overline{W}$ are the two disjoint independent sets inducing G, one can number vertices in W first and then vertices in $\overline{W}$ afterwards.

The construction of instance $\mathcal{I}$ can be done in polynomial time from any Planar 3DM instance, since $|V(G)| = 7|T| - 3q$ and $|E(G)| = 9|T| - 6q$. We now claim that the planar 3DM instance has a matching if and only if $\mathcal{I}$ has a solution with $|E(G)| + \frac{|E(G)|}{3}$ stops.

Suppose that such solution exists. By definition, a vehicle cannot take more than $C = 3$ demands, and by construction, there is no cycle in G of size smaller than $C + 1$. Therefore, Property 5 stated in Section 3.1 applies and $G(E_i)$ must be a forest, for any vehicle i. Thus, vehicle i stops $|E_i| + f_i$ times, where f_i is the number of connected components of $G(E_i)$.

Let p' denote the number of non-empty vehicles in such solution. Without loss of generality, assume $E_i \neq \emptyset$ for $i \leq p'$ and $E_i = \emptyset$ for $i > p'$. The total number of stops of this solution is therefore given by

$$\sum_{i=1}^{p'} |V[E_i]| = \sum_{i=1}^{p'} (|E_i| + f_i) = |E(G)| + \sum_{i=1}^{p'} f_i.$$

Notice that since $f_i \geq 1$, for any $i \leq p'$, one has that

$$\sum_{i=1}^{p'} f_i \geq p' \geq k_{min} = \left\lceil \frac{|E(G)|}{C} \right\rceil \geq \frac{|E(G)|}{3}.$$

This means that a solution with $|E(G)| + \frac{|E(G)|}{3}$ stops has $\sum_{i=1}^{p'} f_i = \frac{|E(G)|}{3}$. Hence, each non-empty vehicle is assigned 3 connected demands and stops 4 times.

As a consequence, the additional paths of length 2 must be each in a different vehicle. In addition, they must form a connected component of size 3 with some other edge incident to i, coming from T. Removing these components leaves exactly one demand incident to each station in S. This decomposition leaves G with connected components of size at most 3, each having exactly one vertex in T.

If $\mathcal{I}$ has a solution with $|E(G)| + \frac{|E(G)|}{3}$ stops, the remaining connected components are claws $K_{1,3}$ with internal vertex in T, and one leaf in each set X, Y and Z. Therefore, such vehicles induce a 3-dimension matching on graph G. This decomposition can be easily reversed to show that any 3-dimension matching induces a solution of $\mathcal{I}$ with $|E(G)| + \frac{|E(G)|}{3}$ stops. ■

The idea used in the proof of Theorem 3.3 can be further exploited to derive a formal proof of $\mathcal{NP}$-Hardness of Intersection U-SNP, for other fixed values of C.

Theorem 3.4 - $\mathcal{NP}$-Hardness of Intersection U-SNP for odd capacities *The Intersection U-SNP is $\mathcal{NP}$-Hard even when restricted to the case where $C = 2k + 1$, for $k \in \mathbb{Z}_+^*$, and G is a planar bipartite graph.*

Proof. Given an instance of Planar 3DM with disjoint sets X, Y, Z of cardinality q and a set of triples T, consider again planar bipartite graph $H = (T, S, E')$ described in Figure 3.5. From H, let us construct graph G throughout the following operations:

i. Replace the edges between T and $S \setminus X$ with paths of length k.

ii. Attach $\deg_H(i) - 1$ disjoint paths of length $k + 1$ to each vertex $i \in S \setminus X$.

iii. Attach $\deg_H(i) - 1$ disjoint paths of length $2k$ to each vertex $i \in X$.

Graph G is illustrated in Figure 3.11 for $C = 5$.

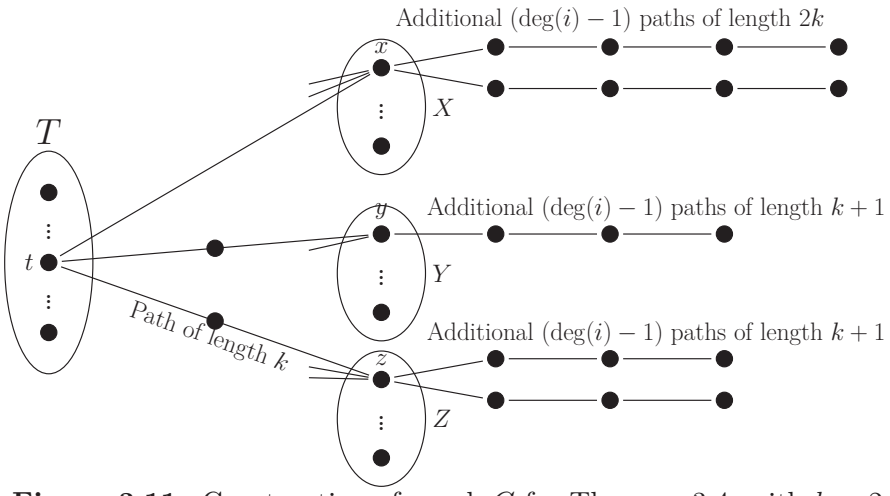


Figure 3.11: Construction of graph G for Theorem 3.4, with $k = 2$

Notice that the construction of G preserves *planarity* and *bipartiteness*. Let $\mathcal{I} = (V(G), E(G), 2k + 1)$ be an instance of Intersection U-SNP obtained by numbering the stations in $V(G)$ in the same manner as done in the proof of Theorem 3.3. The construction of $\mathcal{I}$ can be done in polynomial time as $|V(G)| = (4k+5)|T| - (4k-1)q$ and $|E(G)| = (6k + 3)|T| - (4k + 2)q$.

We claim that the planar 3DM instance has a matching if and only if $\mathcal{I}$ has a solution with $|E(G)| + \frac{|E(G)|}{2k+1}$ stops. Since, by construction, graph G has no cycle of size less than $2k + 2$ (*i.e.,* $C + 1$), all the arguments used for the proof of Theorem 3.3 hold and the rest of this proof is done analogously. ■

From Theorem 3.2 and Theorem 3.4, we set the following corollaries:

Corollary 3.2 - Extended result for U-SNP
The U-SNP is NP-Hard for any fixed capacity $C \geq 2$, even when restricted to the case where G is a planar bipartite graph. ■

Corollary 3.3 - Extended result for Path Traffic Grooming
The Path Traffic Grooming *is NP-Hard for any fixed grooming factor $g \geq 2$, even when restricted to the case where the graph of requests H is a planar bipartite graph.* ■

After proving Intersection U-SNP is $\mathcal{NP}$-Hard for odd capacities, we show next that a similar idea can be used for extending such result to even values of C.

Theorem 3.5 - $\mathcal{NP}$-Hardness of Intersection U-SNP for even capacities
The Intersection U-SNP is $\mathcal{NP}$-Hard even when restricted to the case where $C = 4 + 2k$, for $k \in \mathbb{Z}_+^*$, and G is a planar bipartite graph.

Proof. Given an instance of Planar 3DM with disjoint sets X, Y, Z of cardinality q and a set of triples T, consider again planar bipartite graph $H = (T, S, E')$ described in Figure 3.5. From H, let us construct graph G throughout the following operations:

i. Replace the edges between T and $S \setminus X$ with paths of length $k + 1$.

ii. Replace the edges between T and X with paths of length 2 (*i.e.*, subdivide such edges).

iii. Attach $\deg_H(i) - 1$ disjoint paths of length $k + 3$ to each vertex $i \in S \setminus X$.

iv. Attach $\deg_H(i) - 1$ disjoint paths of length $2k + 2$ to each vertex $i \in X$.

Graph G is illustrated in Figure 3.12 for $C = 10$ (*i.e.*, $k = 3$).

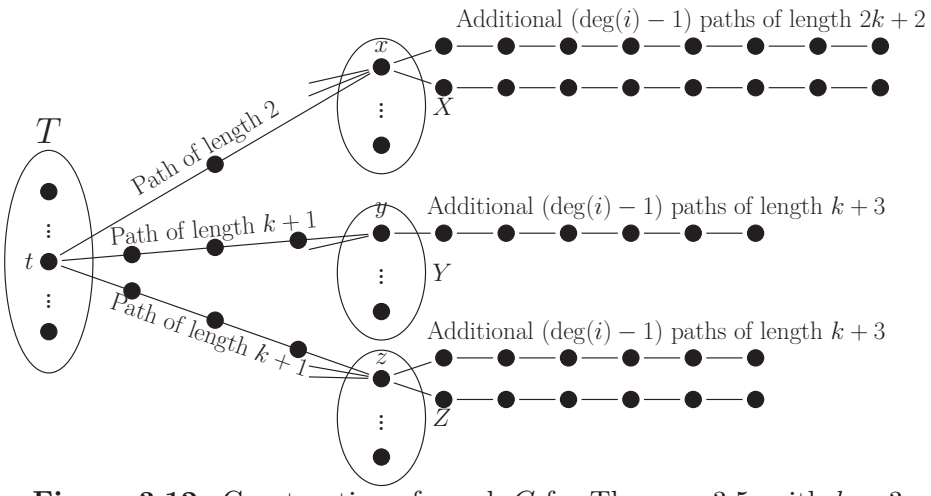


Figure 3.12: Construction of graph G for Theorem 3.5, with $k = 3$

Notice that the construction of G preserves the *planarity* and the *bipartiteness* from graph H. Let $\mathcal{I} = (V(G), E(G), 2k+4)$ be an instance of Intersection U-SNP by numbering the stations in $V(G)$ in the same manner as done in the proof of Theorem 3.3. The construction of $\mathcal{I}$ can be done in polynomial time since k is fixed and $|V(G)| = (6k+10)|T| - (4k-5)q$ and $|E(G)| = (6k+12)|T| - (4k+8)q$.

We claim that the planar 3DM instance has a matching if and only if $\mathcal{I}$ has a solution with $|E(G)| + \frac{|E(G)|}{2k+4}$ stops. Since, by construction, graph G has no cycle of size less than $2k+5$ (*i.e.,* $C+1$), all arguments used for the proof of Theorem 3.3 hold and the rest of this proof is done analogously. ■

From Theorem 3.4 and Theorem 3.5, we set the following corollary.

Corollary 3.4 - Extended result for k-Edge-Partitioning Problem
The k-Edge-Partitioning Problem *is NP-Hard for any fixed* $k \geq 3$, $k \neq 4$, *even when restricted to the case where* G *is a planar bipartite graph.* ■

The key component enabling the development of our proofs of complexity is the ability of rejecting any cycle of size less than or equal to C in the construction of graph G, while maintaining the capacity to reach elements in X, Y and Z from T using only C edges. This is only made possible through the successive subdivisions of edges between T and S. Following this concept, for $C = 4$, only one edge of the claw $\{(t,x),(t,y),(t,z)\}$ is allowed to be subdivided. However, if one tries to subdivide only edges between T and one of the disjoint sets X, Y or Z (*w.l.o.g.*, say Z), cycles of size 4 may appear between sets T and the two other sets (X and Y). As a consequence, the proof of $\mathcal{NP}$-Hardness of Intersection U-SNP when G is planar bipartite remains unclear for $C = 4$. On the other hand, there is no evidence to believe that a tractable algorithm for such case could be conceivable. The fact that $\mathcal{NP}$-Hardness holds for $C = 3$ and $C = 5$, also pushes us in this direction. Therefore, following the steps of Jack Edmonds[2], the following conjecture is proposed.

Conjecture 2 - $\mathcal{NP}$-Hardness of Intersection U-SNP for any fixed capacity
The Intersection U-SNP is NP-Hard even when restricted to the case where $C = 4$ and G is a planar bipartite graph.

[2] "I conjecture that there is no good algorithm for the traveling salesman problem. My reasons are the same as for any mathematical conjecture: (1) It is a legitimate mathematical possibility, and (2) I do not know." – Jack Edmonds, 1966

3.4 Approximability

Once a problem is shown to be $\mathcal{NP}$-Hard, there is little hope that an algorithm with polynomially bounded running time can be achieved. Indeed, finding such algorithm would proof $\mathcal{P} = \mathcal{NP}$, answering one of the "million dollar" questions that has been puzzling mathematicians for such a long time (see Clay Mathematics Institute [34]). For such problems, one may decide to sacrifice optimality in order to obtain a good solution within polynomially bounded time. Algorithms based on such principle are called heuristics. If an heuristic is capable of always providing near-optimal solutions, that is, if there exists a provable guarantee on the quality of the provided solutions, the heuristic is called an approximation algorithm. In this section, approximation algorithms for the U-SNP are investigated. We present some preliminary results that were obtained in collaboration with José R. Correa, Associate Professor at the Department of Industrial Engineering from Universidad de Chile, Santiago, Chile, during a one-month visit at Universidad de Chile.

The approximation ratio is a widely accepted measure of the goodness of approximation algorithms and can be defined as follows. Let P be an optimization (minimization) problem and A be an algorithm providing a feasible solution for P. Given an instance $\mathcal{I}$ of P, let $OPT(\mathcal{I})$ denote the value of an optimal solution to instance $\mathcal{I}$. Moreover, let $A(\mathcal{I})$ denote the value of the solution provided by algorithm A for instance $\mathcal{I}$. The approximation ratio of algorithm A is defined as:

$$R_A = \max_{\mathcal{I}} \frac{A(\mathcal{I})}{OPT(\mathcal{I})}.$$

As previously stated, Path Traffic Grooming is known to be $\mathcal{APX}$-Hard, that is, it can be approximated within a constant ratio but does not admit an approximation ratio of $1+\epsilon$ unless $\mathcal{P} = \mathcal{NP}$. By restriction, one easily deduces that U-SNP is therefore also $\mathcal{APX}$-Hard.

3.4.1 An initial C-approximation algorithm

For developing an approximation algorithm for U-SNP, we use the fact that U-SNP can be solved optimally in polynomial time for $C = 1$ (see Proposition 3.9). Given an instance (V, E, C) of U-SNP, let $\{E_1^*, E_2^*, \ldots, E_p^*\}$ be an optimal solution of (V, E, C) and let $OPT(V, E, C)$ denote its value. It is clear that increasing the vehicle's capacity constitutes a relaxation of the problem and hence

$$OPT(V, E, C) \leq OPT(V, E, C-1) \leq \cdots \leq OPT(V, E, 1). \tag{3.3}$$

Such statement is evidenced by taking a look at formulation $\text{USNP}_{(V,E,C)}$ and noticing that $\text{USNP}_{(V,E,C)} \subseteq \text{USNP}_{(V,E,C+1)}$.

Proposition 3.15 - A C-approximation algorithm for U-SNP
The algorithm described in Proposition 3.9 for solving an instance $(V, E, 1)$ of U-SNP is a C-approximation algorithm for instance (V, E, C).

Proof. If $\{E_1^*, E_2^*, \ldots, E_p^*\}$ is an optimal solution to (V, E, C), one can easily partition demands in E_1^* into C vehicles of capacity 1 by solving the instance $(V, E_1^*, 1)$. Notice that, in the worst case, each "new" unit capacity vehicle stops at every station in $V[E_1^*]$. Therefore,

$$OPT(V, E_1^*, 1) \leq C|V[E_1^*]|.$$

By repeating such procedure for each vehicle $1 \leq i \leq p$, one obtains a solution for (V, E, C) with value

$$\sum_{i=1}^{p} OPT(V, E_i^*, 1) \leq C\Big(OPT(V, E, C)\Big). \tag{3.4}$$

Lastly, it suffices to notice that such constructed solution of (V, E, C) is also a solution for $(V, E, 1)$, and therefore

$$OPT(V, E, 1) \leq \sum_{i=1}^{p} OPT(V, E_i^*, 1).$$

Combining inequalities (3.3), (3.4) and (3.4.1), one finally gets a proof of approximability:

$$OPT(V, E, C) \leq OPT(V, E, 1) \leq C\Big(OPT(V, E, C)\Big). \qquad \blacksquare$$

Notice that if (V, E, C) is an instance of Intersection U-SNP, one may take advantage of the fact that $(V, E, 2)$ is optimally solvable in polynomial time (see Proposition 3.14), to derive a better approximation algorithm. Indeed, by following the same steps as in the proof of Proposition 3.15, one can easily remark that

$$OPT(V, E, C) \leq OPT(V, E, 2) \leq \left\lceil \frac{C}{2} \right\rceil \Big(OPT(V, E, C)\Big).$$

Next we present a better constant-ratio approximation algorithm solving U-SNP when capacity is fixed to 2 – recall that such case is $\mathcal{NP}$-Hard as shown by Theorem 3.1.

3.4.2 A $\frac{5}{3}$-approximation algorithm for $C = 2$

We use the following notation to describe the algorithm. Given an instance $\mathcal{I} = (V, E, 2)$ of U-SNP, let $E_{uv} = \{e \in E : o_e = u \text{ and } d_e = v\}$ denote the set of parallel demands uv in E. Then, consider an algorithm that, at first, merges pairs of parallel demands into a single demand and then solves the problem optimally with $C = 1$ on the resulting graph. Such algorithm is denoted *Merge* and is formally defined as follows.

Algorithm 1 Merge Algorithm

Input: An instance $\mathcal{I} = (V, E, 2)$ of U-SNP
Output: A feasible solution for $\mathcal{I}$
1: $E' = \emptyset$
2: **for all** $u \in V$ **do**
3: **for all** $v \in V$ **do**
4: Add $\left\lceil \frac{|E_{uv}|}{2} \right\rceil$ edges uv to E'
5: **end for**
6: **end for**
7: Optimally solve instance $(V, E', 1)$

Merge Algorithm clearly has a polynomial bounded running time since all its steps can be done in polynomial time. In fact, as stated in Algorithm 1 it may run in $\mathcal{O}(n^2)$ time, but with the right data structures this can be improved to $\mathcal{O}(m)$ time. Moreover, since no two parallel demands can be assigned to the same vehicle in instance $(V, E', 1)$, Merge Algorithm provides a feasible solution for $(V, E, 2)$. We now prove its approximability.

Proposition 3.16 - A $\frac{5}{3}$-approximation algorithm
Merge Algorithm is a $\frac{5}{3}$-approximation algorithm for U-SNP when $C = 2$.

Proof. Let $\text{Merge}(V, E, 2)$ denote the value of the solution obtained by applying Merge Algorithm over instance $(V, E, 2)$. Clearly,

$$OPT(V, E, 2) \leq \text{Merge}(V, E, 2). \tag{3.5}$$

Moreover, for any feasible solution $\{E_1, E_2, \ldots, E_p\}$ to instance $(V, E, 2)$, it is easy to see that

$$\text{Merge}(V, E, 2) \leq \sum_{i=1}^{p} \text{Merge}(V[E_i], E_i, 2). \tag{3.6}$$

Therefore, if $\{E_1^*, E_2^*, \ldots, E_p^*\}$ is an optimal solution to $(V, E, 2)$, it follows that

$$\text{Merge}(V, E, 2) \leq \sum_{i=1}^{p} \text{Merge}(V[E_i^*], E_i^*, 2). \tag{3.7}$$

Without loss of generality, $G[E_i^*]$ is assumed to be a connected graph, for any $1 \leq i \leq p$. We claim that

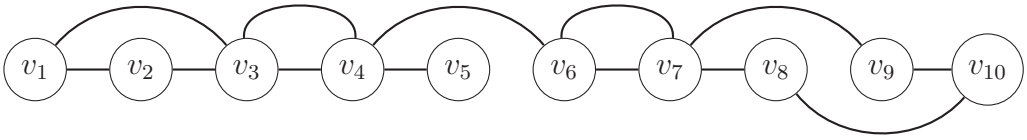
$$\sum_{i=1}^{p} \text{Merge}(V[E_i^*], E_i^*, 2) \leq \sum_{i=1}^{p} \frac{5}{3}|V[E_i^*]| = \frac{5}{3}OPT(V, E, 2).$$

For this, let us consider Merge Algorithm with instance $(V[E_i^*], E_i^*, 2)$ as its input, for each $i \in \{1, \ldots, p\}$. We then show that

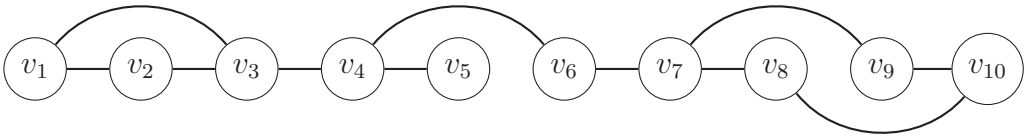
$$\text{Merge}(V[E_i^*], E_i^*, 2) \leq \frac{5}{3}|V[E_i^*]|,$$

for any $1 \leq i \leq p$.

Since $C = 2$, for any pair of nodes $u \in V[E_i^*]$ and $v \in V[E_i^*]$, one has $|E_{uv} \cap E_i^*| \leq 2$. Therefore, graph $G' = (V[E_i^*], E_i^{*\prime})$ constructed in the first phase of Merge Algorithm has no parallel demands (*i.e.*, G' is simple). Figure 3.13 gives an example for the construction of such graph.



(a) Graph $G[E_i^*]$



(b) Graph G' constructed by Merge Algorithm with instance $(V[E_i^*], E_i^*, 2)$ as its input

Figure 3.13: Construction of G'

After constructing $G' = (V[E_i^*], E_i^{*\prime})$, Merge Algorithm optimally solves instance $(V[E_i^*], E_i^{*\prime}, 1)$. From Proposition 3.9, one has that

$$\text{Merge}(V[E_i^*], E_i^*, 2) = \sum_{v \in V[E_i^*]} \max\{|\delta_{G'}^-(v)|, |\delta_{G'}^+(v)|\}.$$

Furthermore, $\max\{|\delta_{G'}^-(v)|, |\delta_{G'}^+(v)|\} \leq 2$, for any $v \in V[E_i^*]$. We next show that given three consecutive nodes v_i, v_{i+1} and v_{i+2} in G', at least one of them cannot have $\max\{|\delta_{G'}^-(v)|, |\delta_{G'}^+(v)|\} = 2$ and hence,

$$\sum_{v \in V[E_i^*]} \max\{|\delta_{G'}^-(v)|, |\delta_{G'}^+(v)|\} \leq \frac{5}{3}|V[E_i^*]|.$$

Suppose that such statement is false. Then, $\max\{|\delta^-_{G'}(v)|, |\delta^+_{G'}(v)|\} = 2$, for $v \in \{v_i, v_{i+1}, v_{i+2}\}$. For this, suppose $|\delta^-_{G'}(v_i)| = 2$. Hence, $|\delta^-_{G'}(v_{i+1})| \leq 1$, for otherwise there exists some node $j \prec v_i$ in $V[E^*_i]$ for which $\Delta_{E^{*\prime}_i}(j) > C$, a contradiction (see Figure 3.14a). Therefore, $|\delta^+_{G'}(v_{i+1})| = 2$. However, this imposes $\max\{|\delta^-_{G'}(v_{i+2})|, |\delta^+_{G'}(v_{i+2})|\} \leq 1$ (see Figure 3.14b). The proof follows the same reasoning if one supposes $|\delta^+_{G'}(v_i)| = 2$.

(a) $|\delta^-_{G'}(v_{i+1})|$ cannot be greater than 1 **(b)** $|\delta^+_{G'}(v_{i+2})|$ cannot be greater than 1

Figure 3.14: Consequences of having $|\delta^-_{G'}(v_i)| = 2$

Finally, one has

$$OPT(V, E, 2) \leq \text{Merge}(V, E, 2) \leq \sum_{i=1}^{p} \text{Merge}(V[E^*_i], E^*_i, 2) \leq \frac{5}{3} OPT(V, E, 2),$$

which proves Merge Algorithm has an approximation ratio of $\frac{5}{3}$. ■

Moreover, the approximation ratio of $\frac{5}{3}$ from Merge Algorithm is tight. This can be easily verified by considering the instance of U-SNP with $C = 2$ described in Figure 3.15. While Merge Algorithm yields a solution with 5 stops for such instance, the optimal solution consists of assigning all three demands to the same vehicle, which produces only 3 stops.

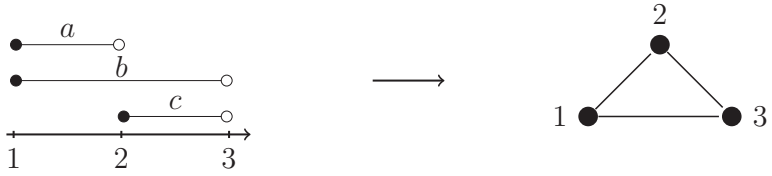


Figure 3.15: Instance showing the tightness of Merge Algorithm's approximation ratio

3.5 Conclusion

In this chapter we have investigated the computational complexity of U-SNP. At an initial stage, optimal solutions properties were studied in order to better understand the combinatorial problem behind U-SNP. A series of lower bounds for the U-SNP were then introduced and some particular cases were shown to be solvable in polynomial time. Based on such results we finally derived a proof of $\mathcal{NP}$-Hardness for the

U-SNP, answering the conjecture proposed by Pimenta et al. [151]. Our presented proofs remain valid for any fixed capacity $C \geq 2$ even if the associated graph G is said to be planar bipartite.

To the best of our knowledge, the variant where every demand intersect a given station has not received particular attention in the literature. Such variant is also taken into account in our complexity study. Figure 3.16 gives a comprehensive picture of the computational complexity of U-SNP when the associated graph G is planar bipartite. It is important to notice that the obtained results are shown to be extensible to other related problems such as the Traffic Grooming problem. In the last section of the chapter, a preliminary study on the approximation of U-SNP is presented, yielding a $\frac{5}{3}$-approximation algorithm for the case where $C = 2$.

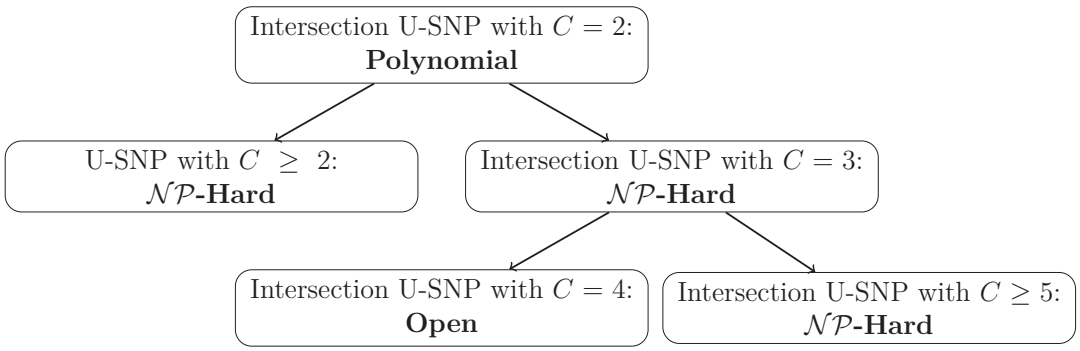


Figure 3.16: An overview of the complexity of U-SNP.

Chapter 4

Polyhedral study

> "True optimization is the revolutionary contribution of modern research to decision processes."
>
> — George B. Dantzig

If one is interested in exact optimal solutions, a common approach when tackling hard problems is to formulate and solve it as a MILP. Many combinatorial problems have taken benefit from the study of the facial structure of the convex hull of feasible solutions – also known as the *integer polyhedron* – for improving the associated MILP performances. Indeed, if a complete description of the integer polyhedron is known, the problem can be solved as a linear program over such polyhedron. Unfortunately, the complete description of such polyhedrons usually involves an exponential number of inequalities. Here, the equivalence between optimization and separation described by Theorem 1.9 in Section 1.5 plays an important role. Indeed, if the associated separation problem can be solved in polynomial time, then the optimization of a linear function over the polyhedron can also be done in polynomial time. The matching problem (see Edmonds [59, 60], Pulleyblank [153]) is a classic example where the integer polyhedron has been fully described and its separation problem shown to be solvable in polynomial time. On the other hand, a partial description of the integer polyhedron of $\mathcal{NP}$-Hard problems is also desired when designing good algorithms. Notable examples of such cases are the traveling salesman problem (see Dantzig et al. [47], Grötschel and Padberg [81], Padberg and Hong [142], Balas and Christofides [9], Fischetti and Toth [69], Applegate et al. [5]) and the vertex packing problem (see Nemhauser and Trotter [136, 137], Wolsey [179]). In this chapter, such polyhedral study is conducted for the U-SNP.

4.1 Methodology and Preliminary Analysis

Let us recall the formulation provided in Pimenta et al. [151] for the U-SNP:

$$\min \sum_{v \in V} \sum_{i \in K} y_v^i \tag{4.1}$$

subject to

$$\sum_{i \in K} x_e^i = 1 \qquad \forall e \in E, \tag{4.2}$$

$$\sum_{e \in \Delta_E(v)} x_e^i \leq C \qquad \forall v \in V, i \in K, \tag{4.3}$$

$$x_e^i - y_v^i \leq 0 \qquad \forall i \in K, e \in E, v \in \{o_e, d_e\}, \tag{4.4}$$

$$x_e^i \in \{0,1\} \qquad \forall e \in E, i \in K, \tag{4.5}$$

$$y_v^i \in \{0,1\} \qquad \forall v \in V, i \in K. \tag{4.6}$$

When dealing with VIPA's fleet management, gathering real instances is not an easy task. To the best of our knowledge, there is no standard benchmark sets of instances when dealing with such problems. In Pimenta et al. [151], Deleplanque et al. [50], Bsaybes et al. [26], instances were randomly generated. Moreover, most of the work done around the Stop Number Problem has considered weighted demands and/or a fixed number of tours (see Section 2.3). This prevents the reuse of the instances considered in such cases for our purposes.

For these reasons, in order to evaluate the strength of formulation (4.1)–(4.6), a new set of 315 randomly generated instances is proposed for the U-SNP as follows. Given a fixed number of stations n, two different numbers (say a and b) are picked at random between 1 and n to create a demand going from station a to station b if $a < b$ (or from b to a, otherwise). Notice however that the density of demands with respect to the number of stations, denoted by $\rho = \frac{m}{n}$, is a much more meaningful parameter than the number of stations n. The density provides information on the average degree of each station (the average degree corresponds to 2ρ) and therefore, allows the decision maker to forecast whether or not demands are likely to share a common station. Furthermore, once the density ρ and the number of demands m have been fixed, the number of stations is directly obtained from the definition of ρ.

A set of different scenarios based on the number of demands m, the capacity C and the density ρ is thus provided with $m \in \{30, 35, 40, 45, 50, 55, 60\}$, $C \in \{2, 5, 8\}$ and $\rho \in \{1.5, 3.0, 4.5\}$. For each combination of parameters, 5 instances were randomly generated. Each generated instance can then be referred to as the code $m\_C\_\rho\_i$, where i refers to the i-th instance generated with the given parameters.

For analysing the performance of formulation (4.1)–(4.6), we implemented it in C++ using CPLEX 12.8. All default settings of CPLEX (presolving, heuristics, cut generation and dynamic search, among other features) were left untouched. A computer machine provided with an Intel Xeon E5, 3.10GHz and 32 Gb of RAM was used for running performance tests. A time limit of 2 hours was set for solving each instance. The number of available threads was set to one.

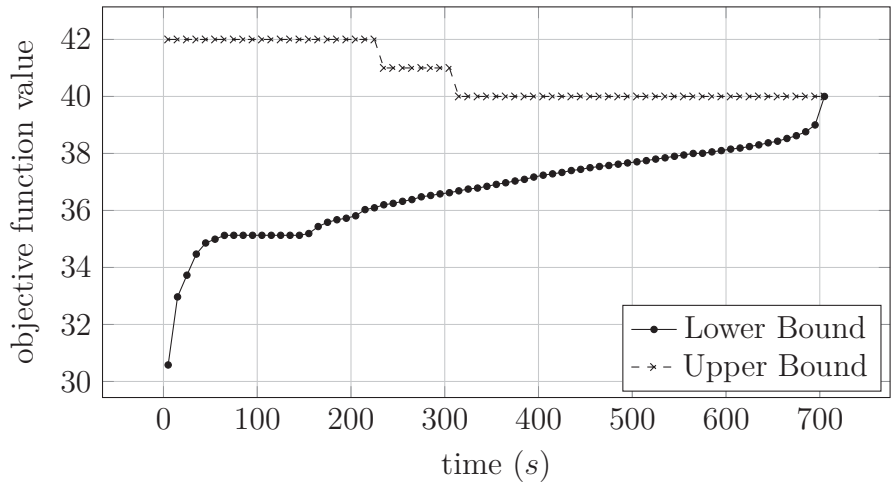
For each combination of parameters m, C and ρ, Table 4.1 gives the average results obtained with such formulation, from the 5 generated instances. For detailed information on each instance, the reader is invited to check Table A.1 on Appendix A. On Table 4.1, section *Instances* provides information about the instance's parameters. Columns *cols* and *rows* refer to the number of variables and inequalities, respectively - after the presolve elimination from CPLEX. Moreover, p_{min} refers to the minimum number of vehicles needed to satisfy all demands and p' to the number of vehicles used on the best solution found.

Next in order, section *Root* provides information about the optimization at the root node of the Branch-and-Bound tree, where $time_r$ refers to the time in seconds needed to solve the root node, LB_r and UB_r refer, respectively, to the best lower and upper bounds known at the root node and gap_r refers to the gap obtained after solving the root node calculated as $gap_r = \frac{UB_r - LB_r}{UB_r}$.

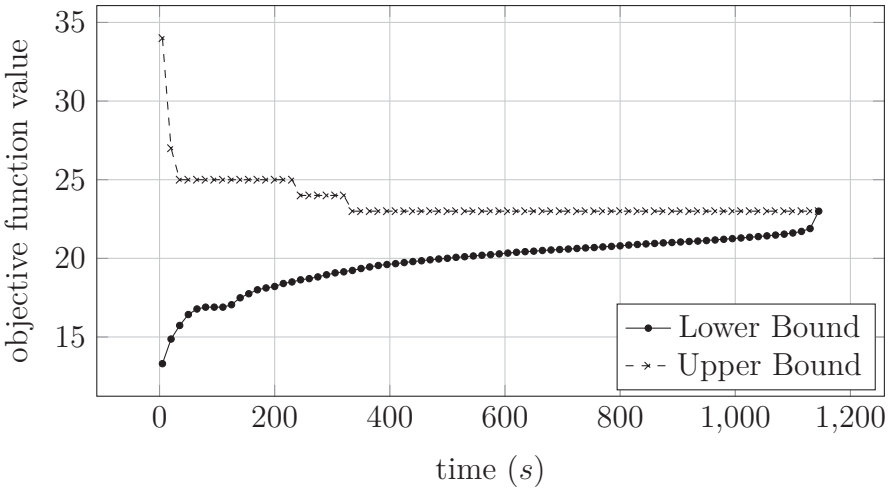
At last, section *Global* provides information about the global optimization process, where $time_g$ refers to the time in seconds needed to solve the optimization problem and *timeBest* to the time in seconds required to find the best integer solution. Notice that $timeBest \leq time_g$. The difference between $time_g$ and *timeBest* corresponds to time spent proving the optimality of the solution found. Moreover, LB_g and UB_g refer, respectively, to the best lower and upper bounds known by the end of the optimization process and gap_g refers to the final gap which is calculated as $gap_g = \frac{UB_g - LB_g}{UB_g}$. Notice that if a solution could be found and proven to be optimal within the time limit, then LB_g equals UB_g and the gap_g is, by definition, null. Column *Nodes* refers to the number of nodes (in thousands) in the Branch-and-Bound tree processed during the optimization and column *NLeft* to the number of nodes in the Branch-and-Bound tree that remained to be processed by the end of the time limit. Clearly, *NLeft* is null if an optimal solution could be found within the time limit. Finally, column $Cuts_{CP}$ refers to the total number of cuts generated by CPLEX during the whole optimization process. More specifically, $Cuts_{CP}$ accounts for Cover cuts (see Crowder et al. [43], Balas [7]), GUB Cover cuts (see Wolsey [180], Gu et al. [86]), Flow Cover cuts (see Padberg et al. [143], Gu et al. [87]), Clique cuts (see Padberg [141]), Gomory Fractional cuts, Mixed-Integer Rounding cuts and Zero-half cuts (see Gomory et al. [80], Gomory [79]), Flow Path cuts (see

Van Roy and Wolsey [176]), Disjunctive cuts (see Balas [8]), Implied Bounds cuts (see Hoffman and Padberg [94]), Multi-Commodity Flow cuts (see Achterberg and Raack [1]) and Lift-and-Project cuts (see Balas et al. [10], Lovász and Schrijver [122]).

By comparing the values of $time_g$ and *timeBest* obtained in Table 4.1, one can easily see that frequently enough the best solution is found way before the end of the optimization process. Indeed, on average, more than 54% of the time spent on the optimization is dedicated to proving the optimality of an incumbent solution. This means that the upper bound converges to the optimum value much faster than the lower bound. Figure 4.1 illustrates such situation by depicting the evolution of lower and upper bounds during the optimization of an instance with 45 demands and capacity 5. Figure 4.1a gives a scenario with sparse demands - density is set to 1.5 - and Figure 4.1b gives a scenario with dense demands - density is set to 4.5.



(a) Sparse scenario: instance 45_5_1.5_1



(b) Dense scenario: instance 45_5_4.5_1

Figure 4.1: Progress of lower and upper bounds

Table 4.1: Computational results of formulation (4.1)–(4.6)

Instances							Root				Global							
m	C	ρ	cols	rows	p_{min}	p'	time$_r$	LB$_r$	UB$_r$	gap$_r$	time$_g$	timeBest	LB$_g$	UB$_g$	gap$_g$	Nodes	NLeft	Cuts$_{CP}$
30	2	1.5	1500	2178	8.8	9.2	1.27	22.5	41.6	45.35	168.9	83.3	35.0	35.0	0.00	20.0	0.0	32.4
30	2	3.0	1200	2046	9.6	9.6	1.34	13.9	39.4	64.8	205.4	53.4	29.6	29.6	0.00	32.5	0.0	36.8
30	2	4.5	1080	1980	8.8	8.8	1.34	9.9	31.4	68.12	33.4	20.4	26.0	26.0	0.00	3.9	0.0	37
35	2	1.5	2030	2933	11.4	11.6	1.78	25.2	52.8	52	758.3	143.3	41.2	41.2	0.00	72.3	0.0	32.2
35	2	3.0	1610	2765	11.0	11.0	1.95	15.5	44.8	64.76	1606.0	963.7	33.6	33.6	0.00	179.2	0.0	38.4
35	2	4.5	1470	2681	11.0	11.0	2.08	12.5	39	67.87	746.0	69.9	31.6	31.6	0.00	108.1	0.0	47.2
40	2	1.5	2640	3768	11.2	12.8	2.92	28.2	63	55.16	5003.7	413.1	46.3	47.6	2.74	324.0	98.8	27.6
40	2	3.0	2120	3632	11.4	11.4	3.02	16.8	51.6	67.31	7200.0	259.2	34.4	38.8	11.33	312.8	181.2	21
40	2	4.5	1920	3520	11.0	11.0	3.25	11.9	45.8	73.75	4151.9	386.1	34.0	35.0	2.88	321.1	101.3	39.8
45	2	1.5	3375	4770	12.4	13.8	3.68	32.2	70.2	54.09	6503.5	1272.5	50.5	52.8	4.28	265.2	79.5	27.8
45	2	3.0	2700	4563	13.4	13.8	3.72	18.9	59.6	68.25	7200.0	2231.8	36.5	45.0	18.82	166.8	110.1	19.4
45	2	4.5	2475	4464	14.2	14.6	4.32	15.5	57	72.61	7200.0	1828.5	37.6	42.0	10.46	228.0	117.5	51
50	2	1.5	4150	6010	14.2	15.2	6.08	35.1	78	54.86	7200.0	2217.8	51.3	60.0	14.46	91.6	45.6	59.2
50	2	3.0	3300	5660	15.4	16.2	5.16	19.9	70.6	71.71	7200.0	1025.8	38.6	51.4	24.88	84.1	40.7	45
50	2	4.5	3050	5430	14.8	15.0	5.09	15.5	62.6	75.18	7200.0	2290.8	35.3	47.0	24.72	103.8	58.8	35.4
55	2	1.5	5005	7227	15.4	16.4	17.12	37.7	85.4	55.75	7200.0	3446.8	52.7	64.4	18.12	64.6	36.2	54.2
55	2	3.0	4015	6787	16.0	16.4	6.78	21.9	77.6	71.79	7200.0	3511	38.1	56.6	32.74	65.1	38.0	39
55	2	4.5	3685	6644	16.6	16.8	6.41	15.9	76.4	79.19	7200.0	3776.3	35.5	51.8	31.31	64.7	35.9	37.4
60	2	1.5	6000	8580	17.4	18.6	29.49	41.9	95.8	56.13	7200.0	4868.6	57.8	72.0	19.80	39.7	20.2	39.2
60	2	3.0	4800	8172	17.8	18.8	13.48	23.9	88.2	72.72	7200.0	6556.6	42.1	63.8	34.04	39.9	25.0	73
60	2	4.5	4380	7860	17.8	18.6	9.77	16.9	83.2	79.63	7200.0	2195.9	35.3	57.6	38.67	43.9	19.8	60.2
30	5	1.5	1500	2166	4.2	4.2	1.9	21.3	28.2	24.34	10.7	5.2	26.8	26.8	0.00	0.9	0.0	13.4
30	5	3.0	1200	2046	4.0	4.0	1.79	12.5	20	37.39	16.2	9.1	18.6	18.6	0.00	1.3	0.0	16.4

Continued on Next Page...

Table 4.1 – Continued

Instances							Root				Global							
m	C	ρ	cols	rows	p_{min}	p'	time$_r$	LB$_r$	UB$_r$	gap$_r$	time$_g$	timeBest	LB$_g$	UB$_g$	gap$_g$	Nodes	NLeft	Cuts$_{CP}$
30	5	4.5	1080	1974	3.8	3.8	1.7	8.8	14.2	37.87	5.1	3	13.4	13.4	0.00	0.3	0.0	21.6
35	5	1.5	2030	2954	4.6	4.6	3.33	24.2	34.6	29.72	66.8	55.2	31.0	31.0	0.00	3.4	0.0	14.2
35	5	3.0	1610	2758	4.6	4.6	2.69	13.4	24.6	44.71	276.2	136.7	21.8	21.8	0.00	15.8	0.0	23
35	5	4.5	1470	2695	4.8	4.8	2.43	9.7	19.2	48.86	22.6	7	16.8	16.8	0.00	1.1	0.0	48.4
40	5	1.5	2640	3928	4.4	4.6	4.53	27.8	41.6	32.45	266.4	131.4	36.0	36.0	0.00	10.7	0.0	15.2
40	5	3.0	2120	3632	5.2	5.2	3.95	15.5	30.6	49.06	914.2	569.9	25.4	25.4	0.00	24.7	0.0	25.2
40	5	4.5	1920	3488	5.0	5.0	3.57	10.8	23.4	53.51	286.6	77.8	19.4	19.4	0.00	9.8	0.0	26.4
45	5	1.5	3375	4752	6.0	6.4	4.79	31.2	46.4	32.72	1191.3	434.3	41.4	41.4	0.00	30.7	0.0	24.8
45	5	3.0	2700	4536	6.0	6.0	4.45	17.5	45.4	60.73	7200.0	614.7	27.6	30.8	10.51	123.8	54.2	37
45	5	4.5	2475	4446	5.8	5.8	4.21	12.9	33.2	61	4336.6	802.1	23.6	24.2	2.59	109.9	15.7	27.2
50	5	1.5	4150	5940	5.8	6.2	8.17	34.4	56.2	38.13	5684.2	1850.3	44.9	46.8	3.92	93.3	23.5	29
50	5	3.0	3300	5670	6.4	6.4	6.16	18.6	51.6	62.34	6589.8	3568.1	28.7	32.8	12.25	74.2	29.2	24.8
50	5	4.5	3050	5470	6.8	6.8	5.66	13.7	47.8	71.19	7200.0	2877.2	23.8	28.8	17.25	88.1	47.7	45.6
55	5	1.5	5005	7161	7.2	7.6	42.2	37.6	70.8	46.4	6586.7	3101.1	47.5	50.6	6.04	65.9	17.0	36.8
55	5	3.0	4015	6864	6.4	6.8	8.89	20.6	60.6	65.69	7200.0	4516.6	29.6	38.0	21.96	53.9	23.9	31.6
55	5	4.5	3685	6578	6.8	6.8	7.61	14.7	56.2	73.8	7200.0	3952.5	24.0	31.6	23.92	61.6	28.9	31
60	5	1.5	6000	8568	6.6	7.2	88.2	40	79.4	48.86	7200.0	2533.3	47.9	54.6	12.15	37.7	18.9	33.6
60	5	3.0	4800	8148	7.0	7.6	11.63	22.7	68.2	66.7	7200.0	6735.9	31.1	42.0	25.85	30.5	13.7	46.8
60	5	4.5	4380	7920	6.8	6.8	22.86	15.9	59.6	73.23	7200.0	4209.3	24.6	34.4	28.36	28.3	13.0	50.2
30	8	1.5	1500	2148	3.0	3.0	1.8	20.6	23.6	12.53	2.3	1.8	23.6	23.6	0.00	0.0	0.0	5.4
30	8	3.0	1200	2046	2.4	2.4	1.4	11.3	14.6	22.25	2.3	1.4	14.6	14.6	0.00	0.0	0.0	4.8
30	8	4.5	1080	1974	2.8	2.8	1.27	6.4	9.2	24.45	2.4	1.4	11.0	11.0	0.00	0.1	0.0	10.8
35	8	1.5	2030	2912	3.2	3.2	3.18	23.7	27.6	14.14	5.7	3.2	27.6	27.6	0.00	0.1	0.0	6.6

Continued on Next Page...

Table 4.1 – Continued

Instances							Root				Global							
m	C	ρ	cols	rows	p_{min}	p'	time$_r$	LB$_r$	UB$_r$	gap$_r$	time$_g$	timeBest	LB$_g$	UB$_g$	gap$_g$	Nodes	NLeft	Cuts$_{CP}$
35	8	3.0	1610	2786	2.8	2.8	2.65	12.8	17.2	24.8	8.6	3.6	17.0	17.0	0.00	0.3	0.0	5.6
35	8	4.5	1470	2674	3.0	3.0	2.34	8.9	12.8	30.4	5.1	2.3	12.8	12.8	0.00	0.1	0.0	8.8
40	8	1.5	2640	3816	3.4	3.4	5.59	26.5	34	21.89	28.6	22.1	32.0	32.0	0.00	0.7	0.0	6.2
40	8	3.0	2120	3608	3.6	3.6	4.84	15	23	34.2	143.9	65.9	21.8	21.8	0.00	3.4	0.0	9.8
40	8	4.5	1920	3504	3.4	3.4	3.21	10.1	15.6	34.61	34.4	6.6	15.2	15.2	0.00	1.1	0.0	14.4
45	8	1.5	3375	4887	3.4	3.4	6.9	30.2	39	22.6	73.4	52.6	36.8	36.8	0.00	1.3	0.0	6.4
45	8	3.0	2700	4581	3.8	3.8	5.49	16.9	27.4	38.03	480.0	279.4	24.8	24.8	0.00	8.0	0.0	13
45	8	4.5	2475	4455	3.6	3.6	6.64	12.1	20.4	40.36	320.8	55.8	19.2	19.2	0.00	7.0	0.0	18.6
50	8	1.5	4150	5910	4.2	4.4	14.34	32.5	45.4	28.19	497.6	460.6	40.2	40.2	0.00	4.5	0.0	13.2
50	8	3.0	3300	5620	4.0	4.0	7.31	18.2	30.6	40.39	5193.9	1731.1	27.9	28.2	1.20	60.4	3.9	11.4
50	8	4.5	3050	5480	4.4	4.4	6.34	13.4	25	46.09	1308.8	587.3	22.2	22.2	0.00	14.0	0.0	15
55	8	1.5	5005	7183	4.2	4.6	45.76	36.3	50.4	27.85	2364.3	473.9	44.8	45.2	0.88	21.1	4.8	10.4
55	8	3.0	4015	6776	4.6	4.6	8	20.2	44.2	53.35	6054.3	2261.8	28.9	31.8	8.77	42.9	17.2	15.8
55	8	4.5	3685	6644	4.4	4.4	7.54	14.3	33.2	56	5477.1	796.5	22.5	24.6	7.96	42.5	10.0	15.4
60	8	1.5	6000	8556	4.6	4.6	62.55	39.9	63.8	36.33	5776.5	2466.4	48.9	50.6	3.36	30.1	5.5	15.4
60	8	3.0	4800	8076	4.8	4.8	24.17	21.8	50.8	56.05	7200.0	3169.4	29.8	35.4	15.86	23.4	10.6	15.6
60	8	4.5	4380	7884	4.8	4.8	8.69	15.5	38	57.99	7200.0	3102.6	23.7	27.6	14.13	31.3	14.2	15.4

A major challenge is therefore to speed up the convergence of the lower bound towards the optimal solution. The next sections of this chapter are dedicated to finding answers to such challenge.

4.2 Relaxations

When solving an integer or combinatorial optimization problem $z = \min \{c^T x : x \in P \cap \mathbb{Z}^n\}$, testing whether or not a given solution x^* is the best possible solution can be quite challenging. A "naive" yet significant remark is that if one is able to find a lower bound $\underline{z} \leq z$ and an upper bound $\bar{z} \geq z$ so that $\underline{z} = c^T x^* = \bar{z}$, then a proof of the optimality of x^* is in hand.

Relaxations often play an important role in providing lower bounds (for a minimization problem) on the optimal value. In this section, we propose two "partial" linear relaxations which yield optimal solutions for the U-SNP. A performance comparison between these two relaxations is provided in Chapter 5. Finally, the "full" linear relaxation is studied, showing that the bounds it provides are particularly weak.

4.2.1 Relaxing the stop variables

It is easy to see that once variables x_e^i are fixed to 0-1 values, that is, once one knows which vehicle takes each demand, the problem turns out to be trivial. Let $\bar{x}_e^i$ represent such fixed-value variables. In fact, the value of variable y_v^i simply becomes 1 if there exists some demand $e \in E$ incident to v for which $\bar{x}_e^i = 1$. On the other hand, if such a demand does not exist, the value of variable y_v^i becomes 0. In other words,

$$y_v^i = \max\{\bar{x}_e^i : e \text{ is incident to } v\},$$

for each $v \in V$ and $i \in K$. Indeed, this is just a consequence of the Property 1 stated in Section 2.2, which claims that in an optimal solution, a vehicle should only stop where its clients ask for and nowhere else.

For this reason, integrality constraints (4.6) may be replaced by

$$0 \leq y_v^i \leq 1 \qquad \forall v \in V, i \in K, \tag{4.7}$$

and the integrality of variables y_v^i in any optimal solution is ensured by (4.1), (4.4) and (4.5). In fact, since variables y_v^i are bounded from below by the stop constraints

(4.4), one only needs the second part of inequalities (4.7), that is,

$$y_v^i \leq 1 \qquad \forall v \in V, i \in K. \tag{4.8}$$

Therefore, the following proposition holds.

Proposition 4.1 - Continuous stop variables for U-SNP
The relaxed formulation defined by (4.1)-(4.5), (4.8) *solves the U-SNP.* ■

Another way of achieving this result is to recognize that if vector $\bar{x} \in \{0,1\}^{m \times p}$ satisfies constraints (4.2) and (4.3), the remaining problem is

$$\min \sum_{v \in V} \sum_{i \in K} y_v^i \tag{4.9}$$

subject to

$$y_v^i \geq \bar{x}_e^i \qquad \forall i \in K, e \in E, v \in \{o_e, d_e\}, \tag{4.10}$$

$$y_v^i \in \{0,1\} \qquad \forall v \in V, i \in K. \tag{4.11}$$

However, the system defined by inequalities (4.8) and (4.10) is clearly total unimodular. Hence, the associated polytope

$$P_y = \{y \in \mathbb{R}^{n \times p} : y \text{ satisfies } (4.8), (4.10)\}$$

is integral. Notice that P_y can be seen as a face of the polyhedron induced by the linear relaxation of formulation (4.1)-(4.5), (4.8), that is,

$$P_y = \{x \in \mathbb{R}^{m \times p}, y \in \mathbb{R}^{n \times p} : (x,y) \text{ satisfies } (4.2)\text{–}(4.4), (4.8)\} \cap \{x \in \mathbb{R}^{m \times p} : x = \bar{x}\}.$$

Therefore, any extreme point (x^*, y^*) of polyhedron

$$\{x \in \mathbb{R}^{m \times p}, y \in \mathbb{R}^{n \times p} : (x,y) \text{ satisfies } (4.2)\text{–}(4.4), (4.8)\},$$

where x^* satisfies the integrality constraints (4.5) also verifies $y^* \in \{0,1\}^{n \times p}$, and hence Proposition 4.1 holds.

4.2.2 Relaxing assignment variables

The reverse question, that is, deciding whether or not one can deduce an optimal assignment of demands to vehicles given where each vehicle stops is less evident. Such question may be answered by showing that the associated polytope is integral when variables y_v^i are fixed to 0-1 values.

In order to achieve such result, consider a vector $\bar{y} \in \{0,1\}^{n \times p}$ that represents the fixed values of the variables y_v^i. Then inequalities (4.4) become

$$x_e^i \leq \bar{y}_v^i \qquad \forall i \in K, e \in E, v \in \{o_e, d_e\}. \tag{4.12}$$

Moreover, let integrality constraints (4.5) be replaced by

$$0 \leq x_e^i \leq 1 \qquad \forall e \in E, i \in K. \tag{4.13}$$

In fact, since variables x_e^i are bounded from above by the stop constraints (4.4), one only needs the first part of inequalities (4.13), that is,

$$x_e^i \geq 0 \qquad \forall e \in E, i \in K. \tag{4.14}$$

Finally, consider the polytope $P_x(V, E, C, \bar{y})$ associated with U-SNP when variables y_v^i are fixed to 0-1 values (*i.e.*, $y_v^i = \bar{y}_v^i$), defined as

$$P_x(V, E, C, \bar{y}) = \{x \in \mathbb{R}^{m \times p} : x \text{ satisfies } (4.2), (4.3), (4.12), (4.14)\}.$$

If vector $\bar{y}$ is known and $P_x(V, E, C, \bar{y})$ is integral, then an integer solution to U-SNP can be obtained in polynomial time through the ellipsoid or interior-point methods (see Khachiyan [104], Karmarkar [102]), and hence the relaxed formulation defined by (4.1)-(4.4), (4.6), (4.14) is capable of solving the U-SNP.

We next show that $P_x(V, E, C, \bar{y})$ is integral for the restricted case where (V, E, C) is an instance of Intersection U-SNP but it is not for the general U-SNP.

Theorem 4.1 - Integrality of $P_x(V, E, C, \bar{y})$ for Intersection U-SNP
Given an instance (V, E, C) of Intersection U-SNP and a vector $\bar{y} \in \{0,1\}^{n \times p}$, the polytope $P_x(V, E, C, \bar{y})$ is integral.

Proof. From Property 3 stated in Section 2.4, capacity constraints (4.3) can be replaced by

$$\sum_{e \in E} x_e^i \leq C \qquad \forall \quad i \in K. \tag{4.15}$$

Hence, the matrix A from the system of inequalities $Ax \leq b$ defined by (4.2), (4.15) has the form

$$A = \begin{bmatrix} I_1 & I_2 & \cdots & I_p \\ \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1} \end{bmatrix}$$

where I_i (for $1 \leq i \leq p$) is a $m \times m$ identity matrix, $\mathbf{1}$ is a m-dimensional unit row-vector and $\mathbf{0}$ is a m-dimensional zero row-vector.

Matrix A has $2m$ rows and m^2 columns (recall that $p = m$), each containing exactly 2 non-zero entries. Let G be the graph obtained from the matrix A such that G has a vertex for each row of A and an edge $e = uv$ for each column such that $A_{u,e} = A_{v,e} = 1$. It follows that G is a complete bipartite graph $K_{m,m}$. Hence, A is totally unimodular (*cf.* Theorem 1.3 and Theorem 1.4) and thus $P_x(V, E, C, \bar{y})$ is integral. ■

The total unimodularity of the system (4.2), (4.3), (4.12), (4.14) holds when (V, E, C) is an instance of Intersection U-SNP. However, it is not preserved in the general case. The following instance serves as a counter-example.

Example 4.1 - Total unimodularity is not present in general U-SNP
Consider the following instance of U-SNP:

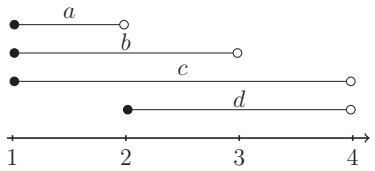


Figure 4.2: An instance of U-SNP

Given the instance illustrated above, the matrix A from the system of inequalities $Ax \leq b$ defined by (4.2), (4.3) is

$$A = \begin{bmatrix} I & I & I & I \\ M & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & M & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & M & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & M \end{bmatrix}, \text{ where } M = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

and I is a 4×4 identity matrix.

Let A' be the sub-matrix of A defined by rows 2,4,5,7,9 and columns 2,3,4,6,8, that is,

$$A' = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Notice that $det(A') = -2$, which proves A is not totally unimodular. □

For this reason, the argument used for proving Theorem 4.1 cannot be applied if one wishes to extend such result to the general U-SNP. It turns out that if (V, E, C) is an instance of the general U-SNP, then $P_x(V, E, C, \bar{y})$ may indeed contain a fractional extreme point, that is, it may not be an integral polytope. For proving such statement let $(V, E, 1)$ be the instance depicted in Figure 4.2 with $C = 1$. If $\bar{y} = \mathbf{1}$, then the following fractional solution $\hat{x}$ belongs to $P_x(V, E, 1, \mathbf{1})$.

$$\begin{array}{llll}
\hat{x}^1_e = \frac{1}{2} & \text{for } e \in \{a, c, d\}, & \hat{x}^1_e = 0 & \text{for } e \in \{b\}, \\
\hat{x}^2_e = \frac{1}{2} & \text{for } e \in \{a, b\}, & \hat{x}^2_e = 0 & \text{for } e \in \{c, d\}, \\
\hat{x}^3_e = \frac{1}{2} & \text{for } e \in \{b, d\}, & \hat{x}^3_e = 0 & \text{for } e \in \{a, c\}, \\
\hat{x}^4_e = \frac{1}{2} & \text{for } e \in \{c\}, & \hat{x}^4_e = 0 & \text{for } e \in \{a, b, d\}.
\end{array}$$

For proving that $\hat{x}$ is indeed an extreme point of $P_x(V, E, 1, \mathbf{1})$, it suffices to show that $\hat{x}$ is the unique solution present in the face induced by the inequalities defining $P_x(V, E, 1, \mathbf{1})$ that are tight for $\hat{x}$.

Notice that the capacity constraints

$$\sum_{e \in \Delta_E(v)} x^i_e \leq C \qquad \forall i \in K, v \in V,$$

are tight for vehicle 1 and stations 1 and 2, for vehicle 2 and station 1, and for vehicle 3 and station 2. Moreover, inequalities

$$x^i_e \geq 0 \qquad \forall e \in E, i \in K,$$

are obviously tight for every zero component of $\hat{x}$. Considering such tight inequalities together with the assignment constraints

$$\sum_{i \in K} x^i_e = 1 \qquad \forall e \in E,$$

one obtains the following system

$$\begin{array}{cccccccccccccccc}
x^1_a & + & x^1_c & & & & & & & & & & & & & = & 1 \\
 & & x^1_c & + & x^1_d & & & & & & & & & & & = & 1 \\
 & & & & & & x^2_a & + & x^2_b & & & & & & & = & 1 \\
 & & & & & & & & & & x^3_b & + & x^3_d & & = & 1 \\
x^1_a & & & & & + & x^2_a & & & & & & & & & = & 1 \\
 & & & & & & & & x^2_b & + & x^3_b & & & & & = & 1 \\
 & & & & x^1_d & & & & & & & & x^3_d & & & = & 1 \\
 & & x^1_c & & & & & & & & & & & + & x^4_c & = & 1
\end{array}$$

for which the unique solution is

$$x_a^1 = x_c^1 = x_d^1 = x_a^2 = x_b^2 = x_b^3 = x_d^3 = x_c^4 = \tfrac{1}{2}. \tag{4.16}$$

It follows directly from (4.16) that $\hat{x}$ is indeed an extreme point of $P_x(V, E, 1, \mathbf{1})$.

4.2.3 Linear relaxation

The strength of a formulation can be measured by the dual bounds (lower bounds for a minimization problem) its linear relaxation is capable of providing. The better the bounds are, the stronger the formulation is. In Wolsey [181], the primal integrality gap is used as an indicator of how strong a given formulation is.

According to this concept, the relaxed formulations previously presented are equally strong as their linear relaxations are identical. Of course, there might still exist some differences on the performances of a branch-and-cut framework based on each of them (as we shall see in Chapter 5) due to the different branching decisions each formulation induces.

Next, we study the strength of formulation (4.1)-(4.6) by analyzing its linear relaxation. For this, let $P_{(V,E,C)}$ denote the polyhedron composed by inequalities (4.2)-(4.4), (4.8), (4.14), that is,

$$P_{(V,E,C)} = \{x \in \mathbb{R}^{m\times p}, y \in \mathbb{R}^{n\times p} : (x, y) \text{ satisfies } (4.2) - (4.4), (4.8), (4.14)\}.$$

Theorem 4.2 - Formulation's strength
The linear relaxation, defined by $\min\{\mathbf{0}x + \mathbf{1}y : (x, y) \in P_{(V,E,C)}\}$*, provides a lower bound of n stops.*

Proof. If one can find a primal and a dual solution of same objective-function value, then the strong duality theorem states that both solutions are optimal solutions for the respective problems. Next, given an instance (V, E, C) of U-SNP, we construct a primal and a dual solution having same cost equal to $|V| = n$.

The following dual formulation is defined by associating a variable α_e with each inequality in (4.2), a variable β_v^i with each inequality in (4.3), a variable γ_e^i with each inequality in (4.4) where $v = o_e$, and a variable μ_e^i with each inequality in (4.4) where $v = d_e$.

$$z_{dual} = \max \sum_{e\in E} \alpha_e - \sum_{v\in V} \sum_{i\in K} C\beta_v^i \tag{4.17}$$

subject to

$$\alpha_e - \sum_{v=o_e}^{d_e-1} \beta_v^i - \gamma_e^i - \mu_e^i \leq 0 \qquad \forall e \in E, i \in K \qquad (4.18)$$

$$\sum_{e \in \delta_G^+(v)} \gamma_e^i + \sum_{e \in \delta_G^-(v)} \mu_e^i \leq 1 \qquad \forall i \in K, v \in V, \qquad (4.19)$$

$$\beta_v^i \geq 0 \qquad \forall v \in V, i \in K, \qquad (4.20)$$

$$\gamma_e^i \geq 0 \qquad \forall e \in E, i \in K, \qquad (4.21)$$

$$\mu_e^i \geq 0 \qquad \forall e \in E, i \in K. \qquad (4.22)$$

We construct a dual feasible solution $(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma}, \widehat{\mu})$ as follows. Let $\widehat{\beta}_v^i = 0$ for all $v \in V, i \in K$. For each $v \in V$, let e' be an arbitrarily chosen edge incident to v, that is, $e' \in \delta_G^+(v) \cup \delta_G^-(v)$. If $e' \in \delta_G^+(v)$, let $\widehat{\gamma}_{e'}^i = 1$ for all $i \in K$, and otherwise, let $\widehat{\mu}_{e'}^i = 1$ for all $i \in K$. Let every other variable $\widehat{\gamma}$ and $\widehat{\mu}$ be 0. Inequalities (4.19) are thus satisfied at equality. Finally, let $\widehat{\alpha}_e = \widehat{\gamma}_e^i + \widehat{\mu}_e^i$ for all $e \in E, i \in K$. Inequalities (4.18) are thus also satisfied at equality and $(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma}, \widehat{\mu})$ is a feasible solution of the dual problem. By construction,

$$\sum_{e \in E} \widehat{\alpha}_e - \sum_{v \in V} \sum_{i \in K} C \widehat{\beta}_v^i = |V| = n.$$

We now give a primal feasible solution with the same cost to prove optimality. Let $\widehat{x}_e^i = \frac{1}{p}$ for all $e \in E, i \in K$, and $\widehat{y}_v^i = \frac{1}{p}$ for all $v \in V, i \in K$. Equations (4.2) are satisfied since $\sum_{i \in K} \frac{1}{p} = 1$. Inequalities (4.3) are satisfied since $\sum_{e \in \Delta_E(v)} \frac{1}{p} = \frac{|\Delta_E(v)|}{p} \leq C$. Inequalities (4.4) are clearly satisfied since $0 \leq \frac{1}{p} \leq \frac{1}{p}$. Therefore, $(\widehat{x}, \widehat{y})$ is a feasible solution of the primal problem. By construction,

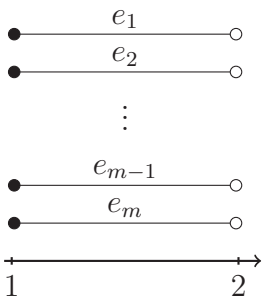
$$\sum_{v \in V} \sum_{i \in K} \widehat{y}_v^i = |V| = n.$$

From strong duality theorem, $(\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma}, \widehat{\mu})$ and $(\widehat{x}, \widehat{y})$ are both optimal solutions from the dual and primal problems, respectively. ■

Notice however, that a lower bound of n stops does not deliver any additional information since such bound is already known from Proposition 3.1. This would not be a problem if n was indeed a good lower bound. Unfortunately, even for the simplest instances, n is far from being a good lower bound and the integrality gap is quite large. The following example illustrates the situation.

Example 4.2 - How weak is the given formulation

Consider the following instance of U-SNP with m demands and capacity C.



Clearly, each non-empty vehicle must stop exactly 2 times. Therefore, an optimal solution is obtained using the minimum number of vehicles. Since $k_{min} = \left\lceil \frac{m}{C} \right\rceil$, the optimal integer solution has a cost of $2 \left\lceil \frac{m}{C} \right\rceil$ stops. Theorem 4.2 indicates, however, that the linear relaxation provides a lower bound of 2 stops for the given instance. Therefore, the integrality gap of the given formulation can reach up to

$$1 - \frac{1}{\left\lceil \frac{m}{C} \right\rceil}.$$ □

This means that if one sets $m = 100$ and $C = 1$ on the given example, the integrality gap equals 99%! Moreover, an asymptotic analysis (see Figure 4.3), that is setting $m \to \infty$, yields a remarkable – in a bad way – integrality gap of 100%. Recall, in addition, that the case where $C = 1$ can be solved in polynomial time. Such observations reveal that there exists a large room for improvements in the formulation (4.1)-(4.6) provided by Pimenta et al. [151].

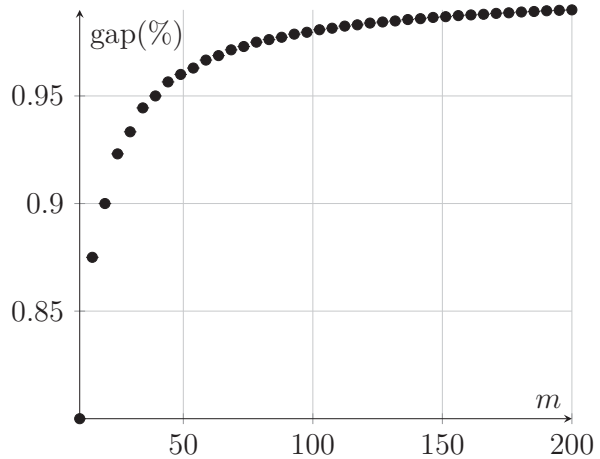


Figure 4.3: Asymptotic analysis of the integrality gap

4.3 Dimension and facial study

In the previous section, the lower bound provided by the linear relaxation of formulation (4.1)-(4.6) was shown to be particularly poor. Such result provides evidences

on why typical Branch-and-Bound approaches have failed on solving such formulation (*cf.* Pimenta et al. [151]). In order to reinforce the given formulation, a natural approach is therefore to investigate the facial structure of the convex hull of the feasible solutions. Recall that for the U-SNP, such convex hull is given by the polytope $\mathcal{P}_{(V,E,C)}$, defined as

$$\mathcal{P}_{(V,E,C)} = \text{conv}(P_{(V,E,C)} \cap \mathbb{Z}^{(n+m)p}),$$

where $P_{(V,E,C)}$ is the polyhedron composed by inequalities (4.2)-(4.4), (4.8), (4.14).

In this section we first investigate the dimension of $\mathcal{P}_{(V,E,C)}$. We then study the inequalities composing $P_{(V,E,C)}$ and identify necessary and sufficient conditions for which they are facet-defining for $\mathcal{P}_{(V,E,C)}$. However, before diving into such study, some additional definitions are made necessary for an easier lecture.

Definition 4.1 - Perfect idleness
*Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP and a solution $(\bar{x}, \bar{y}) \in \mathcal{P}_{(V,E,C)}$, vehicle $i \in K$ is said to be **perfectly idle** if $\bar{x}^i_e = 0$ for every $e \in E$ and $\bar{y}^i_v = 0$ for every $v \in V$. In other words, vehicle i is perfectly idle if it is idle and it does not stop at any station.*

Definition 4.2 - Dumbness
*Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP and a solution $(\bar{x}, \bar{y}) \in \mathcal{P}_{(V,E,C)}$, vehicle $i \in K$ is said to be **dumb** if $\bar{y}^i_v = 1$ for every $v \in V$, that is, vehicle i stops at all stations.*

Definition 4.3 - Perfect dumbness
*Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP and a solution $(\bar{x}, \bar{y}) \in \mathcal{P}_{(V,E,C)}$, vehicle $i \in K$ is said to be **perfectly dumb** if $\bar{x}^i_e = 0$ for every $e \in E$ and $\bar{y}^i_v = 1$ for every $v \in V$. In other words, vehicle i is perfectly dumb if it is idle and dumb.*

The next definitions concern some operations that can be done once a feasible solution of U-SNP is known, in order to derive another solution. Notice that depending on the *input* solution, some of these operations might result in unfeasible solutions. Nonetheless, throughout this chapter we are cautious enough to invoke such operations only on suitable situations.

Definition 4.4 - Transfer
Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP and a solution $(\bar{x}, \bar{y}) \in \mathcal{P}_{(V,E,C)}$, the solution $(\hat{x}, \hat{y}) \in \mathcal{P}_{(V,E,C)}$ is said to be constructed by transferring a chosen demand e' from vehicle k to vehicle j, if

$$
\begin{aligned}
&\hat{x}^j_{e'} = \bar{x}^j_{e'} + \bar{x}^k_{e'}, && \\
&\hat{x}^k_{e'} = 0, && \\
&\hat{x}^i_{e'} = \bar{x}^i_{e'} && \forall i \in K \setminus \{j,k\}, \\
&\hat{x}^i_{e} = \bar{x}^i_{e} && \forall i \in K, e \in E \setminus e', \\
&\hat{y}^j_{o_{e'}} = \bar{y}^j_{o_{e'}} + \bar{y}^k_{o_{e'}}, && \\
&\hat{y}^j_{d_{e'}} = \bar{y}^j_{d_{e'}} + \bar{y}^k_{d_{e'}}, && \\
&\hat{y}^j_{v} = \bar{y}^j_{v} && \forall v \in V \setminus \{o_{e'}, d_{e'}\}, \\
&\hat{y}^i_{v} = \bar{y}^i_{v} && \forall v \in V, i \in K \setminus j.
\end{aligned}
$$

Definition 4.5 - Sequential permutation
Given an instance $\mathcal{I} = (V, E, C)$ of U-SNP, a solution $(\bar{x}, \bar{y}) \in \mathcal{P}_{(V,E,C)}$ and a subset of vehicles $S = \{s_1, \ldots, s_{|S|}\} \subseteq K$, the solution $(\hat{x}, \hat{y}) \in \mathcal{P}_{(V,E,C)}$ is said to be a sequential permutation of $(\bar{x}, \bar{y})$ over S if

$$
\begin{aligned}
&\hat{x}^{s_i}_{e} = \bar{x}^{s_{i+1}}_{e} && \forall i \in \{1, \ldots, |S|-1\}, e \in E, \\
&\hat{x}^{s_{|S|}}_{e} = \bar{x}^{s_1}_{e} && \forall e \in E, \\
&\hat{x}^{i}_{e} = \bar{x}^{i}_{e} && \forall i \in K \setminus S, e \in E, \\
&\hat{y}^{s_i}_{v} = \bar{y}^{s_{i+1}}_{v} && \forall i \in \{1, \ldots, |S|-1\}, v \in V, \\
&\hat{y}^{s_{|S|}}_{v} = \bar{y}^{s_1}_{v} && \forall e \in E, \\
&\hat{y}^{i}_{v} = \bar{y}^{i}_{v} && \forall i \in K \setminus S, v \in V.
\end{aligned}
$$

Notice that for $S = K = \{1, \ldots, p\}$, a solution $\{\hat{E}_1, \ldots, \hat{E}_p\}$ is said to be a sequential permutation of solution $\{E_1, \ldots, E_p\}$ if

$$\{\hat{E}_1, \ldots, \hat{E}_p\} = \{E_2, \ldots, E_p, E_1\}.$$

4.3.1 Dimension

The dimension of a polytope $\mathcal{P} \subseteq \mathbb{R}^d$, denoted by $\dim(\mathcal{P})$, is defined as the maximum number of affinely independent vectors in $\mathcal{P}$ minus one. When $\mathcal{P}$ is full dimensional, that is $\dim(\mathcal{P}) = d$, searching for $d+1$ affinely independent vectors in $\mathcal{P}$ is usually a fairly manageable task. Nonetheless, gathering such vectors for polytopes that are not full dimensional can be rather tricky.

For this reason, let us use an alternative definition of $\dim(\mathcal{P})$: given a polyhedron $\mathcal{P} = \{x \in \mathbb{R}^n : Ax \leq b\}$ where $A^{=}x \leq b^{=}$ is the subsystem of implicit equalities in $Ax \leq b$, the dimension of $\mathcal{P}$ is equal to n minus the rank of $A^{=}$. Next, we show that $\dim(\mathcal{P}_{(V,E,C)}) = (n+m)p - m$ by showing that the only implicit equalities in $\mathcal{P}_{(V,E,C)}$ are those described by constraints (4.2) (up to a linear combination of them).

Theorem 4.3 - No additional implicit equalities in $\mathcal{P}_{(V,E,C)}$
Let $\alpha^T x+\beta^T y = \gamma$ define a general hyperplane in $\mathbb{R}^{(n+m)p}$. If $C \geq 2$ and $\alpha^T x+\beta^T y = \gamma$ is an implicit equality in $\mathcal{P}_{(V,E,C)}$, then it can be written as a linear combination of equations (4.2).

Proof. By definition, if $\alpha^T x + \beta^T y = \gamma$ is an implicit equality in $\mathcal{P}_{(V,E,C)}$, then

$$\mathcal{P}_{(V,E,C)} \subseteq \{(x, y) \in \mathbb{R}^{(n+m)p} : \alpha^T x + \beta^T y = \gamma\}.$$

The proof is organized in three steps:

i. showing $\beta_v^i = 0$ for any station $v \in V$ and any vehicle $i \in K$;

ii. showing $\alpha_e^i = \alpha_e^j$ for any demand $e \in E$ and any pair of vehicles $i \in K, j \in K$;

iii. showing $\gamma = \sum_{e\in E} \alpha_e^i$ for any vehicle $i \in K$.

Let $(\bar{x}, \bar{y})$ be a feasible solution where some arbitrarily chosen vehicle $j \in K$ is perfectly idle. Such a solution exists since Property 4 stated in Section 3.1 holds.

Step i.: Let us construct another feasible solution $(\hat{x}, \hat{y})$ in the following manner.

$$\begin{aligned}
&\hat{x}_e^i = \bar{x}_e^i \quad \forall e \in E, i \in K,\\
&\hat{y}_v^i = \bar{y}_v^i \quad \forall v \in V, i \in K \setminus j,\\
&\hat{y}_v^j = \bar{y}_v^j \quad \forall v \in V \setminus v',\\
&\hat{y}_{v'}^j = 1
\end{aligned}$$

for some arbitrarily chosen $v' \in V$.

Both solutions are feasible and thus satisfy the equation $\alpha^T x + \beta^T y = \gamma$. Therefore, $\beta_{v'}^j = 0$. Since $v' \in V$ and $j \in K$ can be arbitrarily chosen,

$$\beta_v^i = 0 \qquad \forall v \in V, i \in K,$$

and the supposed implicit equality can be rewritten as

$$\sum_{e\in E} \sum_{i\in K} \alpha_e^i x_e^i = \gamma.$$

Step ii.: Consider again solution $(\bar{x}, \bar{y})$. Let e' denote some arbitrarily chosen demand that is assigned to some vehicle $i \in K$. Now construct another feasible solution $(\hat{x}, \hat{y})$ by transferring demand e' from vehicle i to vehicle j. Such construction is possible since vehicle j is idle.

Both solutions are feasible and thus satisfy the equation $\alpha^T x = \gamma$. Therefore $\alpha^i_{e'} = \alpha^j_{e'}$. Since vehicle $j \in K$ and demand $e' \in E$ can be arbitrarily chosen,

$$\alpha^i_{e'} = \alpha^j_{e'} \qquad \forall e \in E, i \in K, j \in K,$$

and the supposed implicit equality can be rewritten as

$$\sum_{e \in E} \sum_{i \in K} \lambda_e x^i_e = \gamma, \tag{4.23}$$

where $\lambda_e = \alpha^i_e$ for any $e \in E, i \in K$.

Step iii.: Finally, take solution $(\bar{x}, \bar{y})$ and construct other $p-1$ feasible solutions by taking $p-1$ times the sequential permutation of $(\bar{x}, \bar{y})$. Notice that, considering this pool of p feasible solutions, each demand is assigned to a vehicle $i \in K$ exactly once. Therefore, all these p solutions satisfy the equation (4.23) and summing up all such equations, one gets

$$\sum_{e \in E} p\lambda_e = p\gamma,$$

which reduces to

$$\sum_{e \in E} \lambda_e = \gamma.$$

Step iii. is thus finished and the supposed implicit equality can be rewritten as

$$\sum_{e \in E} \sum_{i \in K} \lambda_e x^i_e = \sum_{e \in E} \lambda_e,$$

which is clearly a linear combination of equations (4.2) with weights $\lambda \in \mathbb{R}^m$. ■

Notice that the system of equations defined by constraints (4.2) clearly has full-rank and hence

$$\text{rank}(A^=) = m,$$

which yields the following result.

Corollary 4.1 - Dimension of $\mathcal{P}_{(V,E,C)}$
$\dim\left(\mathcal{P}_{(V,E,C)}\right) = (n+m)p - m.$ ■

Notice that this result only remains true under the condition that Property 4 holds, that is, if $p \geq p_{min} + 1$. For the case where $p = p_{min}$, one may have additional implicit equalities other than (4.2). For the sake of scientific curiosity, we take a quick glimpse at this particular case through Propositions 4.2 and 4.3. For the rest of our polyhedral study, however, we assume that $C \geq 2$ (as otherwise, the problem can be easily solved) and that $p = m > p_{min}$ and therefore Property 4 holds.

Proposition 4.2 - Implicit equalities when $p = p_{min}$
If $p = p_{min}$ and there exists a station $v' \in V$ for which $|\Delta_E(v')| = pC$, then inequalities (4.3) *associated with v' are implicit equalities for each $i \in K$.*

Proof. From the definition of p_{min}, if $p = p_{min}$, then there exists at least one feasible solution. By definition, a single vehicle can take at most C demands in $\Delta_E(v')$. Since each demand must be assigned to some vehicle, every vehicle must take exactly C demands in $\Delta_E(v')$. Therefore,

$$\sum_{e \in \Delta_E(v')} x_e^i = C \quad \forall i \in K. \qquad \blacksquare$$

Proposition 4.3 - Further implicit equalities when $p = p_{min}$
If $p = p_{min}$ and there exists a station $v' \in V$ for which

$$\left\lceil \frac{\max\{|\delta_G^-(v')|, |\delta_G^+(v')|\}}{C} \right\rceil = p,$$

then inequalities (4.8) *associated with v' are implicit equalities for each $i \in K$.*

Proof. Without loss of generality, suppose $|\delta_G^+(v')| \geq |\delta_G^-(v')|$. Notice that $\delta_G^+(v') \subseteq \Delta_E(v')$. Hence, the maximum number of demands in $\delta_G^+(v')$ a single vehicle can take is C. Since each demand must be assigned to some vehicle, one needs at least $\left\lceil \frac{|\delta_G^+(v')|}{C} \right\rceil$ vehicles to satisfy such demands. This means that if $p = \left\lceil \frac{|\delta_G^+(v')|}{C} \right\rceil$, then all vehicles must take at least one demand in $\delta_G^+(v')$. Since each demand in $\delta_G^+(v')$ stops at v', every vehicle must stop at v'. Therefore,

$$y_{v'}^i = 1 \quad \forall i \in K. \qquad \blacksquare$$

4.3.2 Facial study

Before tackling the facial study of $\mathcal{P}_{(V,E,C)}$, let us introduce a few lemmas based on the ideas used in the proof of Theorem 4.3. These lemmas will be useful and referenced multiple times in future proofs.

Let

$$\alpha^T x + \beta^T y \leq \gamma \tag{4.24}$$

be a valid inequality for $\mathcal{P}_{(V,E,C)}$, that is,

$$\mathcal{P}_{(V,E,C)} \subseteq \{(x, y) \in \mathbb{R}^{(n+m)p} : \alpha^T x + \beta^T y \leq \gamma\}.$$

Moreover, let

$$F' = \left\{(x, y) \in \mathcal{P}_{(V,E,C)} : \alpha^T x + \beta^T y = \gamma\right\}$$

denote the face of $\mathcal{P}_{(V,E,C)}$ induced by (4.24).

Lemma 4.1. *Given an instance (V, E, C) of U-SNP, a station $v' \in V$ and a vehicle $i' \in K$, let $(\bar{x}, \bar{y}) \in USNP_{(V,E,C)}$ be a feasible solution for (V, E, C) where $\bar{y}^{i'}_{v'} = 0$, and let $(\hat{x}, \hat{y}) \in USNP_{(V,E,C)}$ be a feasible solution constructed in the following manner.*

$$\begin{aligned}
&\hat{x}^i_e = \bar{x}^i_e \quad \forall e \in E, i \in K,\\
&\hat{y}^i_v = \bar{y}^i_v \quad \forall v \in V, i \in K \setminus i',\\
&\hat{y}^{i'}_v = \bar{y}^{i'}_v \quad \forall v \in V \setminus v',\\
&\hat{y}^{i'}_{v'} = 1.
\end{aligned}$$

If both $(\bar{x}, \bar{y})$ and $(\hat{x}, \hat{y})$ belong to the same face

$$F' = \left\{(x, y) \in \mathcal{P}_{(V,E,C)} : \alpha^T x + \beta^T y = \gamma\right\},$$

then $\beta^{i'}_{v'} = 0$.

Proof. If $(\bar{x}, \bar{y}) \in F'$, the following equality holds.

$$\sum_{e\in E}\sum_{i\in K} \alpha^i_e \bar{x}^i_e + \sum_{v\in V}\sum_{i\in K} \beta^i_v \bar{y}^i_v = \gamma. \tag{4.25}$$

If $(\hat{x}, \hat{y}) \in F'$, the following equality holds.

$$\sum_{e\in E}\sum_{i\in K} \alpha^i_e \bar{x}^i_e + \sum_{v\in V}\sum_{i\in K} \beta^i_v \bar{y}^i_v + \beta^{i'}_{v'} = \gamma. \tag{4.26}$$

By subtracting (4.25) from (4.26), one obtains $\beta^{i'}_{v'} = 0$. ■

Lemma 4.2. *Given an instance (V, E, C) of U-SNP, a demand $e' \in E$ and vehicle $i' \in K$, let $(\bar{x}, \bar{y}) \in USNP_{(V,E,C)}$ be a feasible solution for (V, E, C) where vehicle i' is perfectly idle, and let $(\hat{x}, \hat{y}) \in USNP_{(V,E,C)}$ be a feasible solution constructed by transferring demand e' that was assigned to some vehicle $k \in K \setminus i'$ to vehicle i'.*

If both $(\bar{x}, \bar{y})$ and $(\hat{x}, \hat{y})$ belong to the same face

$$F' = \left\{(x, y) \in \mathcal{P}_{(V,E,C)} : \alpha^T x + \beta^T y = \gamma\right\},$$

then $\alpha^{i'}_{e'} + \beta^{i'}_{o_{e'}} + \beta^{i'}_{d_{e'}} = \alpha^k_{e'}$.

Proof. If $(\bar{x}, \bar{y}) \in F'$, the following equality holds:

$$\sum_{e \in E} \sum_{i \in K} \alpha_e^i \bar{x}_e^i + \sum_{v \in V} \sum_{i \in K} \beta_v^i \bar{y}_v^i = \gamma. \tag{4.27}$$

If $(\hat{x}, \hat{y}) \in F'$, the following equality holds:

$$\sum_{e \in E} \sum_{i \in K} \alpha_e^i \bar{x}_e^i - \alpha_{e'}^k + \alpha_{e'}^{i'} + \sum_{v \in V} \sum_{i \in K} \beta_v^i \bar{y}_v^i + \beta_{o_{e'}}^{i'} + \beta_{d_{e'}}^{i'} = \gamma. \tag{4.28}$$

By subtracting (4.27) from (4.28), one obtains $\alpha_{e'}^{i'} + \beta_{o_{e'}}^{i'} + \beta_{d_{e'}}^{i'} = \alpha_{e'}^k$. ■

We are now ready to give the necessary and sufficient conditions for which the inequalities composing $P_{(V,E,C)}$ induce facets of $\mathcal{P}_{(V,E,C)}$. Let us begin by analysing the trivial inequalities (4.8) and (4.14).

Theorem 4.4 - Inequalities (4.8) **defining facets**
Inequalities

$$y_v^i \leq 1 \qquad \forall v \in V, i \in K, \tag{4.8}$$

are facet-defining for $\mathcal{P}_{(V,E,C)}$.

Proof. Let v' be any station in V and i' be any vehicle in K. We show that face F, defined as

$$F = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : y_{v'}^{i'} = 1 \right\},$$

is a maximal face of $\mathcal{P}_{(V,E,C)}$, and therefore inequalities (4.8) are facet-defining. More specifically, we show that if there exists some face F' of $\mathcal{P}_{(V,E,C)}$, defined as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \alpha_e^i x_e^i + \sum_{v \in V} \sum_{i \in K} \beta_v^i y_v^i = \gamma \right\},$$

for which $F \subseteq F'$, then F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \omega y_{v'}^{i'} = \sum_{e \in E} \lambda_e + \omega \right\},$$

that is, a linear combination of (4.8) itself and the implicit equalities (4.2), with weights $\omega \in \mathbb{R}$ and $\lambda \in \mathbb{R}^m$, respectively. As a consequence, one has that $F = F'$.

The proof is organized in the following three steps:

i. showing $\beta_v^i = 0$ for any $v \in V, i \in K \setminus i'$ and $\beta_v^{i'} = 0$ for any $v \in V \setminus v'$;

ii. showing $\alpha_e^j = \alpha_e^{i'}$ for any $e \in E, j \in K \setminus i'$;

iii. showing $\gamma = \sum_{e \in E} \alpha_e^i + \omega$ for any $i \in K$.

Step i.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is idle and stops only at v' (*i.e.*, $\bar{y}_{v'}^{i'} = 1$ and $\bar{y}_v^{i'} = 0$ for every $v \in V \setminus v'$). By definition, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.1 applies for any chosen station $v \in V \setminus v'$. Therefore, $\beta_v^{i'} = 0$ for any $v \in V \setminus v'$.

Consider now that $(\bar{x}, \bar{y})$ is a feasible solution where some arbitrarily chosen vehicle $j \in K \setminus i'$ is perfectly idle, and vehicle i' is dumb. By definition, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.1 applies for any chosen station $v \in V$, and therefore $\beta_v^j = 0$ for any $v \in V$. Since vehicle j can be arbitrarily chosen (as long as $j \neq i'$), $\beta_v^j = 0$ for any $v \in V, j \in K \setminus i'$.

Let $\omega \in \mathbb{R}$ be some real number such that $\omega = \beta_{v'}^{i'}$. Then, face F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \alpha_e^i x_e^i + \omega y_{v'}^{i'} = \gamma \right\}.$$

Step ii.: Let $(\bar{x}, \bar{y})$ be a feasible solution where some arbitrarily chosen vehicle $j \in K \setminus i'$ is perfectly idle and vehicle i' is dumb. By definition, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.2 applies for vehicle j and any chosen demand $e \in E$. Notice that a demand $e \in E$ can be assigned to any vehicle $k \in K \setminus j$, by taking the necessary number of sequential permutations of $(\bar{x}, \bar{y})$ over $K \setminus j$. Since all such solutions belong to F, we have that $\alpha_e^j = \alpha_e^k$ for any $e \in E, k \in K$.

Let $\lambda \in \mathbb{R}^m$ be some real vector such that $\lambda_e = \alpha_e^j$ for any $e \in E$. Face F' can now be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \omega x_{e'}^{i'} = \gamma \right\}.$$

Step iii.: Let $(\bar{x}, \bar{y})$ be a feasible solution where all vehicles are dumb. By definition, $(\bar{x}, \bar{y}) \in F$. Let us construct other $p - 1$ solutions by taking $p - 1$ times a sequential permutation of $(\bar{x}, \bar{y})$ over K. Since vehicle i' stops at station v' in all such solutions, each constructed solution belongs to F and therefore satisfies

$$\sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \omega = \gamma. \tag{4.29}$$

Notice that, considering this pool of p feasible solutions (the $p - 1$ constructed solutions plus $(\bar{x}, \bar{y})$), each demand is assigned to a vehicle $i \in K$ exactly once. Summing up the p equations (4.29) associated with the pool of solutions considered,

one obtains

$$p \sum_{e \in E} \lambda_e + p\omega = p\gamma.$$

Therefore, $\sum_{e \in E} \lambda_e + \omega = \gamma$ and face F' can finally be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \omega y_{v'}^{i'} = \sum_{e \in E} \lambda_e + \omega \right\}. \qquad \blacksquare$$

It is interesting to notice that even if constraints (4.8) are facet-defining, they are not significant from an optimization point of view. Such statement might seem odd and even contradictory. However, it turns out that U-SNP has a very precise objective function: minimizing the number of stops. For this reason, one may remove constraints (4.8) from $P_{(V,E,C)}$, and an optimal solution will still satisfy such constraints. Proposition 4.4 indicates such fact.

Proposition 4.4 - Constraints (4.8) **are not necessary**
Let $P'_{(V,E,C)}$ be the polyhedron defined by inequalities (4.2)-(4.4), (4.14). *If $(\widehat{x}, \widehat{y})$ is an optimal solution of the formulation defined by $P'_{(V,E,C)}$ under the objective function* (4.1), *then constraints* (4.8) *are satisfied by $(\widehat{x}, \widehat{y})$. That is,*

$$\widehat{y}_v^i \leq 1 \qquad \forall v \in V, i \in K.$$

Proof. Suppose $(\widehat{x}, \widehat{y})$ is an optimal solution not respecting inequality (4.8) for a given station v' and vehicle i'. Then, $\widehat{y}_{v'}^{i'} = 1 + \epsilon > 1$, for some $\epsilon > 0$.

From constraints (4.2) and (4.14),

$$0 \leq \widehat{x}_e^i \leq 1 \qquad \forall e \in E, i \in K.$$

Therefore, one can construct a solution $(\bar{x}, \bar{y})$ in the following manner:

$$\begin{aligned} &\bar{x}_e^i := \widehat{x}_e^i && \forall e \in E, i \in K, \\ &\bar{y}_v^i := \widehat{y}_v^i && \forall v \in V \setminus v', i \in K \setminus i', \\ &\bar{y}_{v'}^{i'} := \widehat{y}_{v'}^{i'} - \epsilon. \end{aligned}$$

Solution $(\bar{x}, \bar{y})$ is clearly feasible since it respects all constraints (4.2)-(4.4), (4.14). Moreover, the objective-function value of solution $(\bar{x}, \bar{y})$ is, by definition, smaller than that of solution $(\widehat{x}, \widehat{y})$. Therefore, $(\widehat{x}, \widehat{y})$ is not optimal – a contradiction. $\blacksquare$

Of course, from the moment one changes the gradient of the objective function (4.1), Proposition 4.4 is no longer valid, and constraints 4.8 might need to be reintegrated to the formulation.

Theorem 4.5 - Inequalities (4.14) **defining facets**
Inequalities

$$x_e^i \geq 0 \qquad \forall e \in E, i \in K, \tag{4.14}$$

are facet-defining for $\mathcal{P}_{(V,E,C)}$ *if and only if* $p \geq 3$.

Proof. Let e' be any demand in E and i' be any vehicle in K. We show that face F, defined as

$$F = \left\{(x, y) \in \mathcal{P}_{(V,E,C)} : x_{e'}^{i'} = 0\right\},$$

is a maximal face of $\mathcal{P}_{(V,E,C)}$, and therefore inequalities (4.14) are facet-defining. More specifically, we show that if there exists some face F' of $\mathcal{P}_{(V,E,C)}$, defined as

$$F' = \left\{(x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \alpha_e^i x_e^i + \sum_{v \in V} \sum_{i \in K} \beta_v^i y_v^i = \gamma\right\},$$

for which $F \subseteq F'$, then F' can be rewritten as

$$F' = \left\{(x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \omega x_{e'}^{i'} = \sum_{e \in E} \lambda_e\right\},$$

that is, a linear combination of (4.14) itself and the implicit equalities (4.2), with weights $\omega \in \mathbb{R}$ and $\lambda \in \mathbb{R}^m$, respectively. As a consequence, one has that $F = F'$.

The necessity of condition $p \geq 3$, comes from the fact that if $K = \{i', j'\}$ (*i.e.*, $p = 2$) and $x_{e'}^{i'} = 0$, then $x_{e'}^{j'} = 1$ from equalities (4.2). Therefore, $F \subseteq F'$ where

$$F' = \left\{(x, y) \in \mathcal{P}_{(V,E,C)} : x_{e'}^{j'} - y_{o_{e'}}^{j'} = 0\right\}$$

is the face induced by inequality (4.4) associated with demand e', vehicle j' and station $o_{e'}$.

The proof of sufficiency is organized in the following three steps:

i. showing $\beta_v^i = 0$ for any $v \in V, i \in K$;

ii. showing $\alpha_e^j = \alpha_e^{i'}$ for any $e \in E \setminus e', j \in K$, and $\alpha_{e'}^j = \alpha_{e'}^k$ for any $j \in K \setminus i', k \in K \setminus i'$;

iii. showing $\sum_{e \in E} \alpha_e^i = \gamma$ for any $i \in K$.

Step i.: Let $(\bar{x}, \bar{y})$ be a feasible solution where some arbitrarily chosen vehicle $i \in K$ is perfectly idle, and demand e' is not assigned to vehicle i'. Such solution exists, since $p \geq 3$ and Property 4 holds. By definition, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.1 applies for any chosen station $v \in V$, and therefore $\beta_v^i = 0$ for every $v \in V$.

Since vehicle i can be arbitrarily chosen, $\beta_v^i = 0$ for any $v \in V, i \in K$, and face F' can be rewritten as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \alpha_e^i x_e^i = \gamma \right\}.$$

Step ii.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is perfectly idle. By definition, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.2 applies for any chosen demand $e \in E \setminus e'$. Notice that a demand $e \in E \setminus e'$ can be assigned to any vehicle $j \in K \setminus i'$, by taking the necessary number of sequential permutations of $(\bar{x}, \bar{y})$ over $K \setminus i'$. Since all such solutions belong to F, one has that $\alpha_e^j = \alpha_e^{i'}$ for any $e \in E \setminus e', j \in K$. Let $\lambda_e \in \mathbb{R}$ be some real number such that $\lambda_e = \alpha_e^j$ for any $e \in E \setminus e', j \in K$.

Consider now that $(\bar{x}, \bar{y})$ is a feasible solution where some arbitrarily chosen vehicle $j \in K$ is perfectly idle, and demand e' is not assigned to vehicle i'. By definition, $(\bar{x}, \bar{y}) \in F$. Notice that demand e' can be assigned to any vehicle $j \in K \setminus i'$, by taking the necessary number of sequential permutations of $(\bar{x}, \bar{y})$ over $K \setminus i'$. All such solutions belong to F. Therefore, from Lemma 4.2, $\alpha_{e'}^j = \alpha_{e'}^k$ for any $j \in K \setminus i', k \in K \setminus i'$. Let $\lambda_{e'} \in \mathbb{R}$ be some real number such that $\lambda_{e'} = \alpha_{e'}^j$ for any $j \in K \setminus i'$. Moreover, let $\omega \in \mathbb{R}$ be some real number such that $\omega = \alpha_{e'}^{i'} - \lambda_{e'}$.

Face F' can now be rewritten as

$$\begin{aligned} F' &= \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E \setminus e'} \sum_{i \in K} \lambda_e x_e^i + \sum_{i \in K \setminus i'} \lambda_{e'} x_{e'}^i + \lambda_{e'} x_{e'}^{i'} + \omega x_{e'}^{i'} = \gamma \right\} \\ &= \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \omega x_{e'}^{i'} = \gamma \right\}. \end{aligned}$$

Step iii.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is perfectly idle. By definition, $(\bar{x}, \bar{y}) \in F$. Let us construct other $p-2$ solutions by doing $p-2$ times a sequential permutation of $(\bar{x}, \bar{y})$ over $K \setminus i'$. Since vehicle i' remains idle in all such solutions, each constructed solution belongs to F and therefore satisfies

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i = \gamma. \tag{4.30}$$

Notice that, considering this pool of $p-1$ feasible solutions (the $p-2$ constructed solutions plus $(\bar{x}, \bar{y})$), each demand is assigned to a vehicle $i \in K$ exactly once. Summing up equations (4.30) for each $p-1$ solution considered, one obtains

$$(p-1) \sum_{e \in E} \lambda_e = (p-1)\gamma.$$

Therefore, $\sum_{e \in E} \lambda_e = \gamma$ and face F' can finally be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \omega x_{e'}^{i'} = \sum_{e \in E} \lambda_e \right\}. \qquad \blacksquare$$

Through Theorem 4.4 and Theorem 4.5, we provide the necessary and sufficient conditions for which inequalities (4.8) and (4.14) define facets of $\mathcal{P}_{(V,E,C)}$. We now continue such study by considering next inequalities (4.4) and (4.3).

Theorem 4.6 - Inequalities (4.4) **defining facets**
The stop inequalities

$$x_e^i - y_v^i \leq 0 \qquad \forall i \in K, e \in E, v \in \{o_e, d_e\}, \tag{4.4}$$

are facet-defining for $\mathcal{P}_{(V,E,C)}$.

Proof. Let e' be any demand in E and i' be any vehicle in K. The proof described below is done for $v = o_{e'}$, but it can trivially be adapted for $v = d_{e'}$. We show that face F, defined as

$$F = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : x_{e'}^{i'} - y_{o_{e'}}^{i'} = 0 \right\},$$

is a maximal face of $\mathcal{P}_{(V,E,C)}$, and therefore a facet of $\mathcal{P}_{(V,E,C)}$. More specifically, we show that if there exists some face F' of $\mathcal{P}_{(V,E,C)}$, defined as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \alpha_e^i x_e^i + \sum_{v \in V} \sum_{i \in K} \beta_v^i y_v^i = \gamma \right\},$$

for which $F \subseteq F'$, then F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \omega x_{e'}^{i'} - \omega y_{o_{e'}}^{i'} = \sum_{e \in E} \lambda_e \right\},$$

that is, a linear combination of (4.4) itself and the implicit equalities (4.2), with weights $\omega \in \mathbb{R}$ and $\lambda \in \mathbb{R}^m$, respectively. As a consequence, one has that $F = F'$.

The proof is organized in the following five steps:

i. showing $\beta_v^i = 0$ for any $v \in V, i \in K \setminus i'$ and $\beta_v^{i'} = 0$ for any $v \in V \setminus o_{e'}$;

ii. showing $\alpha_e^{i'} = \alpha_e^j$ for any $e \in E \setminus e', j \in K$;

iii. showing $\alpha_{e'}^j = \alpha_{e'}^k$ for any $j \in K \setminus i', k \in K \setminus i'$;

iv. showing $\alpha_{e'}^j = \alpha_{e'}^{i'} + \beta_{o_{e'}}^{i'}$ for any $j \in K \setminus i'$;

v. showing $\gamma = \sum_{e \in E} \alpha_e^i$ for any $i \in K \setminus i'$.

Step i.: Let $(\bar{x}, \bar{y})$ be a feasible solution where some arbitrarily chosen vehicle $j \in K \setminus i'$ is perfectly idle, and demand e' is assigned to vehicle i' (*i.e.*, $\bar{x}_{e'}^{i'} = 1$). Since $\bar{x}_{e'}^{i'} = \bar{y}_{o_{e'}}^{i'} = 1$, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.1 applies for any chosen station $v \in V$, and therefore $\beta_v^j = 0$ for any $v \in V$. Since vehicle j can be arbitrarily chosen among $K \setminus i'$, $\beta_v^i = 0$ for any $v \in V, i \in K \setminus i'$.

Consider now that $(\bar{x}, \bar{y})$ is a feasible solution where vehicle i' is perfectly idle. Since $\bar{x}_{e'}^{i'} = \bar{y}_{o_{e'}}^{i'} = 0$, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.1 applies for any chosen station $v \in V \setminus o_{e'}$, and therefore $\beta_v^{i'} = 0$ for any $v \in V \setminus o_{e'}$. Let $\omega \in \mathbb{R}$ be some real number such that $\omega = -\beta_{o_{e'}}^{i'}$. Then, face F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \alpha_e^i x_e^i - \omega y_{o_{e'}}^{i'} = \gamma \right\}.$$

Step ii.: Let $(\bar{x}, \bar{y})$ be a feasible solution such that some arbitrarily chosen demand $e \in E \setminus e'$ and demand e' are assigned to vehicle i'. Moreover, consider that some arbitrarily chosen vehicle $j \in K \setminus i'$ is perfectly idle. Such solution exists since ≥ 2. Since $\bar{x}_{e'}^{i'} = \bar{y}_{o_{e'}}^{i'} = 1$, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.2 applies for vehicle j and demand e, and therefore $\alpha_e^{i'} = \alpha_e^j$. Since vehicle j and demand e can be both arbitrarily chosen among $K \setminus i'$ and $E \setminus e'$, respectively, one has $\alpha_e^{i'} = \alpha_e^j$ for any $e \in E \setminus e'$ and $j \in K$.

Let $\lambda_e \in \mathbb{R}$ be some real number such that $\lambda_e = \alpha_e^{i'}$ for any $e \in E \setminus e'$. Then, face F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E \setminus e'} \sum_{i \in K} \lambda_e x_e^i + \sum_{i \in K} \alpha_{e'}^i x_{e'}^i - \omega y_{o_{e'}}^{i'} = \gamma \right\}.$$

Step iii.: If $p = 2$, the proof is trivial. Hence, let us assume $p \geq 3$. Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is perfectly idle and some arbitrarily chosen demand $e \in E \setminus e'$ is the only demand assigned to some arbitrarily chosen vehicle $j \in K \setminus i'$. Fixing vehicle i' to be idle and vehicle j to take a single demand e, leaves $(p-2)C$ available seats for $m-1$ demands. By definition, $m-1$ demands can occupy at most $m-1$ seats simultaneously. Therefore, one must ensure that, $(p-2)C \geq m-1$. Since $C \geq 2$, one has $(p-2)C \geq 2p-4$. Notice however, that $2p-4 \geq m-1$ if and only if $p \geq 3$, since $p = m$. Hence, one has a proof of the existence of such $(\bar{x}, \bar{y})$ solution. Therefore, $(\bar{x}, \bar{y}) \in F$ and the following equality holds:

$$\sum_{e \in E} \sum_{i \in K} \alpha_e^i \bar{x}_e^i - \omega \bar{y}_{o_{e'}}^{i'} = \gamma.$$

Considering solution $(\bar{x}, \bar{y})$, let $k \in K \backslash \{i', j\}$ denote the vehicle on which demand e' is assigned to. Now construct another feasible solution $(\hat{x}, \hat{y})$ by transferring demand e' from vehicle k to vehicle j. Since solution $(\hat{x}, \hat{y})$ belongs to F, the following equality holds:

$$\sum_{e \in E} \sum_{i \in K} \alpha_e^i \bar{x}_e^i + \alpha_{e'}^j - \alpha_{e'}^k - \omega \bar{y}_{o_{e'}}^{i'} = \gamma.$$

It easily follows that $\alpha_{e'}^j = \alpha_{e'}^k$. Notice that demand e' can be assigned to any vehicle $k \in K \setminus \{i', j\}$, by taking the necessary number of sequential permutations of $(\bar{x}, \bar{y})$ over $K \setminus \{i', j\}$. Therefore,

$$\alpha_{e'}^j = \alpha_{e'}^k \qquad \forall k \in K \setminus \{i'\}.$$

Let $\lambda_{e'} \in \mathbb{R}$ be some real number such that $\lambda_{e'} = \alpha_{e'}^i$ for any $i \in K \setminus i'$. Then, face F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E \setminus e'} \sum_{i \in K} \lambda_e x_e^i + \sum_{i \in K \setminus i'} \lambda_{e'} x_{e'}^i + \alpha_{e'}^{i'} x_{e'}^{i'} - \omega y_{o_{e'}}^{i'} = \gamma \right\}.$$

Step iv.: Let $(\bar{x}, \bar{y})$ be a feasible solution such that vehicle i' is perfectly idle. Since $\bar{x}_{e'}^{i'} = \bar{y}_{o_{e'}}^{i'} = 0$, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.2 applies for vehicle i' and demand e'. Therefore, $\alpha_e^{i'} - \omega = \lambda_e$ and face F' can now be rewritten as

$$\begin{aligned} F' &= \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E \setminus e'} \sum_{i \in K} \lambda_e x_e^i + \sum_{i \in K \setminus i'} \lambda_{e'} x_{e'}^i + \lambda_{e'} x_{e'}^{i'} + \omega x_{e'}^{i'} - \omega y_{o_{e'}}^{i'} = \gamma \right\} \\ &= \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E \setminus e'} \sum_{i \in K} \lambda_e x_e^i + \sum_{i \in K} \lambda_{e'} x_{e'}^i + \omega x_{e'}^{i'} - \omega y_{o_{e'}}^{i'} = \gamma \right\} \\ &= \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \omega x_{e'}^{i'} - \omega y_{o_{e'}}^{i'} = \gamma \right\}. \end{aligned}$$

Step v.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is perfectly idle. By definition, $(\bar{x}, \bar{y}) \in F$. Let us construct other $p-2$ solutions by taking $p-2$ times the sequential permutation of $(\bar{x}, \bar{y})$ over $K \setminus i'$. Since vehicle i' is perfectly idle in all such solutions, each constructed solution belongs to F and therefore satisfies

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i = \gamma. \tag{4.31}$$

Notice that, considering this pool of $p-1$ feasible solutions (the $p-2$ constructed

solutions plus $(\bar{x}, \bar{y})$), each demand is assigned to a vehicle $i \in K \setminus i'$ exactly once. Summing up the $p-1$ equations (4.31) associated with the pool of solutions considered, one obtains

$$(p-1)\sum_{e\in E}\lambda_e = (p-1)\gamma.$$

Therefore, $\gamma = \sum_{e\in E}\lambda_e$ and face F' can finally be rewritten as

$$F' = \left\{(x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in E}\sum_{i\in K}\lambda_e x_e^i + \omega x_{e'}^{i'} - \omega y_{o_{e'}}^{i'} = \sum_{e\in E}\lambda_e.\right\}. \qquad \blacksquare$$

Theorem 4.7 - Inequalities (4.3) **defining facets**
The capacity constraints

$$\sum_{e\in\Delta_E(v)} x_e^i \leq C \qquad \forall v \in V, i \in K, \tag{4.3}$$

are facet-defining for a given station $v' \in V$ and vehicle $i' \in K$ if and only if

1. *there is no station $v \in V$ such that $\Delta_E(v') \subset \Delta_E(v)$;*
2. *for any station $v \in V$, $|\Delta_E(v') \setminus \delta(v)| \geq C$.*

Proof. We show that face F, defined as

$$F = \left\{(x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in\Delta_E(v')} x_e^{i'} = C\right\},$$

is a maximal face of $\mathcal{P}_{(V,E,C)}$, and therefore inequalities (4.3) associated with station $v' \in V$ and vehicle $i' \in K$ is facet-defining. More specifically, we show that if there exists some face F' of $\mathcal{P}_{(V,E,C)}$, defined as

$$F' = \left\{(x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in E}\sum_{i\in K}\alpha_e^i x_e^i + \sum_{v\in V}\sum_{i\in K}\beta_v^i y_v^i = \gamma\right\},$$

for which $F \subseteq F'$, then F' can be rewritten as

$$F' = \left\{(x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in E}\sum_{i\in K}\lambda_e x_e^i + \sum_{e\in\Delta_E(v')}\omega x_e^{i'} = \omega C + \sum_{e\in E}\lambda_e\right\},$$

that is, a linear combination of (4.3) itself and the implicit equalities (4.2), with weights $\omega \in \mathbb{R}$ and $\lambda \in \mathbb{R}^m$, respectively. As a consequence, one has that $F = F'$.

The necessity of condition 1, comes from the fact that if there exists a station $v \in V$ for which $\Delta_E(v') \subset \Delta_E(v)$, then $\sum_{e\in\Delta_E(v')} x_e^{i'} = C$ implies $\sum_{e\in\Delta_E(v)} x_e^{i'} =$

C. Therefore, inequalities (4.3) for station v' are dominated by inequalities (4.3) associated with station v. Condition 2 is necessary since if there exists a station $v \in V$, for which $|\Delta_E(v') \setminus \delta(v)| < C$, then any assignment of C demands in $\Delta_E(v')$ to vehicle i' would induce the vehicle to stop at station v, that is $y_v^{i'} = 1$.

The proof of sufficiency is organized in the following five steps:

i. showing $\beta_v^i = 0$ for any $v \in V, i \in K$;

ii. showing $\alpha_e^j = \alpha_e^k$ for any $e \in E, j \in K \setminus i', k \in K \setminus i'$;

iii. showing $\alpha_e^{i'} = \alpha_e^j$ for any $e \in E \setminus \Delta_E(v'), j \in K$;

iv. showing $\alpha_{e_1}^{i'} - \alpha_{e_1}^j = \alpha_{e_2}^{i'} - \alpha_{e_2}^j$ for any $e_1 \in \Delta_E(v'), e_2 \in \Delta_E(v'), j \in K \setminus i'$;

v. showing $\gamma = \sum_{e \in E} \alpha_e^j + C(\alpha_{e'}^{i'} - \alpha_{e'}^j)$ for any $e' \in \Delta_E(v'), j \in K \setminus i'$.

Step i.: Let $(\bar{x}, \bar{y})$ be a feasible solution where some arbitrarily chosen vehicle $j \in K \setminus i'$ is perfectly idle, and vehicle i' is fully charged at station v'. Such solution clearly belongs to F. Then, Lemma 4.1 applies for any chosen station $v \in V$, and therefore $\beta_v^j = 0$ for any $v \in V$. Since vehicle j can be arbitrarily chosen among $K \setminus i'$, $\beta_v^i = 0$ for any $v \in V, i \in K \setminus i'$.

Consider now that $(\bar{x}, \bar{y})$ is a feasible solution where vehicle i' is fully charged at station v' and does not stops at some arbitrarily chosen station $v \in V$. Such a solution exists from condition 2. Then, Lemma 4.1 applies, and therefore $\beta_v^{i'} = 0$. Since station $v \in V$ can be arbitrarily chosen, $\beta_v^{i'} = 0$ for any $v \in V$. Hence, face F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \alpha_e^i x_e^i = \gamma \right\}.$$

Step ii.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is fully charged at station v' and some arbitrarily chosen vehicle $j \in K \setminus i'$ is perfectly idle. Moreover, consider that some arbitrarily chosen demand $e \in E$ is assigned to some arbitrarily chosen vehicle $k \in K \setminus \{i', j\}$. Such solution exists since condition 2 implies that $|\Delta_E(v')| \geq C + 1$. Since $(\bar{x}, \bar{y}) \in F$, Lemma 4.2 applies for vehicle j and demand e, and therefore $\alpha_e^j = \alpha_e^k$. Since vehicle j and demand e can be both arbitrarily chosen among $K \setminus i'$ and E, respectively, one has $\alpha_e^j = \alpha_e^k$ for any $e \in E, j \in K \setminus i', k \in K \setminus i'$.

Let $\lambda_e \in \mathbb{R}$ be some real number such that $\lambda_e = \alpha_e^j$ for any $e \in E$ and $j \in K \setminus i'$. Then, face F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in E} \alpha_e^{i'} x_e^{i'} = \gamma \right\}.$$

Step iii.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is fully charged at station v' and some arbitrarily chosen vehicle $j \in K \setminus i'$ is perfectly idle. Moreover, consider that some arbitrarily chosen demand $e' \in E \setminus \Delta_E(v')$ is also assigned to vehicle i'. Such solution exists from condition 1. Since $(\bar{x}, \bar{y}) \in F$, Lemma 4.2 applies for vehicle j and demand e', and therefore $\alpha_{e'}^{i'} = \alpha_{e'}^{j}$. Since vehicle j and demand e' can be both arbitrarily chosen among $K \setminus i'$ and $E \setminus \Delta_E(v')$, respectively, one has $\alpha_{e'}^{i'} = \alpha_{e'}^{j} = \lambda_{e'}$ for any $e' \in E \setminus \Delta_E(v'), j \in K$.

Then, face F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in E \setminus \Delta_E(v')} \lambda_e x_e^{i'} + \sum_{e \in \Delta_E(v')} \alpha_e^{i'} x_e^{i'} = \gamma \right\}.$$

Step iv.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is fully charged at station v'. Let $S \subset \Delta_E(v')$ denote the set of demands assigned to vehicle i'. Moreover, consider that some arbitrarily chosen demand $e' \in \Delta_E(v')$ is the only demand assigned to some arbitrarily chosen vehicle $j \in K \setminus i'$. Since $(\bar{x}, \bar{y}) \in F$, the following equality holds:

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E \setminus \Delta_E(v')} \lambda_e \bar{x}_e^{i'} + \sum_{e \in \Delta_E(v')} \alpha_e^{i'} \bar{x}_e^{i'} = \gamma.$$

Now construct another feasible solution $(\hat{x}, \hat{y})$ by exchanging demands e' and some arbitrarily chosen demand $e'' \in S$. Then, $(\hat{x}, \hat{y})$ also belong to F, and the following equality holds:

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E \setminus \Delta_E(v')} \lambda_e \bar{x}_e^{i'} + \sum_{e \in \Delta_E(v')} \alpha_e^{i'} x_e^{i'} - \lambda_{e'} + \alpha_{e'}^{i'} - \alpha_{e''}^{i'} + \lambda_{e''} = \gamma.$$

Therefore, $\alpha_{e'}^{i'} - \lambda_{e'} = \alpha_{e''}^{i'} - \lambda_{e''}$. Since demands e' and e'' can be arbitrarily chosen among $\Delta_E(v')$, one obtains

$$\alpha_{e'}^{i'} - \lambda_{e'} = \alpha_{e''}^{i'} - \lambda_{e''} \qquad \forall e' \in \Delta_E(v'), e'' \in \Delta_E(v').$$

Let $\omega \in \mathbb{R}$ be some real number such that $\omega = \alpha_e^{i'} - \lambda_e$ for any $e \in \Delta_E(v')$. Then, face F' can be rewritten as

$$\begin{aligned} F' &= \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in E \setminus \Delta_E(v')} \lambda_e x_e^{i'} + \sum_{e \in \Delta_E(v')} (\omega + \lambda_e) x_e^{i'} = \gamma \right\} \\ &= \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in E} \lambda_e x_e^{i'} + \sum_{e \in \Delta_E(v')} \omega x_e^{i'} = \gamma \right\} \end{aligned}$$

$$= \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \sum_{e \in \Delta_E(v')} \omega x_e^{i'} = \gamma \right\}.$$

Step v.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is fully charged at station v'. Let $S \subset \Delta_E(v')$ denote the set of demands assigned to vehicle i'. Let us construct other $p-2$ solutions by taking $p-2$ times the sequential permutation of $(\bar{x}, \bar{y})$ over $K \setminus i'$. Since vehicle i' is fully charged at station v' in all such solutions, each constructed solution belongs to F and therefore satisfies

$$\sum_{e \in E \setminus S} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in S} \lambda_e + \sum_{e \in S} \omega = \gamma. \tag{4.32}$$

Notice that, by definition, $|S| = C$. Therefore, equality (4.32) can be rewritten as

$$\sum_{e \in E \setminus S} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in S} \lambda_e + \omega C = \gamma. \tag{4.33}$$

Considering this pool of $p-1$ feasible solutions (the $p-2$ constructed solutions plus $(\bar{x}, \bar{y})$), each demand in $E \setminus S$ is assigned to a vehicle $i \in K \setminus i'$ exactly once. Summing up the $p-1$ equations (4.33) associated with the pool of solutions considered, one obtains

$$(p-1) \sum_{e \in E \setminus S} \lambda_e + (p-1) \sum_{e \in S} \lambda_e + (p-1)\omega C = (p-1)\gamma,$$

which reduces to

$$\sum_{e \in E \setminus S} \lambda_e + \sum_{e \in S} \lambda_e + \omega C = \gamma.$$

Therefore, $\gamma = \sum_{e \in E} \lambda_e + \omega C$ and face F' can finally be rewritten as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \sum_{e \in \Delta_E(v')} \omega x_e^{i'} = \sum_{e \in E} \lambda_e + \omega C \right\},$$

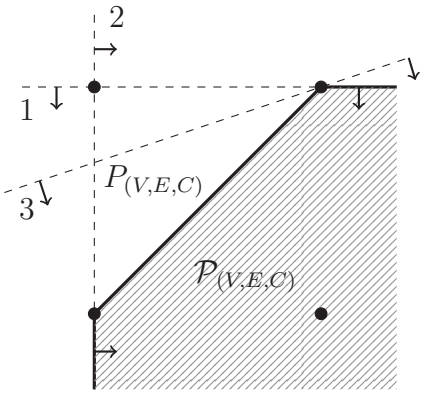
and the proof is done. ■

4.4 New valid inequalities and facets

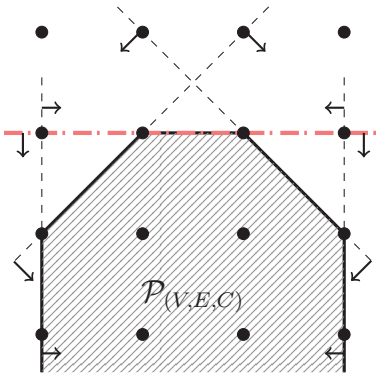
In the previous section the necessary and sufficient conditions for which the inequalities composing polyhedron $P_{(V,E,C)}$ define facets of $\mathcal{P}_{(V,E,C)}$ were given. Indeed, whenever a necessary condition was not satisfied, the inequality under analysis was shown to be dominated by some other inequality composing $P_{(V,E,C)}$. At this point,

an inexperienced or reckless reader may question himself if $P_{(V,E,C)}$ isn't, in fact, the exact same polyhedron as $\mathcal{P}_{(V,E,C)}$. Nonetheless, the results from Section 4.2 clearly indicates that not only this is not the case, but also that the objective function gradient points towards an extreme point of $P_{(V,E,C)}$ that does not belong to $\mathcal{P}_{(V,E,C)}$, that is, a fractional extreme point. Figure 4.4 illustrates how such situation can be achieved. Of course, the reader should keep in mind that Figure 4.4 gives an example on a simple 2-dimensional space, and thus its reasoning should be abstracted to the higher dimensional space on which polyhedrons $P_{(V,E,C)}$ and $\mathcal{P}_{(V,E,C)}$ are inserted.

In Figure 4.4, each dot represents an integer solution. In Figure 4.4a, polyhedron $\mathcal{P}_{(V,E,C)}$ is given by the shaded area, while $P_{(V,E,C)}$ is defined as the intersection of halfspaces induced by inequalities 1, 2 and 3. Clearly, inequality 3 does not define a facet of $\mathcal{P}_{(V,E,C)}$. Moreover, such inequality is dominated by inequality 1, since every solution of $\mathcal{P}_{(V,E,C)}$ satisfying inequality 3 at equality also satisfies inequality 1 at equality. In Figure 4.4b, polyhedron $P_{(V,E,C)}$ is defined as the intersection of halfspaces induced by the inequalities represented in grey dashed lines. Notice that all such inequalities are facet-defining for $\mathcal{P}_{(V,E,C)}$. Nonetheless, they are not sufficient for completely characterizing $\mathcal{P}_{(V,E,C)}$. Indeed, the inequality represented by the red line is required.



(a) Inequalities composing $P_{(V,E,C)}$ not defining facets of $\mathcal{P}_{(V,E,C)}$ are dominated by other inequalities composing $P_{(V,E,C)}$



(b) All inequalities composing $P_{(V,E,C)}$ define facets of $\mathcal{P}_{(V,E,C)}$. Yet, $P_{(V,E,C)} \neq \mathcal{P}_{(V,E,C)}$.

Figure 4.4: Illustration of $\mathcal{P}_{(V,E,C)}$ and $P_{(V,E,C)}$

This section focuses on the search for new valid inequalities capable of strengthening the formulation given by $P_{(V,E,C)}$ and thus reducing its integrality gap.

4.4.1 Strong capacity inequalities

Theorem 4.8 - Validity of strong capacity inequalities
The following Strong Capacity inequalities are valid for $\mathcal{P}_{(V,E,C)}$.

$$\sum_{e \in \delta_G^-(v)} x_e^i - C y_v^i \leq 0 \qquad \forall v \in V, i \in K, \tag{4.34a}$$

$$\sum_{e \in \delta_G^+(v)} x_e^i - C y_v^i \leq 0 \qquad \forall v \in V, i \in K. \tag{4.34b}$$

Proof. On any feasible solution, if vehicle $i \in K$ does not stop at station $v \in V$ (*i.e.* $y_v^i = 0$) then it cannot take any demand with origin (or destination) on v. On the other hand, if it stops at station v (*i.e.* $y_v^i = 1$), then it is capable of serving at most C demands. ■

Such inequalities yield a new lower bound capable of enhancing the trivial lower bound of n stops from Proposition 3.1. By summing up inequalities (4.34a) for each vehicle $i \in K$, one obtains

$$\sum_{e \in \delta_G^-(v)} \sum_{i \in K} x_e^i - C \sum_{i \in K} y_v^i \leq 0 \qquad \forall v \in V.$$

However, from constraints (4.2) one has that

$$\sum_{e \in \delta_G^-(v)} \sum_{i \in K} x_e^i = |\delta_G^-(v)| \qquad \forall v \in V,$$

and therefore,

$$\sum_{i \in K} y_v^i \geq \frac{|\delta_G^-(v)|}{C} \qquad \forall v \in V. \tag{4.35}$$

Since $\sum_{i \in K} y_v^i$ should be an integer value, one can derive the following Chvàtal-Gomory cuts from (4.35):

$$\sum_{i \in K} y_v^i \geq \left\lceil \frac{|\delta_G^-(v)|}{C} \right\rceil \qquad \forall v \in V.$$

Of course, the same reasoning can be applied to (4.34b) to obtain

$$\sum_{i \in K} y_v^i \geq \left\lceil \frac{\max\{|\delta_G^-(v)|, |\delta_G^+(v)|\}}{C} \right\rceil \qquad \forall v \in V. \tag{4.36}$$

After summing up inequalities (4.36) for each station $v \in V$, one derives the less trivial lower bound from Proposition 3.2. Such result is summarized by the following

Proposition.

Proposition 4.5 - Reinforcement from strong capacity inequalities
Let $P'_{(V,E,C)} = \{(x,y) \in P_{(V,E,C)} : (x,y)$ *satisfies* $(4.34), (4.36)\}$. *Then,*

$$\min\left\{\mathbf{0}x + \mathbf{1}y : (x,y) \in P'_{(V,E,C)}\right\} \geq \sum_{v \in V} \left\lceil \frac{\max\{|\delta_G^-(v)|, |\delta_G^+(v)|\}}{C} \right\rceil . \qquad \blacksquare$$

It follows that, for $C = 1$, the inclusion of inequalities (4.34), (4.36) allows the linear relaxation $\min\{\mathbf{0}x + \mathbf{1}y : (x,y) \in P'_{(V,E,C)}\}$ to find a solution of optimal cost (*cf.* Proposition 3.9). In addition, if the associated graph $G = (V,E)$ is a star, then the linear relaxation also yields a solution of optimal cost for any capacity value C (*cf.* Proposition 3.11).

As a consequence, such inequalities are capable of cutting off the fractional optimal solution obtained by the linear relaxation proposed for Example 4.2. Indeed, in Example 4.2, $\max\{|\delta_G^-(v)|, |\delta_G^+(v)|\} = m$ for any of the two stations in V. Therefore, including inequalities (4.34), (4.36) reinforces the given formulation to a point where its linear relaxation is capable of finding a solution with cost $2\left\lceil \frac{m}{C} \right\rceil$. In other words, the integrality gap is reduced to 0 for the instance depicted in Example 4.2.

Table 4.2 gives a perspective on how such inequalities are capable of strengthening the formulation by comparing the gaps at the root node for the formulations with and without the inclusion of *strong capacity* inequalities. The results obtained with the original formulation (4.1)–(4.6) are displayed under section *Default*, while section *StrongCap* displays the results obtained with the original formulation (4.1)–(4.6) reinforced by *strong capacity* inequalities (4.34), (4.36) included on demand. For each formulation, Table 4.2 shows the best lower bound (LB_r) and upper bound (UB_r) known at the root node of the Branch-and-Bound tree. The gap at the root node is given under column gap_r, and the number of *strong capacity* inequalities added to the formulation is provided under column *cuts*. Finally, the gap percentage closed by the introduction of *strong capacity* inequalities is given under column $\Delta_{gap}(\%)$, measured as $\frac{D-S}{D}$, where D and S refer to the gap_r under formulation *Default* and *StrCap*, respectively.

Table 4.2: How Strong Capacity inequalities reinforce the formulation

Instances				Default			StrCap				Comparison
m	C	ρ	i	LB_r	UB_r	gap_r	LB_r	UB_r	gap_r	cuts	$\Delta_{gap}(\%)$
30	2	1.5	1	21.9	39	43.96	32	37	13.51	276	69.27
30	2	3.0	1	13.8	40	65.39	27	33	18.18	453	72.20
30	2	4.5	1	10	35	71.43	23	25	8	329	88.80
35	2	1.5	1	24.7	56	55.84	35	44	20.45	711	63.38
35	2	3.0	1	14.8	50	70.4	31	36	13.89	1172	80.27
35	2	4.5	1	13.8	38	63.66	31	33	6.06	409	90.48

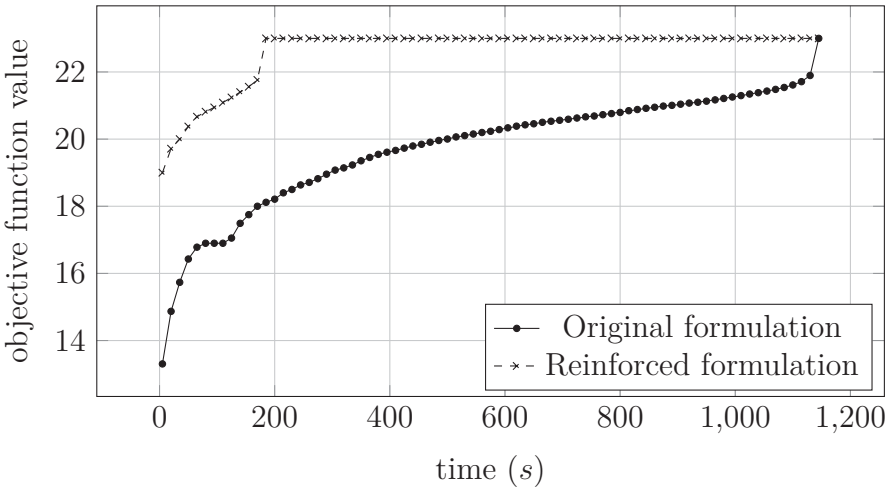
Continued on Next Page...

Table 4.2 – Continued

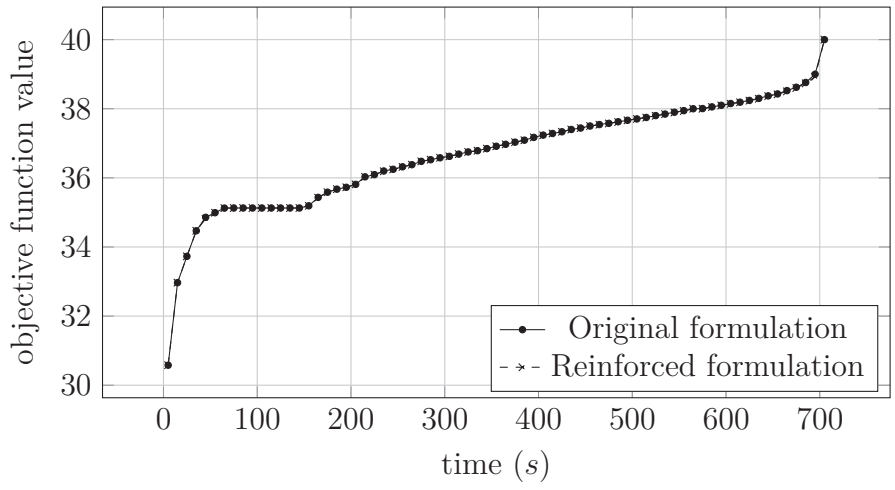
Instances				Default			StrCap				Comparison
m	C	ρ	i	LB_r	UB_r	gap_r	LB_r	UB_r	gap_r	cuts	$\Delta_{gap}(\%)$
40	2	1.5	1	27.8	64	56.55	41	53	22.64	1057	59.96
40	2	3.0	1	16.9	52	67.5	33	38	13.16	764	80.50
40	2	4.5	1	12	42	71.43	33	35	5.71	664	92.01
45	2	1.5	1	31.7	66	51.94	42	57	26.32	1209	49.33
45	2	3.0	1	18.8	54	65.12	39	46	15.22	1937	76.63
45	2	4.5	1	14	58	75.86	39	43	9.3	987	87.74
50	2	1.5	1	36.9	78	52.74	52.7	68	22.55	1677	57.24
50	2	3.0	1	20	75	73.33	45	55	18.18	2611	75.21
50	2	4.5	1	14.8	62	76.05	44	48	8.33	2007	89.05
55	2	1.5	1	38.9	91	57.3	53.5	68	21.32	2048	62.79
55	2	3.0	1	21.9	80	72.64	48	61	0	19584	100.00
55	2	4.5	1	15.9	74	78.58	46	50	8	2524	89.82
60	2	1.5	1	41.9	104	59.67	62	79	21.52	3252	63.93
60	2	3.0	1	23.9	78	69.33	56	65	13.85	14341	80.02
60	2	4.5	1	16.8	85	80.22	52	58	10.34	3397	87.11
30	5	1.5	1	19.5	27	27.74	20.4	29	29.56	71	-6.56
30	5	3.0	1	12.5	20	37.54	15.1	20	24.73	168	34.12
30	5	4.5	1	9.4	16	40.97	14	14	0	210	100.00
35	5	1.5	1	23.4	34	31.05	23.8	39	38.9	83	-25.28
35	5	3.0	1	13.4	23	41.93	15	25	40	158	4.60
35	5	4.5	1	10.3	21	50.92	16.6	19	12.63	337	75.20
40	5	1.5	1	28.3	49	42.27	28	39	28.26	50	33.14
40	5	3.0	1	15.7	29	46.02	19	30	36.64	324	20.38
40	5	4.5	1	10.8	24	55.2	15.2	20	24	436	56.52
45	5	1.5	1	30.6	42	27.19	30.6	42	27.19	0	0.00
45	5	3.0	1	17.5	47	62.87	20.5	34	39.72	429	36.82
45	5	4.5	1	12.6	34	62.94	19.6	28	30	926	52.34
50	5	1.5	1	34.5	52	33.59	35.7	69	48.26	146	-43.67
50	5	3.0	1	18.5	40	53.66	22.9	40	42.87	596	20.11
50	5	4.5	1	13.9	45	69.18	21.6	33	34.55	979	50.06
55	5	1.5	1	37.6	76	50.55	37.7	73	48.3	81	4.45
55	5	3.0	1	20.7	65	68.13	26.7	47	43.23	933	36.55
55	5	4.5	1	14.5	54	73.08	22.2	36	38.33	1174	47.55
60	5	1.5	1	38.5	74	47.91	39.5	76	47.97	161	-0.13
60	5	3.0	1	22.6	65	65.2	28.5	55	48.1	836	26.23
60	5	4.5	1	16.6	55	69.82	23.7	35	32.38	1374	53.62
30	8	1.5	1	21.4	24	10.94	21.4	24	10.94	0	0.00
30	8	3.0	1	11.9	16	25.87	12.3	18	31.82	33	-23.00
30	8	4.5	1	7.9	11	28.53	10.1	13	22.08	92	22.61
35	8	1.5	1	24.7	29	14.94	24.7	29	14.94	0	0.00
35	8	3.0	1	12.9	18	28.55	14.2	18	20.99	52	26.48
35	8	4.5	1	8.8	12	26.54	9.5	13	27.08	53	-2.03
40	8	1.5	1	26.7	32	16.46	26.7	32	16.46	0	0.00
40	8	3.0	1	14.4	22	34.71	14.4	22	34.71	0	0.00
40	8	4.5	1	10.2	14	27.06	11.2	16	30.26	64	-11.83
45	8	1.5	1	31.4	42	25.32	31.4	42	25.32	0	0.00
45	8	3.0	1	17.4	30	42.15	20.1	30	32.98	140	21.76
45	8	4.5	1	12.2	19	35.57	16	24	33.33	203	6.30
50	8	1.5	1	33.8	47	28	33.8	47	28	0	0.00
50	8	3.0	1	18.2	29	37.4	18.5	34	45.7	60	-22.19
50	8	4.5	1	13.3	24	44.55	15.4	26	40.94	175	8.10
55	8	1.5	1	35.6	49	27.33	35.6	49	27.33	0	0.00
55	8	3.0	1	20.3	39	47.86	21.6	51	57.69	218	-20.54
55	8	4.5	1	14.4	33	56.46	15.8	31	49.17	175	12.91
60	8	1.5	1	38.9	76	48.78	38.9	76	48.78	0	0.00
60	8	3.0	1	22.5	41	45.04	22.6	43	47.37	66	-5.17
60	8	4.5	1	15.4	33	53.31	19.3	31	37.63	537	29.41

Strong capacity inequalities are quite effective when the associated graph is dense with respect to C, that is, when nodes usually have degree greater than C. However, they fail to strengthen the formulation when the graph is relatively sparse. It is worth noting that the addition of such inequalities can only reinforce LB_r. Nonetheless, introducing such inequalities can slightly impact the CPLEX heuristics responsible for generating the upper bounds depicted under UB_r. Indeed, the few negative results obtained for Δ_{gap} are due to this fact.

Figure 4.5 shows the evolution of the lower bounds during the optimization process of both formulations for the instance with 45 demands and capacity 5. Figure 4.5a gives a scenario with dense demands – density is set to 4.5 – and Figure 4.5b gives a scenario with sparse demands – density is set to 1.5. As expected, there is almost no improvement on the lower bound for the sparse case, whilst in the dense scenario a considerable improvement can be verified.



(a) Dense scenario: density 4.5



(b) Sparse scenario: density 1.5

Figure 4.5: Progress of lower bounds

As a matter of fact, it easy to see that if $\max\{|\delta_G^-(v)|, |\delta_G^+(v)|\} \leq C$, such constraints are dominated by a sum of inequalities (4.4). Indeed, this implies that even for the particular case of Intersection U-SNP where $C = 2$ (that can be solved in polynomial time), an important integrality gap can still be obtained even after the inclusion of Strong Capacity inequalities to the formulation. The following example illustrates such situation.

Example 4.3 - Strong capacity inequalities can fail to reinforce the given formulation

Consider the instance of Intersection U-SNP given by the following graph with capacity $C = 2$ and an even number of stations $n \geq 4$:

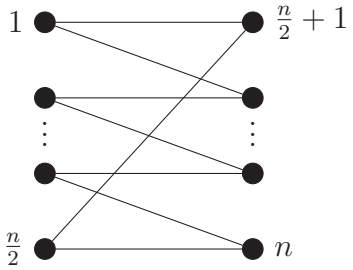


Figure 4.6: Example of instance where strong capacity inequalities are useless

Notice that in the given example, $m = n$. The optimal solution obtained with the polynomial time algorithm described in Section 3.2 has $\frac{3m}{2}$ stops. Since $\max\{|\delta_G^-(v)|, |\delta_G^+(v)|\} \leq C$ for any $v \in V$, the introduced inequalities are redundant. Therefore, Theorem 4.2 still applies and the linear relaxation provides a lower bound of m stops for the given instance. It follows that the integrality gap equals $\frac{1}{3}$. □

Finally, we give the necessary and sufficient conditions for which inequalities (4.34a) and (4.34b) define facets of $\mathcal{P}_{(V,E,C)}$.

Theorem 4.9 - Strong capacity inequalities defining facets

The strong capacity constraint (4.34a) *associated with a station* $v \in V$ *and a vehicle* $i \in K$ *is facet-defining if and only if*

$$|\delta_G^-(v)| \geq C + 1.$$

Proof. We show that face F, defined as

$$F = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in \delta_G^-(v')} x_e^{i'} - C y_{v'}^{i'} = 0 \right\},$$

is a maximal face of $\mathcal{P}_{(V,E,C)}$, and therefore inequality (4.34a) associated with station v' and vehicle i' is facet-defining. More specifically, we show that if there exists some

face F' of $\mathcal{P}_{(V,E,C)}$, defined as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in E}\sum_{i\in K} \alpha_e^i x_e^i + \sum_{v\in V}\sum_{i\in K} \beta_v^i y_v^i = \gamma \right\},$$

for which $F \subseteq F'$, then F' can be rewritten as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in E}\sum_{i\in K} \lambda_e x_e^i + \sum_{e\in \delta_G^-(v')} \omega x_e^{i'} - \omega C y_{v'}^{i'} = \sum_{e\in E} \lambda_e \right\},$$

that is, a linear combination of (4.34a) itself and the implicit equalities (4.2), with weights $\omega \in \mathbb{R}$ and $\lambda \in \mathbb{R}^m$, respectively. As a consequence, one has that $F = F'$.

The necessity of condition $|\delta_G^-(v)| \geq C+1$, comes from the fact that if $|\delta_G^-(v)| \leq C$, then summing up inequalities

$$x_e^{i'} - y_{d_e}^{i'} \leq 0$$

for each $e \in \delta_G^-(v)$, yields the inequality

$$\sum_{e\in\delta_G^-(v')} x_e^{i'} - |\delta_G^-(v)| y_{v'}^{i'} \leq 0,$$

which dominates (4.34a).

The proof of sufficiency is organized in the following six steps:

i. showing $\beta_v^{i'} = 0$ for any $v \in V \setminus v'$ and $\beta_v^i = 0$ for any $v \in V, i \in K \setminus i'$;

ii. showing $\alpha_e^j = \alpha_e^k$ for any $e \in E, j \in K \setminus i', k \in K \setminus i'$;

iii. showing $\alpha_e^{i'} = \alpha_e^j$ for any $e \in E \setminus \delta_G^-(v'), j \in K$;

iv. showing $\alpha_{e'}^{i'} - \alpha_{e'}^j = \alpha_{e''}^{i'} - \alpha_{e''}^j$ for any $e' \in \delta_G^-(v'), e'' \in \delta_G^-(v'), j \in K \setminus i'$;

v. showing $\alpha_e^{i'} - \alpha_e^j = \frac{-\beta_{v'}^{i'}}{C}$ for any $e \in \delta_G^-(v'), j \in K \setminus i'$;

vi. showing $\gamma = \sum_{e\in E} \alpha_e^i$ for any $i \in K \setminus i'$.

Step i.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is perfectly idle. By definition, $\sum_{e\in\delta_G^-(v')} \bar{x}_e^{i'} = C\bar{y}_{v'}^{i'} = 0$ and therefore, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.1 applies for any chosen station $v \in V \setminus v'$. Therefore,

$$\beta_v^{i'} = 0 \qquad \forall v \in V \setminus v'.$$

Consider now that $(\bar{x}, \bar{y})$ is a feasible solution where some arbitrarily chosen vehicle $j \in K \setminus i'$ is perfectly idle, and C demands among $\delta_G^-(v)$ are assigned to vehicle i'. By definition, $\sum_{e \in \delta_G^-(v')} \bar{x}_e^{i'} = C\bar{y}_{v'}^{i'} = C$ and therefore, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.1 applies for any chosen station $v \in V$, and therefore $\beta_v^j = 0$ for any $v \in V$. Since vehicle j can be arbitrarily chosen (as long as $j \neq i'$),

$$\beta_v^i = 0 \qquad \forall v \in V, i \in K \setminus i'.$$

Let $\omega \in \mathbb{R}$ be some real number such that $\omega = -\frac{\beta_{v'}^{i'}}{C}$. Then, face F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \alpha_e^i x_e^i - \omega C y_{v'}^{i'} = \gamma \right\},$$

Step ii.: Let $(\bar{x}, \bar{y})$ be a feasible solution where some arbitrarily chosen vehicle $j \in K \setminus i'$ is perfectly idle, and C demands among $\delta_G^-(v)$ are assigned to vehicle i'. Moreover, consider that some arbitrarily chosen demand $e \in E$ is assigned to some arbitrarily chosen vehicle $k \in K \setminus \{i', j\}$. Such solution exists since $|\delta_G^-(v')| \geq C + 1$ and $p = m$. Since $(\bar{x}, \bar{y}) \in F$, Lemma 4.2 applies for vehicle j and demand e, and therefore $\alpha_e^j = \alpha_e^k$. Since vehicle j and demand e can be both arbitrarily chosen among $K \setminus i'$ and E, respectively, one has

$$\alpha_e^j = \alpha_e^k \qquad \forall e \in E, j \in K \setminus i', k \in K \setminus i'.$$

Let $\lambda_e \in \mathbb{R}$ be some real number such that $\lambda_e = \alpha_e^j$ for any $e \in E$ and $j \in K \setminus i'$. Then, face F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in E} \alpha_e^{i'} x_e^{i'} - \omega C y_{v'}^{i'} = \gamma \right\}.$$

Step iii.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is perfectly idle. Moreover, consider that some arbitrarily chosen demand $e' \in E \setminus \delta_G^-(v')$ is assigned to some arbitrarily chosen vehicle $j \in K \setminus i'$. Since $(\bar{x}, \bar{y}) \in F$, Lemma 4.2 applies for vehicle i' and demand e', and therefore $\alpha_{e'}^{i'} = \alpha_{e'}^j$. Since vehicle j and demand e' can be both arbitrarily chosen among $K \setminus i'$ and $E \setminus \delta_G^-(v')$, respectively, one has

$$\alpha_{e'}^{i'} = \alpha_{e'}^j = \lambda_{e'} \qquad \forall e' \in E \setminus \Delta_E(v'), j \in K.$$

Then, face F' can be rewritten as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in E \setminus \delta_G^-(v')} \lambda_e x_e^{i'} + \sum_{e \in \delta_G^-(v')} \alpha_e^{i'} x_e^{i'} - \omega C y_{v'}^{i'} = \gamma \right\}.$$

Step iv.: Let $(\bar{x}, \bar{y})$ be a feasible solution where C demands among $\delta_G^-(v')$ are assigned to vehicle i'. Let $S \subset \delta_G^-(v')$ denote this set of C demands. Moreover, consider that some arbitrarily chosen demand $e' \in \delta_G^-(v')$ is the only demand assigned to some arbitrarily chosen vehicle $j \in K \setminus i'$. Since $(\bar{x}, \bar{y}) \in F$, the following equality holds.

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E \setminus \delta_G^-(v')} \lambda_e \bar{x}_e^{i'} + \sum_{e \in \delta_G^-(v')} \alpha_e^{i'} \bar{x}_e^{i'} - \omega C \bar{y}_{v'}^{i'} = \gamma.$$

Now construct another feasible solution $(\hat{x}, \hat{y})$ by exchanging demands e' and some arbitrarily chosen demand $e'' \in S$. Then, $(\hat{x}, \hat{y})$ also belong to F, and the following equality holds.

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E \setminus \delta_G^-(v')} \lambda_e \bar{x}_e^{i'} + \sum_{e \in \delta_G^-(v')} \alpha_e^{i'} x_e^{i'} - \lambda_{e'} + \alpha_{e'}^{i'} - \alpha_{e''}^{i'} + \lambda_{e''} - \omega C \bar{y}_{v'}^{i'} = \gamma.$$

Therefore, $\alpha_{e'}^{i'} - \lambda_{e'} = \alpha_{e''}^{i'} - \lambda_{e''}$. Since demands e' and e'' can be arbitrarily chosen among $\delta_G^-(v')$, one obtains

$$\alpha_{e'}^{i'} - \lambda_{e'} = \alpha_{e''}^{i'} - \lambda_{e''} \qquad \forall e' \in \delta_G^-(v'), e'' \in \delta_G^-(v').$$

Step v.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is perfectly idle, and C demands among $\delta_G^-(v)$ are assigned to some arbitrarily chosen vehicle $j \in K \setminus i'$. Let $S \subset \delta_G^-(v')$ denote this set of C demands. Since $(\bar{x}, \bar{y}) \in F$, the following equality holds.

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i = \gamma. \tag{4.37}$$

Now construct another feasible solution $(\hat{x}, \hat{y})$ by transferring demands in S from vehicle j to vehicle i'. The constructed solution $(\bar{x}, \bar{y})$ also belongs to F. Therefore,

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in S} \alpha_e^{i'} - \sum_{e \in S} \lambda_e - \omega C = \gamma. \tag{4.38}$$

By subtracting (4.37) from (4.38), one obtains

$$\sum_{e \in S} (\alpha_e^{i'} - \lambda_e) - \omega C = 0.$$

However, from step iv., it is clear that $\alpha_e^{i'} - \lambda_e$ is a constant for any $e \in \delta_G^-(v)$. Moreover, by definition, $|S| = C$. Therefore, if e' is taken to be any arbitrarily chosen demand in S, one has that

$$C(\alpha_{e'}^{i'} - \lambda_{e'}) - \omega C = 0 \qquad \forall e' \in \delta_G^-(v),$$

and thus

$$\alpha_{e'}^{i'} - \lambda_{e'} = \omega = \frac{-\beta_{v'}^{i'}}{C} \qquad \forall e' \in \delta_G^-(v).$$

Then, face F' can now be rewritten as

$$\begin{aligned} F' &= \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in E}\sum_{i\in K\setminus i'} \lambda_e x_e^i + \sum_{e\in E\setminus\delta_G^-(v')} \lambda_e x_e^{i'} + \sum_{e\in\delta_G^-(v')} (\lambda_e+\omega) x_e^{i'} - \omega C y_{v'}^{i'} = \gamma \right\} \\ &= \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in E}\sum_{i\in K\setminus i'} \lambda_e x_e^i + \sum_{e\in E} \lambda_e x_e^{i'} + \sum_{e\in\delta_G^-(v')} \omega x_e^{i'} - \omega C y_{v'}^{i'} = \gamma \right\} \\ &= \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in E}\sum_{i\in K} \lambda_e x_e^i + \sum_{e\in\delta_G^-(v')} \omega x_e^{i'} - \omega C y_{v'}^{i'} = \gamma \right\}. \end{aligned}$$

Step vi.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is perfectly idle. By definition, $(\bar{x}, \bar{y}) \in F$. Let us construct other $p-2$ feasible solutions by taking $p-2$ times the sequential permutation of $(\bar{x}, \bar{y})$ over $K \setminus i'$. Since vehicle i' is perfectly idle in all such solutions, each constructed solution belongs to F and therefore satisfies

$$\sum_{e\in E}\sum_{i\in K\setminus i'} \lambda_e x_e^i = \gamma. \tag{4.39}$$

Notice that, considering this pool of $p-1$ feasible solutions (the $p-2$ constructed solutions plus $(\bar{x}, \bar{y})$), each demand is assigned to a vehicle $i \in K \setminus i'$ exactly once. Summing up the $p-1$ equations (4.39) associated with the pool of solutions considered, one obtains

$$(p-1)\sum_{e\in E} \lambda_e = (p-1)\gamma.$$

Therefore, $\gamma = \sum_{e\in E} \lambda_e$ and face F' can finally be rewritten as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in E}\sum_{i\in K} \lambda_e x_e^i + \sum_{e\in\delta_G^-(v')} \omega x_e^{i'} - \omega C y_{v'}^{i'} = \sum_{e\in E} \lambda_e \right\}. \qquad \blacksquare$$

Theorem 4.10 - Strong capacity inequalities defining facets
The strong capacity constraints (4.34b) *associated with station* $v' \in V$ *and vehicle*

$i' \in K$ is facet-defining if and only if

$$|\delta_G^+(v)| \geq C + 1.$$

Proof. Analogous to the proof of Theorem 4.9 ■

4.4.2 k-cardinality tree inequalities

In Fischetti et al. [70], a *k-cardinality tree* is defined as a connected acyclic graph $T = (V(T), E(T))$ such that $|E(T)| = k$. Based on this definition, a new family of valid inequalities, denoted *k-cardinality tree* inequalities, is presented below.

Theorem 4.11 - Validity of k-cardinality tree inequalities
The following k-cardinality tree inequalities are valid for $\mathcal{P}_{(V,E,C)}$.

$$\sum_{e \in S} x_e^i - \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) y_v^i \leq 0 \qquad \forall i \in K, j \in V, S \subseteq \Delta_E(j), \tag{4.40}$$

such that $G[S]$ is a k-cardinality tree for $k \geq C + 1$.

Proof. The proof of validity is done by showing that inequalities (4.40) can be seen as Chvátal-Gomory cuts (see Gomory et al. [80], Chvátal [31]). More precisely, we show that given a k-cardinality tree $T = G[S]$, for $k \geq C + 1$, a conical combination of inequalities (4.3) and (4.4), yields the inequality

$$k \sum_{e \in E(T)} x_e^i - k \sum_{v \in V(T)} (\deg_T(v) - 1) y_v^i \leq C,$$

which can be divided by k and the independent term $\frac{C}{k}$ rounded down to obtain:

$$\sum_{e \in E(T)} x_e^i - \sum_{v \in V(T)} (\deg_T(v) - 1) y_v^i \leq \left\lfloor \frac{C}{k} \right\rfloor = 0.$$

For this, we arbitrarily choose a leaf node $r \in V(T)$ to be the root of T. For each edge $e = uv \in E(T)$, such that $u \prec v$ (remark that $v \neq r$), multiply inequality $x_e^i - y_u^i \leq 0$ by $(k - 1 - |E(T_v)|)$ and inequality $x_e^i - y_v^i \leq 0$ by $|E(T_v)|$, where T_v is the subtree of T rooted in v (*cf.* Section 1.1). Likewise, consider the following inequality that is dominated by (4.3) since $E(T) \subseteq \Delta_E(j)$.

$$\sum_{e \in E(T)} x_e^i \leq C.$$

Summing up all such inequalities yields a new valid inequality where each variable

x_e^i for $e \in E(T)$ has coefficient k since

$$(k - 1 - |E(T_v)|) + |E(T_v)| + 1 = k.$$

Furthermore, on the right-hand side the independent term equals C. To finish the proof, let us show that variable y_v^i has coefficient $-k(\deg_T(v) - 1)$ for any $v \in V(T)$.

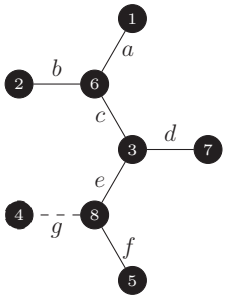
Consider a node $u \in V(T)$. By definition u has $(\deg_T(u) - 1)$ children and each child v of u contributes to the final coefficient of y_u^i with $-(k - 1 - |E(T_v)|)$. Therefore, the total contribution coming from its children equals

$$-(k-1)(\deg_T(u) - 1) + (|E(T_u)| - (\deg_T(u) - 1)) = -k(\deg_T(u) - 1) + |E(T_u)|.$$

Finally, since the parent of u contributes with $-|E(T_u)|$, the proof is done. ■

Corollary 4.2 - Chvàval-Gomory inequalities of rank 1
The k-cardinality tree inequalities (4.40) *are Chvàval-Gomory inequalities of rank 1.* ■

Let $G[S]$ be a k-cardinality tree such that $k \geq C + 2$ and $S \subseteq \Delta_E(j)$, for some $j \in V$. Notice that the k-cardinality tree inequalities associated with tree $G[S]$ are always dominated by the k-cardinality tree inequalities associated with $G[S']$ where $S' \subset S$ and $|S'| > C$. In fact, the former can be written as a sum of stop inequalities (4.4) and the k-cardinality tree inequality associated with $G[S']$ (see Figure 4.7). For this reason, we consider from now on that the edge set inducing a k-cardinality tree inequality has exactly $C + 1$ edges.



$$x_a^i + x_b^i + x_c^i + x_d^i + x_e^i + x_f^i - 2y_3^i - 2y_6^i - y_8^i \leq 0$$
$$x_g^i - y_8^i \leq 0$$
$$\overline{x_a^i + x_b^i + x_c^i + x_d^i + x_e^i + x_f^i + x_g^i - 2y_3^i - 2y_6^i - 2y_8^i \leq 0}$$

Figure 4.7: How a k-cardinality tree gets dominated. Full edges in S', dashed edges in $S \setminus S'$

Next we investigate the effective strength of *k-cardinality tree* inequalities (4.40). For this, recall the instance depicted in Example 4.3. The inclusion of *k-cardinality tree* inequalities is capable of cutting off the fractional optimal solution obtained by the linear relaxation proposed for such instance. Indeed, in Example 4.3, all demands intersect station $\frac{m}{2}$, that is, $\Delta_E(\frac{m}{2}) = E$. Moreover, each station $v \in V$

has exactly 2 demands incident to it. Therefore, the following inequalities define *k-cardinality tree* inequalities for any given vehicle $i \in K$:

$$x_e^i + x_{e'}^i + x_{e''}^i - y_u^i - y_v^i \leq 0, \qquad \forall e = uv \in E, \tag{4.41}$$

where $e' \neq e$ and $e'' \neq e$ are the demands (other than e) incident to u and v, respectively. Figure 4.8 illustrates the 3-cardinality tree associated with such inequalities.

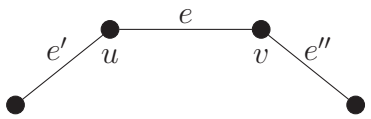


Figure 4.8: A 3-cardinality tree associated with inequalities (4.41)

It follows that each demand in E appears in exactly 3 different inequalities (4.41). Moreover, each station in V appears in exactly 2 different inequalities (4.41). Hence, summing up inequalities (4.41) for each demand $e \in E$, one obtains

$$\sum_{e \in E} 3x_e^i - \sum_{v \in V} 2y_v^i \leq 0, \tag{4.42}$$

for any given vehicle $i \in K$. Since (4.42) is valid for any $i \in K$, one may sum up inequalities (4.42) for each vehicle $i \in K$ to obtain

$$\sum_{e \in E} \sum_{i \in K} 3x_e^i - \sum_{v \in V} \sum_{i \in K} 2y_v^i \leq 0.$$

However, considering constraints (4.2) one has that

$$3m - \sum_{v \in V} \sum_{i \in K} 2y_v^i \leq 0,$$

which yields the following lower bound

$$\sum_{v \in V} \sum_{i \in K} y_v^i \geq \frac{3m}{2}.$$

Therefore, including *k-cardinality tree* inequalities (4.40) reinforces the given formulation to a point where its linear relaxation is capable of finding a solution with cost $\frac{3m}{2}$ for the instance depicted in Example 4.3. In other words, for the given instance, the integrality gap is reduced to 0.

Indeed, the family of *k-cardinality tree* inequalities (4.40) is particularly efficient when it comes to strengthening the given formulation for relatively sparse instances. Unfortunately, the number of inequalities within this family is exponentially large, and therefore, adding all such inequalities to the formulation should be avoided

in practice. This issue will be discussed in more detail in Section 5.4. To prove the strength of such inequalities, the necessary and sufficient conditions for which inequalities (4.40) define facets of $\mathcal{P}_{(V,E,C)}$ are given below.

Theorem 4.12 - k-cardinality tree inequalities defining facets
The k-cardinality tree inequalities (4.40) *are facet-defining under the following necessary and sufficient conditions.*

i. $G[S]$ is not a star;

ii. if $G[S]$ contains exactly 2 non-leaf nodes, then $E \setminus S$ cannot contain an edge (u, v) such that u and v are the non-leaf nodes of $G[S]$ – Figure 4.9 illustrates the case.

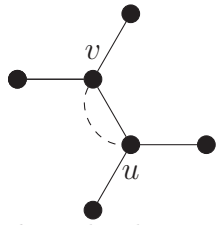


Figure 4.9: Non-facet-defining k-cardinality tree. Full edges in S, dashed edge in $E \setminus S$.

Proof. We first show necessity of conditions i. and ii.. If $G[S]$ is a star with central node v', then $\deg_{G[S]}(v') - 1 = C$, since $G[S]$ has $(C + 1)$ edges. Moreover, either $S \subseteq \delta^-_G(v')$ or $S \subseteq \delta^+_G(v')$. Therefore, the inequality is dominated by either (4.34a) or (4.34b).

Since $G[S]$ is a $(C+1)$-cardinality tree, if $G[S]$ contains exactly 2 non-leaf nodes, say v' and v'', then $\deg_{G[S]}(v') - 1 + \deg_{G[T]}(v'') - 1 = C$. Thus, if there exists an edge $e' = (v', v'') \in E \setminus T$, the inequality

$$x^{i'}_{e'} + \sum_{e \in S} x^{i'}_e - \sum_{v \in V(G[T])} (\deg_{G[T]}(v) - 1) y^{i'}_v \leq 0.$$

is obviously valid and dominates (4.40). Notice that such remark can be used to lift inequalities (4.40) whenever condition ii. does not hold.

We now show that if conditions i. and ii. apply, then (4.40) is facet-defining. For this, we show that face F, defined as

$$F = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in S} x^{i'}_e - \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) y^{i'}_v = 0 \right\},$$

is a maximal face of $\mathcal{P}_{(V,E,C)}$, and therefore inequality (4.40) associated with tree $G[S]$ and vehicle i' is facet-defining. More specifically, we show that if there exists

some face F' of $\mathcal{P}_{(V,E,C)}$, defined as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \alpha_e^i x_e^i + \sum_{v \in V} \sum_{i \in K} \beta_v^i y_v^i = \gamma \right\},$$

for which $F \subseteq F'$, then F' can be rewritten as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \sum_{e \in S} \omega x_e^{i'} - \sum_{v \in V[S]} \omega(\deg_{G[S]}(v) - 1) y_v^{i'} = \sum_{e \in E} \lambda_e \right\},$$

that is, a linear combination of (4.40) itself and the implicit equalities (4.2), with weights $\omega \in \mathbb{R}$ and $\lambda \in \mathbb{R}^m$, respectively. As a consequence, one has that $F = F'$.

The proof is organized in the following seven steps:

i. showing $\beta_v^i = 0$ for any $v \in V, i \in K \setminus i'$;

ii. showing $\alpha_e^j = \alpha_e^k$ for any $e \in E, j \in K \setminus i', k \in K \setminus i'$;

iii. showing $\beta_v^{i'} = 0$ for any $v \in V \setminus I(S)$, where $I(S) = \{v \in V[S] : \deg_{G[S]}(v) \geq 2\}$ is the set of internal vertices of $G[S]$;

iv. showing $\gamma = \sum_{e \in E} \alpha_e^i$ for any $i \in K \setminus i'$;

v. showing $\alpha_{e'}^{i'} - \alpha_{e'}^j = \alpha_{e''}^{i'} - \alpha_{e''}^j$ for any $e' \in S, e'' \in S, j \in K \setminus i'$;

vi. showing $\alpha_e^{i'} = \alpha_e^j$ for any $e \in E \setminus S, j \in K$;

vii. showing $\beta_v^{i'} = -(\deg_{G[S]}(v) - 1)(\alpha_e^{i'} - \alpha_e^j)$ for any $v \in V[S]$, $e \in S$.

Step i.: Let $(\bar{x}, \bar{y})$ be a feasible solution where some arbitrarily chosen vehicle $j \in K \setminus i'$ is perfectly idle, and all demands in $S \setminus e'$ are assigned to vehicle i', for some arbitrarily chosen demand $e' \in S$. By definition, $\sum_{e \in S} \bar{x}_e^{i'} = C = \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) \bar{y}_v^{i'}$ and therefore, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.1 applies for any chosen station $v \in V$. Since vehicle j can be arbitrarily chosen among $K \setminus i'$, one has

$$\beta_v^i = 0 \qquad \forall v \in V, i \in K \setminus i'.$$

Hence, face F' can now be rewritten as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \alpha_e^i x_e^i + \sum_{v \in V} \beta_v^i y_v^{i'} = \gamma \right\}.$$

Step ii.: Consider again solution $(\bar{x}, \bar{y})$ defined in the proof of *Step i.*. Since $(\bar{x}, \bar{y}) \in F$, Lemma 4.2 applies for vehicle j and any demand $e \in E \setminus (S \setminus e')$.

Since vehicle j and demand e' can be both arbitrarily chosen among $K \setminus i'$ and S, respectively, one has

$$\alpha_e^j = \alpha_e^k \qquad \forall e \in E, j \in K \setminus i', k \in K \setminus i'.$$

Let $\lambda_e \in \mathbb{R}$ be some real number such that $\lambda_e = \alpha_e^j$ for each $e \in E$ and $j \in K \setminus i'$. Then, face F' can be rewritten as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in E} \alpha_e^{i'} x_e^{i'} + \sum_{v \in V} \beta_v^i y_v^{i'} = \gamma \right\}.$$

Step iii.: Let $(\bar{x}, \bar{y})$ be a feasible solution where vehicle i' is perfectly idle. By definition, $\sum_{e \in S} \bar{x}_e^{i'} = \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) \bar{y}_v^{i'} = 0$ and therefore, $(\bar{x}, \bar{y}) \in F$. Then, Lemma 4.1 applies for any chosen station $v \in V \setminus I(S)$. Therefore,

$$\beta_v^{i'} = 0 \qquad \forall v \in V \setminus I(S),$$

and face F' can now be rewritten as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in E} \alpha_e^{i'} x_e^{i'} + \sum_{v \in I(S)} \beta_v^i y_v^{i'} = \gamma \right\}.$$

Step iv.: Consider again solution $(\bar{x}, \bar{y})$ defined in the proof of *Step iii.*. Let us construct other $p-2$ feasible solutions by taking $p-2$ times the sequential permutation of $(\bar{x}, \bar{y})$ over $K \setminus i'$. Since vehicle i' is perfectly idle in all such solutions, each constructed solution belongs to F and therefore satisfies

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i = \gamma. \tag{4.43}$$

Notice that, considering this pool of $p-1$ feasible solutions (the $p-2$ constructed solutions plus $(\bar{x}, \bar{y})$), each demand is assigned to a vehicle $i \in K \setminus i'$ exactly once. Summing up the $p-1$ equations (4.43) associated with the pool of solutions considered, one obtains

$$(p-1) \sum_{e \in E} \lambda_e = (p-1)\gamma.$$

Therefore, $\gamma = \sum_{e \in E} \lambda_e$ and face F' can be rewritten as

$$F' = \left\{ (x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in E} \alpha_e^{i'} x_e^{i'} + \sum_{v \in I(S)} \beta_v^i y_v^{i'} = \sum_{e \in E} \lambda_e \right\}.$$

Step v.: Let $(\bar{x}, \bar{y})$ be a feasible solution where all demands in $S \setminus e'$ are assigned to vehicle i', for some arbitrarily chosen demand $e' \in S$. Moreover, consider that e' is the only demand assigned to some arbitrarily chosen vehicle $j \in K \setminus i'$. Since $(\bar{x}, \bar{y}) \in F$, the equality

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E} \alpha_e^{i'} \bar{x}_e^{i'} + \sum_{v \in I(S)} \beta_v^i \bar{y}_v^{i'} = \sum_{e \in E} \lambda_e$$

holds. Now construct another feasible solution $(\hat{x}, \hat{y})$ by exchanging demands e' and some arbitrarily chosen demand $e'' \in S \setminus e'$. Then, $(\hat{x}, \hat{y})$ also belong to F, and the equality

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E} \alpha_e^{i'} \bar{x}_e^{i'} + \sum_{v \in I(S)} \beta_v^i \bar{y}_v^{i'} - \lambda_{e'} + \alpha_{e'}^{i'} - \alpha_{e''}^{i'} + \lambda_{e''} = \sum_{e \in E} \lambda_e.$$

holds. Therefore, $\alpha_{e'}^{i'} - \lambda_{e'} = \alpha_{e''}^{i'} - \lambda_{e''}$. Since demands e' and e'' can be both arbitrarily chosen among S and $S \setminus e'$, respectively, one has that the difference $\alpha_e^{i'} - \lambda_e$ is constant for any $e \in S$. Let $\omega \in \mathbb{R}$ denote this constant.

Hence, $\alpha_e^{i'} = \lambda_e + \omega$ for any $e \in S$ and face F' can be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e x_e^i + \sum_{e \in E \setminus S} \alpha_e^{i'} x_e^{i'} + \sum_{e \in S} (\lambda_e + \omega) x_e^{i'} + \sum_{v \in I(S)} \beta_v^i y_v^{i'} = \sum_{e \in E} \lambda_e \right\}.$$

Step vi. is organized in three stages.

Step vi.a: Let us first focus on demands in $E \setminus S$ that are not incident to the internal vertices of $G[S]$, denoted by $I(S)$. Let $(\bar{x}, \bar{y})$ be a feasible where vehicle i' is perfectly idle. By definition, $\sum_{e \in S} \bar{x}_e^{i'} = \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) \bar{y}_v^{i'} = 0$. Therefore, $(\bar{x}, \bar{y}) \in F$ and the equality

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i = \sum_{e \in E} \lambda_e$$

holds. Now construct another feasible solution $(\hat{x}, \hat{y})$ by transferring some arbitrarily chosen demand $e' \in E \setminus S$ that is not incident to any node in $I(S)$ from some vehicle $i \in K \setminus i'$ to vehicle i'. The new solution also belongs to F, since $\sum_{e \in S} \hat{x}_e^{i'} = \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) \hat{y}_v^{i'} = 0$, and hence

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \alpha_{e'}^{i'} - \lambda_{e'} = \sum_{e \in E} \lambda_e.$$

Therefore, $\alpha_{e'}^{i'} = \lambda_{e'}$ for any demand $e' \in E \setminus S$ that is not incident to $I(S)$, since demand e' can be arbitrarily chosen.

Step vi.b: Next, let us consider demands in $E \setminus S$ that are incident to exactly one node in $I(S)$. For this, let e' be an arbitrarily chosen demand in $E \setminus S$ that is incident to exactly one node $v' \in I(S)$. Then, choose a leaf node $r \in V[S])$ to be the root of tree $G[S]$, such that v' is not a child of r. Notice that this operation is always possible since $G[S]$ is not a star. Let $T_{v'}$ denote the subtree of $G[S]$ rooted in v' and let $(\bar{x}, \bar{y})$ be a feasible solution where all demands in $E(T_{v'})$ – and only demands in $E(T_{v'})$ – are assigned to vehicle i'. Notice that, by definition,

$$\sum_{v \in V(T_{v'})} (\deg_{T_{v'}}(v) - 1) = 2|E(T_{v'})| - |V(T_{v'})| = |E(T_{v'})| - 1.$$

Since v' has a parent in $G[S]$,

$$\sum_{v \in V(T_{v'})} (\deg_{G[S]}(v) - 1) = 1 + \sum_{v \in V(T_{v'})} (\deg_{T_{v'}}(v) - 1) = |E(T_{v'})|.$$

Therefore,

$$\sum_{e \in S} \bar{x}_e^{i'} = \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1)\bar{y}_v^{i'} = |E(T_{v'})|,$$

and hence, $(\bar{x}, \bar{y}) \in F$ and the equality

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E \setminus S} \alpha_e^{i'} \bar{x}_e^{i'} + \sum_{e \in S} (\lambda_e + \omega)\bar{x}_e^{i'} + \sum_{v \in I(S)} \beta_v^i \bar{y}_v^{i'} = \sum_{e \in E} \lambda_e$$

holds. Now construct another feasible solution $(\hat{x}, \hat{y})$ by transferring demand e' from some vehicle $i \in K \setminus i'$ to vehicle i'. Such operation is only possible since $|E(T_{v'})| < C$. The new solution also belongs to F, since $\sum_{e \in S} \hat{x}_e^{i'} = \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1)\hat{y}_v^{i'} = |E(T_{v'})|$, and hence

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E \setminus S} \alpha_e^{i'} \bar{x}_e^{i'} + \sum_{e \in S} (\lambda_e + \omega)\bar{x}_e^{i'} + \sum_{v \in I(S)} \beta_v^i \bar{y}_v^{i'} + \alpha_{e'}^{i'} - \lambda_{e'} = \sum_{e \in E} \lambda_e.$$

Therefore, $\alpha_{e'}^{i'} = \lambda_{e'}$ for any demand $e' \in E \setminus S$ that is incident to exactly one node in $I(S)$, since demand e' can be arbitrarily chosen.

Step vi.c: Finally, let us consider demands in $E \setminus S$ that are incident to two nodes in $I(S)$. For this, let $e' = u'v'$ be an arbitrarily chosen demand in $E \setminus S$ such that $u' \in I(S)$ and $v' \in I(S)$. Suppose first that e' is not parallel to any demand in S. Then, there exists a unique path P in $G[S]$ from u' to v', with at least three nodes. Let $I(P) = \{v \in V(P) : \deg_P(v) \geq 2\}$ denote the set of internal vertices of P. By definition, $I(P) \neq \emptyset$. Choose some node in $I(P)$ to be the root of $G[S]$. Let $(\bar{x}, \bar{y})$

be a feasible solution where all demands in $E(T_{u'}) \cup E(T_{v'})$ – and only demands in $E(T_{u'}) \cup E(T_{v'})$ – are assigned to vehicle i' (see Figure 4.10). Notice that, the same reasoning used in *Step vi.b* can be applied here to show that

$$\sum_{e \in S} \bar{x}_e^{i'} = \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1)\bar{y}_v^{i'} = |E(T_{u'}) \cup E(T_{v'})|.$$

Therefore, $(\bar{x}, \bar{y}) \in F$ and the equality

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E \setminus S} \alpha_e^{i'} \bar{x}_e^{i'} + \sum_{e \in S} (\lambda_e + \omega)\bar{x}_e^{i'} + \sum_{v \in I(S)} \beta_v^i \bar{y}_v^{i'} = \sum_{e \in E} \lambda_e$$

holds. Now construct another feasible solution $(\hat{x}, \hat{y})$ by transferring demand e' from some vehicle $i \in K \setminus i'$ to vehicle i'. Such operation is only possible since $|E(T_{u'}) \cup E(T_{v'})| < C$. The new solution also belongs to F, since $\sum_{e \in S} \hat{x}_e^{i'} = \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1)\hat{y}_v^{i'} = |E(T_{u'}) \cup E(T_{v'})|$, and hence

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E \setminus S} \alpha_e^{i'} \bar{x}_e^{i'} + \sum_{e \in S} (\lambda_e + \omega)\bar{x}_e^{i'} + \sum_{v \in I(S)} \beta_v^i \bar{y}_v^{i'} + \alpha_{e'}^{i'} - \lambda_{e'} = \sum_{e \in E} \lambda_e.$$

Thus, $\alpha_{e'}^{i'} = \lambda_{e'}$ for any demand $e' \in E \setminus S$ that is not parallel to any demand in S but is incident to two nodes in $I(S)$.

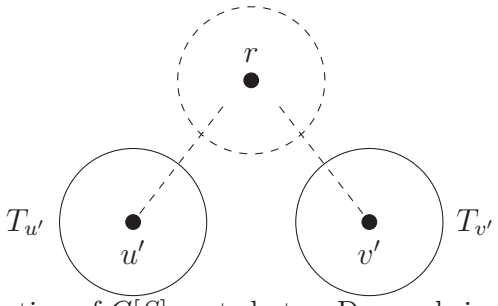


Figure 4.10: Illustration of $G[S]$ rooted at r. Demands in $T_{u'} \cup T_{v'}$ are assigned to vehicle i' on solution $(\bar{x}, \bar{y})$

There remains now only the case where demand e' is incident to two nodes in $I(S)$ and parallel to some demand in S. For this, root $G[S]$ from some node r in $I(S) \setminus \{u', v'\}$. Such node exists from condition ii.. *W.l.o.g.* let u' be the end-node of e'' that is closest to r. Let $(\bar{x}, \bar{y})$ be a feasible solution where all demands in $E(T_{u'})$ – and only demands in $E(T_{u'})$ – are assigned to vehicle i'. Once again, by definition, $(\bar{x}, \bar{y}) \in F$ since $\sum_{e \in S} \bar{x}_e^{i'} = \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1)\bar{y}_v^{i'} = |E(T_{u'})|$, and hence the equality

$$\sum_{e \in E} \sum_{i \in K \setminus i'} \lambda_e \bar{x}_e^i + \sum_{e \in E \setminus S} \alpha_e^{i'} \bar{x}_e^{i'} + \sum_{e \in S} (\lambda_e + \omega)\bar{x}_e^{i'} + \sum_{v \in I(S)} \beta_v^i \bar{y}_v^{i'} = \sum_{e \in E} \lambda_e$$

holds. Now construct another feasible solution $(\hat{x}, \hat{y})$ by transferring demand e' from some vehicle $i \in K \setminus i'$ to vehicle i'. Such operation is only possible since $|E(T_{u'})| < C$. The new solution also belongs to F, and hence

$$\sum_{e\in E}\sum_{i\in K\setminus i'} \lambda_e \bar{x}_e^i + \sum_{e\in E\setminus S} \alpha_e^{i'}\bar{x}_e^{i'} + \sum_{e\in S}(\lambda_e+\omega)\bar{x}_e^{i'} + \sum_{v\in I(S)} \beta_v^i \bar{y}_v^{i'} + \alpha_{e'}^{i'} - \lambda_{e'} = \sum_{e\in E}\lambda_e.$$

Therefore, $\alpha_{e'}^{i'} = \lambda_{e'}$ for any demand $e' \in E \setminus S$ that is incident to two nodes in $I(S)$ and parallel to some demand in S. This finishes *Step vi.* and one can rewrite face F' as

$$F' = \left\{(x,y) \in \mathcal{P}_{(V,E,C)} : \sum_{e\in E}\sum_{i\in K} \lambda_e x_e^i + \sum_{e\in S}\omega x_e^{i'} + \sum_{v\in I(S)} \beta_v^i y_v^{i'} = \sum_{e\in E}\lambda_e\right\}.$$

Step vii.: Choose some leaf node in $V(G[S])$ to be the root of $G[S]$, and let v' be the parent of the deepest leaf in $G[S]$. Then, let $(\bar{x}, \bar{y})$ be a feasible solution where all demands in $E(T_{v'})$ – and only demands in $E(T_{v'})$ – are assigned to vehicle i'. Notice that $T_{v'}$ is a star composed by the children of v' and hence $|E(T_{v'})| = (\deg_{G[S]}(v')-1)$. Therefore, $(\bar{x}, \bar{y}) \in F$ and the equality

$$\sum_{e\in E\setminus E(T_{v'})}\sum_{i\in K\setminus i'} \lambda_e \bar{x}_e^i + \sum_{e\in E(T_{v'})}(\lambda_e+\omega) + \beta_{v'}^{i'} = \sum_{e\in E}\lambda_e$$

holds. By doing the a sequential permutation $(p-2)$ times over $K \setminus i'$ (just as in *Step iv.*), one obtains

$$(p-1)\sum_{e\in E\setminus E(T_{v'})}\lambda_e + (p-1)\sum_{e\in E(T_{v'})}(\lambda_e+\omega) + (p-1)\beta_{v'}^{i'} = (|K|-1)\sum_{e\in E}\lambda_e,$$

which reduces to

$$\beta_{v'}^{i'} = -\sum_{e\in E(T_{v'})}\omega = -|E(T_{v'})|\omega = -(\deg_{G[S]}(v')-1)\omega.$$

As a result, the proof for $\beta_{v'}^{i'}$, where v' is the parent of the deepest leaf in $G[S]$, is finished. Notice however, that a subtree T_v of $G[S]$ can be defined as the union of its children subtrees and the edges that ties v to its children, that is,

$$T_v = \bigcup_{i\in \mathrm{ch}(v)} \{T_i \cup vi\}.$$

Figure 4.11 illustrates such composition. This notion can then be used to climb up the tree $G[S]$ until the unique child of the root node r (recall that r is chosen to be

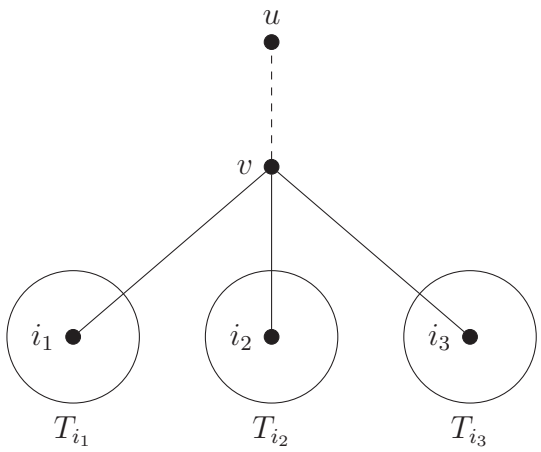


Figure 4.11: Illustration of T_v, where v has 3 children and u is the parent of v

a leaf node). By doing so, the value of $\beta_v^{i'}$, for each $v \in I(T)$, is recursively deduced. For any node $v' \in I(S)$, one obtains

$$\beta_{v'}^{i'} + \sum_{i \in \text{ch}(v')} \sum_{v \in V(T_i)} \beta_v^{i'} = -(\deg_{G[S]}(v') - 1)\omega - \sum_{i \in \text{ch}(v')} \sum_{v \in V(T_i)} (\deg_{G[S]}(v) - 1)\omega.$$

Therefore, $\beta_v^{i'} = -(\deg_{G[S]}(v) - 1)\omega$ for any $v \in V[S]$, and face F' can finally be rewritten as

$$F' = \left\{ (x, y) \in \mathcal{P}_{(V,E,C)} : \sum_{e \in E} \sum_{i \in K} \lambda_e x_e^i + \sum_{e \in S} \omega x_e^{i'} - \sum_{v \in V[S]} \omega(\deg_{G[S]}(v) - 1) y_v^{i'} = \sum_{e \in E} \lambda_e \right\}$$

which concludes the proof. ■

4.4.3 Generalizing k-cardinality tree inequalities

In order to construct a *k-cardinality tree* inequality, one must look for an edge-set S of $C + 1$ demands intersecting a given station $v \in V$ (*i.e.*, $S \subseteq \Delta_E(v)$) such that S spans a tree (*i.e.*, $G[S]$ is a $(C + 1)$-cardinality tree). Nonetheless, such structure might not be available for some instances of U-SNP. If this is the case, it is possible that none of the reinforcing valid inequalities presented so far is able to strengthen the formulation. Example 4.4 illustrates an instance where such case appears.

Example 4.4 - *k-cardinality tree* inequalities can fail to reinforce the given formulation

Consider the instance $\mathcal{I}$ of U-SNP with $C = 2$ depicted in Figure 4.12.

Notice that in the given instance $\mathcal{I}$, there exists no set $S \subseteq \Delta_E(v)$ spanning a $(C+1)$-cardinality tree, for any $v \in V$. This means that none of the presented valid inequalities can be used to reinforce the formulation. Therefore, Theorem 4.2 still

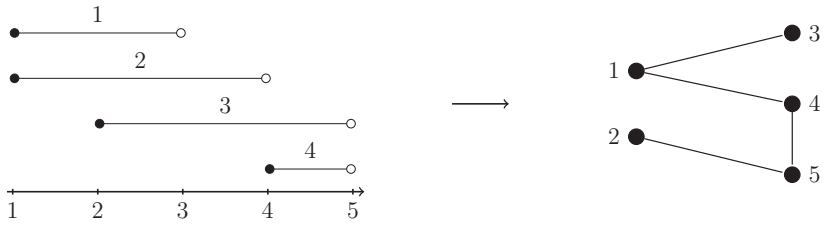
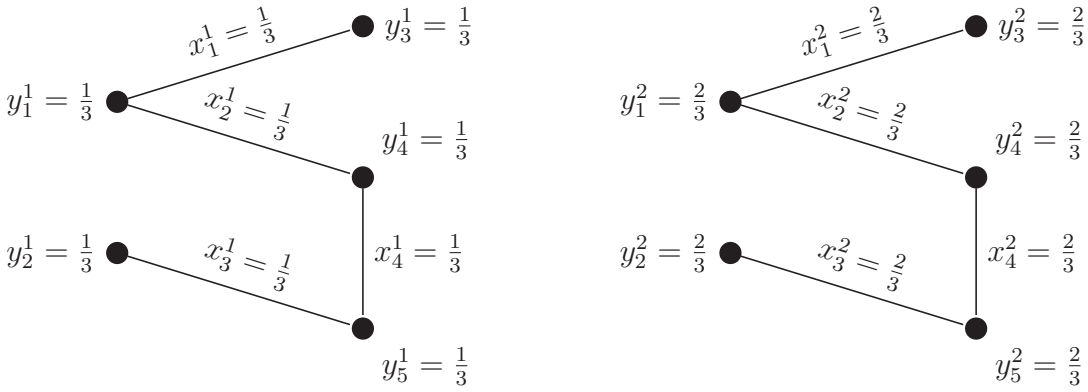


Figure 4.12: An instance where there is no k-cardinality tree inequality

applies, and hence the linear relaxation finds a fractional solution of cost 5. Such fractional solution is depicted below, where only two vehicles are used.



(a) Vehicle 1 **(b)** Vehicle 2

Figure 4.13: Illustration of the obtained fractional solution

Nonetheless, it is easy to see that any feasible solution should stop at least 6 times. □

Indeed, the idea behind *k-cardinality tree* inequalities is to search for a "somehow" connected structure that violates the vehicle's capacity at some given station. On *k-cardinality tree* inequalities such structure is requested to be "fully connected" – recall that $G[S]$ is a tree – and "fully contained" in some intersection point – recall that $S \subseteq \Delta_E(v)$ for a given $v \in V$. Two questions emerges directly from this remark:

i. whether or not one can also consider "partially connected" structures;

ii. whether or not one can also consider structures that are only "partially contained" in some intersection point.

In order to answer such questions, two generalizations of *k-cardinality tree* inequalities are suggested. Each of these generalizations are capable of cutting off the fractional solution proposed in Example 4.4.

Considering partially connected structures

A generalization of *k-cardinality tree* inequalities for taking into account forests instead of trees is proposed below. For this, let us define a *k-cardinality forest* as an acyclic graph F such that $|E(F)| = k$.

Theorem 4.13 - Validity of k-cardinality forest inequalities
The following k-cardinality forest inequalities are valid for $\mathcal{P}_{(V,E,C)}$:

$$\sum_{e\in S} x_e^i - \sum_{v\in V[S]} (\deg_{G[S]}(v) - 1)y_v^i \leq q - 1 \qquad \forall i \in K, j \in V, S \subseteq \Delta_E(j), \tag{4.44}$$

such that $G[S]$ is a $(C+1)$-cardinality forest composed by $1 \leq q \leq C$ connected components.

Proof. The proof of validity is once again done by showing that inequalities (4.44) can be seen as Chvátal-Gomory cuts. More precisely, we show that given a (C+1)-cardinality forest $F = G[S]$ composed by q trees, each denoted by T^k for $1 \leq k \leq q$, a conical combination of inequalities (4.3) and (4.4), yields the inequality

$$(C+q)\sum_{k=1}^{q}\left(\sum_{e\in E(T^k)} x_e^i\right) - (C+1)\sum_{k=1}^{q}\left(\sum_{v\in V(T^k)} (\deg_{T^k}(v) - 1)y_v^i\right) \leq qC,$$

which can be divided by $C+1$. After rounding down the coefficients, one obtains

$$\left\lfloor \frac{C+q}{C+1} \right\rfloor \sum_{e\in E(F)} x_e^i - \sum_{v\in V(F)} (\deg_F(v) - 1)y_v^i \leq \left\lfloor \frac{qC}{C+1} \right\rfloor. \tag{4.45}$$

Since $1 \leq q \leq C$, one has that $C+1 \leq C+q \leq 2C$ and hence

$$\left\lfloor \frac{C+q}{C+1} \right\rfloor = 1.$$

Moreover,

$$\left\lfloor \frac{qC}{C+1} \right\rfloor = \left\lfloor \frac{(C+1)q - q}{C+1} \right\rfloor = \left\lfloor q - \frac{q}{C+1} \right\rfloor = q - 1.$$

Therefore, inequality (4.45) reduces to

$$\sum_{e\in E(F)} x_e^i - \sum_{v\in V(F)} (\deg_F(v) - 1)y_v^i \leq q - 1.$$

Such conical combination is constructed as follows. We arbitrarily choose a leaf node $r \in V(T^k)$ to be the root of T^k, for $k = 1, \ldots, q$. For each edge $e = uv \in E(T^k)$, such that $u \prec v$ (remark that $v \neq r$), multiply inequality $x_e^i - y_u^i \leq 0$ by $(C - |E(T_v^k)|)$

and inequality $x_e^i - y_v^i \leq 0$ by $|E(T_v^k)|$, where T_v^k is the subtree of T^k rooted in v (*cf.* Section 1.1). Likewise, consider the inequality

$$\sum_{e \in E(F)} x_e^i \leq C,$$

which is dominated by (4.3) since $E(F) \subseteq \Delta_E(j)$, then multiply it by q. Summing up all such inequalities yields a new valid inequality where each variable x_e^i for $e \in E(F)$ has coefficient $C + q$. Furthermore, on the right-hand side the independent term equals qC.

To finish the proof, let us show that for any $v \in V(F)$, variable y_v^i has coefficient $-((C+1)(\deg_F(v) - 1))$. For this, consider a node $u \in V(T^k)$. By definition u has $(\deg_F(u) - 1)$ children and each child v of u contributes to the final coefficient of y_u^i with $-(C - |E(T_v^k)|)$. Therefore, the total contribution coming from its children equals

$$-C(\deg_F(u)-1)+(|E(T_u^k)|-(\deg_F(u)-1)) = -((C+1)(\deg_F(u)-1))+|E(T_u^k)|.$$

Finally, since the parent of u contributes with $-|E(T_u^k)|$, the proof is done. ■

Corollary 4.3 - Chvàval-Gomory inequalities of rank 1
The k-cardinality forest inequalities (4.44) *are Chvàval-Gomory inequalities of rank 1.* ■

Notice that, considering again instance $\mathcal{I}$ depicted in Example 4.4, the inequalities

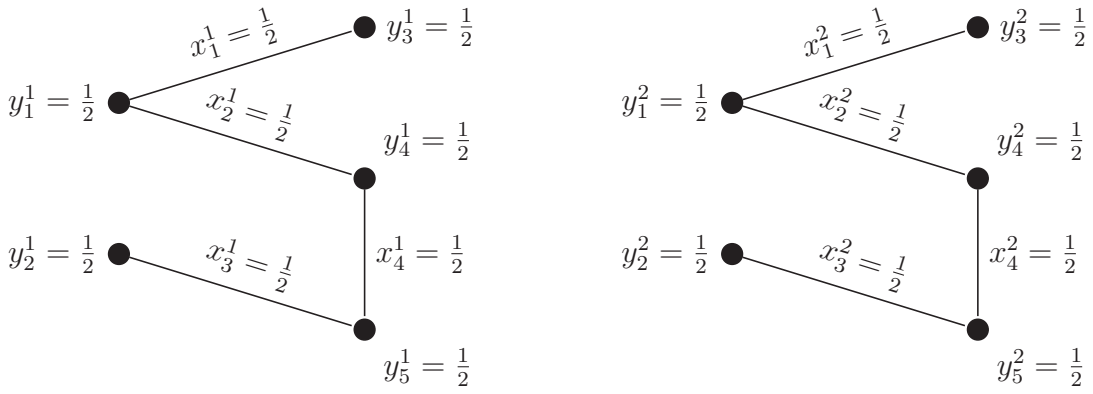
$$x_1^i + x_2^i + x_3^i - y_1^i \leq 1 \qquad \forall\, i \in K, \tag{4.46}$$

are *k-cardinality forest* inequalities capable of cutting off the fractional solution proposed in Figure 4.13. Moreover, such inequalities are actually facet-defining[1] for $\mathcal{P}_\mathcal{I}$. The inclusion of such inequalities, however, does not close the integrality gap. In fact, a new fractional solution of same objective-function value is obtained. Figure 4.14 illustrates such solution. In order, to close this integrality gap, a different generalization of *k-cardinality tree* inequalities is required as one shall see next.

Considering partially intersected structures

On *k-cardinality tree* inequalities (4.40), the edge-set S is required to be a subset of $\Delta_E(v)$ for some given $v \in V$. This condition is imposed in order to prevent the

[1]This can be easily verified with PORTA (see Christof et al. [30]). PORTA is a free software, distributed under the GNU General Public Licence, containing a collection of routines developed by for analyzing polytopes and polyhedrons.



(a) Vehicle 1 **(b)** Vehicle 2

Figure 4.14: Illustration of the fractional solution obtained after the inclusion of *k-cardinality forest* inequalities.

existence of a feasible solution where every demand in S is assigned to the same vehicle (recall that $|S| = C + 1$). Notice however that this same result can be achieved by imposing that $S \subseteq E$ is a subset of demands covering $C+1$ demands in $\Delta_E(v)$, that is, $|S \cap \Delta_E(v)| = C+1$. A generalization of *k-cardinality tree* inequalities for taking into account such relaxed condition is proposed below.

Theorem 4.14 - Validity of k-cover tree inequalities
The following k-cover tree inequalities are valid for $\mathcal{P}_{(V,E,C)}$:

$$\sum_{e \in S} x_e^i - \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) y_v^i \leq 0 \qquad \forall i \in K, j \in V, S \subseteq E, \tag{4.47}$$

such that $G[S]$ is a tree and $|S \cap \Delta_E(j)| = C + 1$.

Proof. The proof of validity is done by showing that inequalities (4.47) can be seen as Chvátal-Gomory cuts. More precisely, we show that given a subset $S \subseteq E$ such that $T = G[S]$ is a tree and $|S \cap \Delta_E(j)| = C+1$, a conical combination of inequalities (4.3), (4.4) and (4.13) yields the inequality

$$|S| \sum_{e \in E(T)} x_e^i - |S| \sum_{v \in V(T)} (\deg_T(v) - 1) y_v^i \leq C + |S \setminus \Delta_E(j)|,$$

which can be divided by $|S|$ and the independent term $\frac{C+|S \setminus \Delta_E(j)|}{|S|}$ rounded down to obtain

$$\sum_{e \in E(T)} x_e^i - \sum_{v \in V(T)} (\deg_T(v) - 1) y_v^i \leq \left\lfloor \frac{C + |S \setminus \Delta_E(j)|}{|S|} \right\rfloor. \tag{4.48}$$

Since $|S \cap \Delta_E(j)| = C + 1$, one has that $C + |S \setminus \Delta_E(j)| = |S| - 1$, and therefore,

inequality (4.48) reduces to

$$\sum_{e \in E(T)} x_e^i - \sum_{v \in V(T)} (\deg_T(v) - 1) y_v^i \leq 0.$$

Such conical combination is constructed as follows. We arbitrarily choose a leaf node $r \in V(T)$ to be the root of T. For each edge $e = uv \in E(T)$, such that $u \prec v$ (remark that $v \neq r$), multiply inequality $x_e^i - y_u^i \leq 0$ by $(|S| - 1 - |E(T_v)|)$ and inequality $x_e^i - y_v^i \leq 0$ by $|E(T_v)|$, where T_v is the subtree rooted in v (*cf.* Section 1.1). Likewise, consider the inequality

$$\sum_{e \in E(T) \cap \Delta_E(j)} x_e^i \leq C.$$

that is dominated by (4.3). At last, consider inequalities (4.13)

$$x_e^i \leq 1,$$

for each $e \in E(T) \setminus \Delta_E(j)$. Summing up all such inequalities yields a new valid inequality where each variable x_e^i for $e \in E(T)$ has coefficient $|S|$. Furthermore, on the right-hand side the independent term equals $C + |S \setminus \Delta_E(j)|$.

To finish the proof, let us show that for any $v \in V(T)$, variable y_v^i has coefficient $-|S|(\deg_T(v) - 1)$. Consider a node $u \in V(T)$. By definition, u has $(\deg_T(u) - 1)$ children and each child v of u contributes to the final coefficient of y_u^i with $-(|S| - 1 - |E(T_v)|)$. Therefore, the total contribution coming from its children equals

$$-(|S| - 1)(\deg_F(u) - 1) + (|E(T_u)| - (\deg_F(u) - 1)) = -|S|(\deg_T(u) - 1) + |E(T_u)|.$$

Finally, since the parent of u contributes with $-|E(T_u)|$, the proof is done. ■

Corollary 4.4 - Chvàval-Gomory inequalities of rank 1
The k-cover tree inequalities (4.47) *are Chvàval-Gomory inequalities of rank 1.* ■

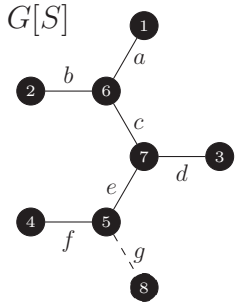
We next investigate conditions that are necessary to hold for *k-cover tree* inequalities (4.47) to define facets of $\mathcal{P}_{(V,E,C)}$.

Let $G[S]$ be a tree such that $|S \cap \Delta_E(j)| = C + 1$, for some $j \in V$. Notice that if $G[S]$ has a leaf edge e such that $e \notin \Delta_E(j)$, then the k-cover tree inequalities associated with $G[S]$ are always dominated by the k-cover tree inequalities associated with $G[S \setminus e]$. Indeed, the former can be written as a sum of

i. stop inequalities (4.4) and

ii. the k-cover tree inequality associated with $G[S \setminus e]$.

Figure 4.15 illustrates such case by depicting an instance where $C = 3$, $G[S]$ is a tree covering $C + 1$ demands in $\Delta_E(4)$ and g is a leaf edge in $S \setminus \Delta_E(4)$.



$$x_g^i - y_5^i \leq 0 \ [i]$$
$$x_a^i + x_b^i + x_c^i + x_d^i + x_e^i + x_f^i - y_5^i - 2y_6^i - 2y_7^i \leq 0 \ [ii.]$$
$$\overline{x_a^i + x_b^i + x_c^i + x_d^i + x_e^i + x_f^i + x_g^i - 2y_5^i - 2y_6^i - 2y_7^i \leq 0}$$

Figure 4.15: How a k-cover tree gets dominated.

This yields a necessary condition to be satisfied by a facet-defining k-cover tree inequality: every leaf edge of $G[S]$ must be in $\Delta_E(j)$. However, such condition is not sufficient. In fact, the k-cover tree inequalities induced by $G[S \setminus g]$ in Figure 4.15 satisfy such condition but are dominated by the k-cover tree inequalities induced by $G[S \setminus \{f, g\}]$. Notice that $G[S \setminus \{f, g\}]$ does not cover $C + 1$ demands in $\Delta_E(4)$, but it does in $\Delta_E(5)$. Therefore, the previously condition established can be further improved as follows. A k-cover tree inequality is facet-defining *only if* the associated tree $G[S]$ is not included in any other tree $G[S']$ associated with another valid k-cover tree inequality.

With this in mind, consider again instance $\mathcal{I}$ depicted in Example 4.4. The following inequalities are *k-cover tree* inequalities capable of cutting off the fractional solution proposed in Figure 4.13 obtained after the inclusion of the *k-cardinality forest* inequalities

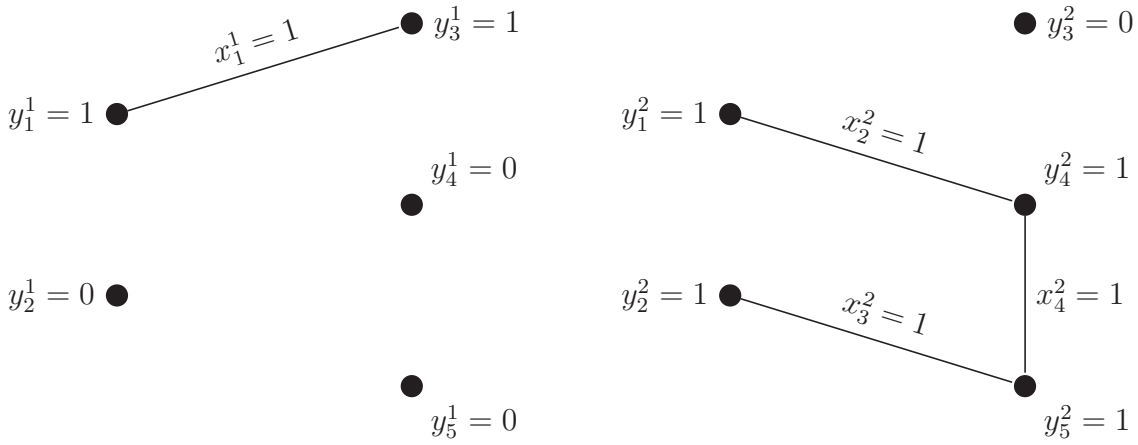
$$x_1^i + x_2^i + x_3^i + x_4^i - y_1^i - y_4^i - y_5^i \leq 0 \qquad \forall\ i \in K. \tag{4.49}$$

Moreover, such inequalities are actually facet-defining[2] for $\mathcal{P}_{\mathcal{I}}$ and their inclusion not only closes the integrality gap but also provides an integer solution of optimal cost as depicted in Figure 4.16.

Combining both generalizations

As expected, one can now combine the two previously presented generalizations of *k-cardinality tree* inequalities, giving rise to the so-called *k-cover forest* inequalities.

[2]Verified with PORTA software (see Christof et al. [30]).



(a) Vehicle 1 **(b)** Vehicle 2

Figure 4.16: Illustration of the integer solution obtained after the inclusion of *k-cover tree* inequalities.

Theorem 4.15 - Validity of k-cover forest inequalities
The following k-cover forest inequalities are valid for $\mathcal{P}_{(V,E,C)}$:

$$\sum_{e \in S} x_e^i - \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) y_v^i \leq q - 1 \qquad \forall i \in K, j \in V, S \subseteq E \tag{4.50}$$

such that $G[S]$ is a forest composed by $1 \leq q \leq |S|$ connected components and $|S \cap \Delta_E(j)| = C + 1$.

Proof. The proof of validity is once again done by showing that inequalities (4.50) can be seen as Chvátal-Gomory cuts. More precisely, we show that, given a forest $F = G[S]$ composed by q trees, each denoted by T^k for $1 \leq k \leq q$, and such that $|S \cap \Delta_E(j)| = C + 1$, a conical combination of inequalities (4.3), (4.4) and (4.14), yields the inequality

$$(|S|-1+q) \sum_{k=1}^{q} \left(\sum_{e \in E(T^k)} x_e^i \right) - |S| \sum_{k=1}^{q} \left(\sum_{v \in V(T^k)} (\deg_{T^k}(v) - 1) y_v^i \right) \leq qC + q|S \setminus \Delta_E(j)|,$$

which can be divided by $|S|$. After rounding down the coefficients one obtains

$$\left\lfloor \frac{|S| - 1 + q}{|S|} \right\rfloor \sum_{e \in E(F)} x_e^i - \sum_{v \in V(F)} (\deg_F(v) - 1) y_v^i \leq \left\lfloor \frac{qC + q|S \setminus \Delta_E(j)|}{|S|} \right\rfloor. \tag{4.51}$$

Since $1 \leq q \leq |S|$, one has that $|S| \leq |S| - 1 + q \leq 2|S| - 1$ and hence,

$$\left\lfloor \frac{|S| - 1 + q}{|S|} \right\rfloor = 1.$$

Moreover, since $|S \cap \Delta_E(j)| = C + 1$,

$$\left\lfloor \frac{q(C + |S \setminus \Delta_E(j)|)}{|S|} \right\rfloor = \left\lfloor \frac{q(|S| - 1)}{|S|} \right\rfloor = \left\lfloor \frac{q|S| - q}{|S|} \right\rfloor = \left\lfloor q - \frac{q}{|S|} \right\rfloor = q - 1.$$

Therefore, inequality (4.51) reduces to

$$\sum_{e \in E(F)} x_e^i - \sum_{v \in V(F)} (\deg_F(v) - 1) y_v^i \leq q - 1. \tag{4.52}$$

Such conical combination is constructed as follows. We arbitrarily choose a leaf node $r \in V(T^k)$ to be the root of T^k, for $k = 1, \ldots, q$. For each edge $e = uv \in E(T^k)$, such that $u \prec v$ (remark that $v \neq r$), multiply inequality $x_e^i - y_u^i \leq 0$ by $(|S| - 1 - |E(T_v^k)|)$ and inequality $x_e^i - y_v^i \leq 0$ by $|E(T_v^k)|$, where T_v^k is the subtree of T^k rooted in v (*cf.* Section 1.1). Likewise, consider the inequality

$$\sum_{e \in E(F) \cap \Delta_E(j)} x_e^i \leq C,$$

which is dominated by (4.3), and multiply it by q. At last, consider inequalities (4.13)

$$x_e^i \leq 1,$$

for each $e \in E(F) \setminus \Delta_E(j)$ and multiply each by q. Summing up all such inequalities yields a new valid inequality where each variable x_e^i for $e \in E(F)$ has coefficient $|S| - 1 + q$. Furthermore, on the right-hand side the independent term equals $q(C + |S \setminus \Delta_E(j)|)$.

To finish the proof, let us show that variable y_v^i has coefficient $-|S|(\deg_F(v) - 1)$ for any $v \in V(F)$. Consider a node $u \in V(T^k)$. By definition u has $(\deg_F(u) - 1)$ children and each child v of u contributes to the final coefficient of y_u^i with $-(|S| - 1 - |E(T_v^k)|)$. Therefore, the total contribution coming from its children equals $-|S|(\deg_T(u) - 1) + |E(T_u^k)|$. Finally, since the parent of u contributes with $-|E(T_u^k)|$, the proof is done. ■

4.4.4 Girth inequalities

So far, only the sparsest structures – such as stars, trees and forests – have been exploited to derive valid inequalities for $\mathcal{P}_{(V,E,C)}$. This issue is now addressed, by looking at some denser structures.

In graph theory, the girth of a graph G corresponds to the length of the smallest cycle contained in G. In other words, a graph is said to have girth k if it contains

no cycle of size smaller than k. Suppose $G = (V, E)$ has girth k. Then, $G[S]$ is a forest for any $S \subseteq E$ such that $|S| \leq k - 1$. Based on such remark, a new family of valid inequalities, named *girth* inequalities, is introduced here below.

Theorem 4.16 - Validity of girth inequalities
The following girth inequalities are valid for $\mathcal{P}_{(V,E,C)}$*:*

$$\sum_{e \in S}(C+1)x_e^i - \sum_{v \in V[S]} Cy_v^i \leq 0 \qquad \forall i \in K, j \in V, S \subseteq \Delta_E(j), \tag{4.53}$$

such that $G[S]$ *has girth greater than or equal to* $C + 1$*.*

Proof. Let $(\bar{x}, \bar{y}) \in \mathcal{P}_{(V,E,C)} \cap \mathbb{Z}^{(n+m)p}$ represent an arbitrary feasible solution of U-SNP. Moreover, let $S_i \subseteq S$ denote the set of demands in S that are assigned to vehicle i on solution $(\bar{x}, \bar{y})$. If $|S_i| = 0$, then $\sum_{e \in S}(C+1)\bar{x}_e^i = 0$ and (4.53) is obviously valid. Hence, assume $|S_i| \geq 1$. By definition,

$$\sum_{e \in S}(C+1)\bar{x}_e^i = \sum_{e \in S_i}(C+1) = C|S_i| + |S_i|.$$

Since $|S_i| \leq C$ and $G[S]$ has girth greater than or equal to $C + 1$, $G[S_i]$ must be a forest and hence vehicle i stops on at least $|S_i| + 1$ stations in $V[S]$. Therefore,

$$\sum_{v \in V[S]} C\bar{y}_v^i \geq C|S_i| + C.$$

Finally, one has that

$$\sum_{e \in S}(C+1)\bar{x}_e^i = C|S_i| + |S_i| \leq C|S_i| + C \leq \sum_{v \in V[S]} C\bar{y}_v^i,$$

which proves inequality (4.53) is valid. ■

Notice that from the introduction of *girth* inequalities, one may derive new Chvàtal-Gomory cuts. For this, let $S \subseteq \Delta_E(j)$ for some $j \in V$ be a subset of edges such that $G[S]$ has girth greater than or equal to $C + 1$. Then, the following inequalities belong to the family of *girth* inequalities:

$$\sum_{e \in S}(C+1)x_e^i - \sum_{v \in V[S]} Cy_v^i \leq 0 \qquad \forall i \in K.$$

By summing up such inequalities, one obtains

$$\sum_{v \in V[S]} \sum_{i \in K} Cy_v^i \geq \sum_{e \in S} \sum_{i \in K}(C+1)x_e^i.$$

However, from constraints (4.2) one has that

$$\sum_{e \in S} \sum_{i \in K} (C+1) x_e^i = (C+1)|S|,$$

and therefore,

$$\sum_{v \in V[S]} \sum_{i \in K} y_v^i \geq \frac{(C+1)|S|}{C} = |S| + \frac{|S|}{C}.$$

Since $\sum_{v \in V[S]} \sum_{i \in K} y_v^i$ should be an integer value, one can derive the following Chvàtal-Gomory cuts.

$$\sum_{v \in V[S]} \sum_{i \in K} y_v^i \geq |S| + \left\lceil \frac{|S|}{C} \right\rceil \qquad \forall j \in V, S \subseteq \Delta_E(j), \tag{4.54}$$

such that $G[S]$ has girth greater than or equal to $C+1$.

This means that the introduction of inequalities (4.53) and (4.54) allows the linear relaxation to yield the lower bound presented in Proposition 3.4. Such result is summarized below.

Proposition 4.6 - Reinforcement from girth inequalities
Let $P'_{(V,E,C)} = \{(x,y) \in P_{(V,E,C)} : (x,y)$ *satisfies* $(4.53), (4.54)\}$. *Then, if* (V,E,C) *is an instance of Intersection U-SNP such that* $G = (V,E)$ *has girth greater than or equal to* $C+1$,

$$\min \left\{ \mathbf{0}x + \mathbf{1}y : (x,y) \in P'_{(V,E,C)} \right\} \geq m + \left\lceil \frac{m}{C} \right\rceil.$$

Proof. Inequalities (4.53) and (4.54) are valid for $S = E$. ■

4.4.5 Generating further valid inequalities

In Grötschel et al. [84, 83], the authors study the *linear ordering* polytope and the *acyclic subgraph* polytope, respectively. Both these studies elegantly explore the idea of using a facet-defining inequality of a given polytope $P \subseteq \mathbb{R}^n$ to derive other facet-defining inequalities for a higher-dimension related polytope $Q \subseteq \mathbb{R}^{n+d}$. Notice that for many combinatorial problems, the lower-dimension polytope P can be seen as a face of the higher-dimension polytope Q. This means that the (partial) characterization of a face of a polytope can yield information on how to (partially) characterize other faces of this same polytope. Based on this idea, we show how one may generate new valid inequalities for the U-SNP through the exploitation of the properties of the facet-defining *k-cardinality tree* inequalities (4.40).

Theorem 4.17 - Providing new valid inequalities
Let

$$\sum_{e\in\Delta_E(j)} a_e^i x_e^i + \sum_{v\in V} b_v^i y_v^i \leq r \tag{4.55}$$

be any valid inequality for $\mathcal{P}_{(V,E,C)}$ such that $b_v^i \leq 0$ for any $v \in V$. Given a vehicle $i \in K$, a station $j \in V$ and a C-cardinality tree $G[S]$ for $S \subseteq \Delta_E(j)$, the inequality

$$\sum_{e\in\Delta_E(j)} a_e^i x_e^i + \sum_{v\in V} b_v^i y_v^i + \sum_{e\in S} x_e^i - \sum_{v\in V[S]} (\deg_{G[S]}(v)-1) y_v^i \leq r \tag{4.56}$$

is also valid for $\mathcal{P}_{(V,E,C)}$, if and only if

$$\sum_{e\in S} a_e^i + \sum_{v\in V[S]} b_v^i \leq r-1.$$

Proof. Let $(\bar{x},\bar{y})$ be any feasible solution of $\mathcal{P}_{(V,E,C)}$. Notice that, from Theorem 4.11,

$$\sum_{e\in S} x_e^i - \sum_{v\in V[S]} (\deg_{G[S]}(v)-1) y_v^i \leq 0 \tag{4.57}$$

is a valid inequality for $\mathcal{P}_{(V,E,C-1)}$. With this in mind, if $\sum_{e\in S} \bar{x}_e^i \leq C-1$, then (4.57) is satisfied by $(\bar{x},\bar{y})$. Consequently, $(\bar{x},\bar{y})$ also satisfies (4.56) since (4.55) is valid.

Thus, the only possible way for $(\bar{x},\bar{y})$ to violate (4.56) is if all C demands in S are assigned to vehicle i. This means that

i. $\sum_{e\in S} \bar{x}_e^i = C$,

ii. $\sum_{v\in V[S]} (\deg_{G[S]}(v)-1)\bar{y}_v^i = C-1$,

iii. $\sum_{e\in\Delta_E(j)} a_e^i \bar{x}_e^i = \sum_{e\in S} a_e^i$,

iv. $\sum_{v\in V} b_v^i \bar{y}_v^i = \sum_{v\in V[S]} b_v^i + \sum_{v\in V\setminus V[S]} b_v^i \bar{y}_v^i$.

It follows directly that (4.56) is valid if and only if

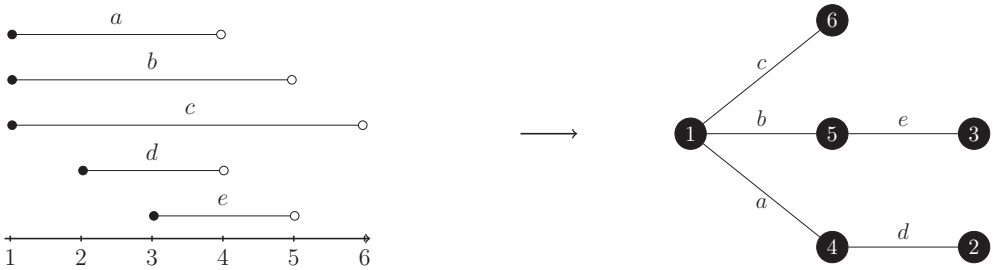
$$\sum_{e\in S} a_e^i + \sum_{v\in V[S]} b_v^i + \sum_{v\in V\setminus V[S]} b_v^i y_v^i \leq r-1.$$

Since $b_v^i \leq 0$ for all $v \in V$, it is clear that $\sum_{v\in V\setminus V[S]} b_v^i \bar{y}_v^i \leq 0$. Moreover, this bound is achieved when $\bar{y}_v^i = 0$ for all $v \in V \setminus V[S]$ and hence, the proof is done. ■

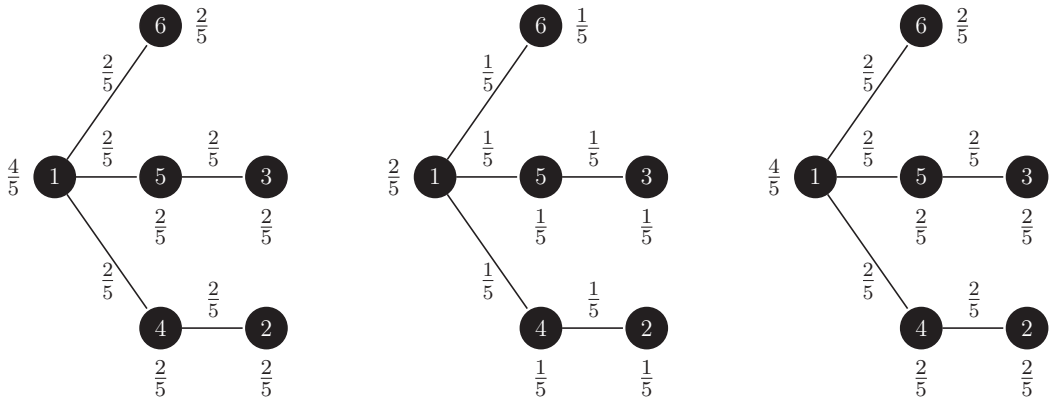
It is curious to remark that *k-cardinality tree* inequalities (4.40) can be obtained through the procedure described in Theorem 4.17 by taking inequalities (4.4) as

the starting valid inequality. It follows that one may now use *k-cardinality tree* inequalities (4.40) as a starting valid inequality to derive new non-redundant valid inequalities. The following example illustrates such situation.

Example 4.5 - Generating new valid inequalities
Consider the following instance $\mathcal{I}$ of U-SNP with $C = 2$.



Even after the addition of *all* valid *k-cardinality tree* inequalities to the formulation, its linear relaxation still finds a fractional solution of cost 7. Such fractional solution is depicted below, where the values of variables x and y are given next to their corresponding edge and node, respectively.



(a) Vehicle 1 **(b)** Vehicle 2 **(c)** Vehicle 3

Figure 4.17: Illustration of the obtained fractional solution

Notice that in the given instance $\mathcal{I}$,

$$x_a^i + x_c^i + x_d^i - y_1^i - y_4^i \leq 0, \qquad \forall\, i \in K, \tag{4.58}$$

are valid *k-cardinality tree* inequalities. Obviously, the solution depicted in Figure 4.17 satisfies such inequalities. Moreover, consider the inequality

$$x_b^i + x_e^i - y_5^i \leq 0, \qquad \forall\, i \in K, \tag{4.59}$$

induced by the C-cardinality tree formed by edges b and e. Inequality (4.59) is clearly not valid for the given instance. Notice however, that if $C = 1$, then inequality

(4.59) becomes valid. With this in mind one may use inequalities (4.58) and (4.59) in Theorem 4.17 to derive the following new valid inequality:

$$x_a^i + x_b^i + x_c^i + x_d^i + x_e^i - y_1^i - y_4^i - y_5^i \leq 0, \qquad \forall\ i \in K. \tag{4.60}$$

Such inequality is capable of cutting off the fractional solution proposed in Figure 4.17. Moreover, the inclusion of inequality (4.60) to the formulation provides an integer solution of optimal cost. □

4.5 Conclusion

In order to solve the U-SNP, we have studied in this chapter an integer programming formulation due to Pimenta et al. [151]. Based on such formulation, we provided two relaxations that are both capable of solving U-SNP. However, such formulation was shown to be particularly weak in the sense that its linear relaxation does not provide any significant information. In other words, the dual (lower) bounds it provides are trivial. In order to reinforce such formulation, we studied the facial structure of polytope $\mathcal{P}_{(V,E,C)}$ defined as the convex hull of the feasible solutions. For this, we have first investigated the dimension of such integer polyhedron as well as the necessary and sufficient conditions under which the inequalities composing the weak formulation from Pimenta et al. [151] are facet-defining. At last, several families of valid inequalities capable of reinforcing the formulation were introduced and the necessary and sufficient facet-defining conditions for some of these families were presented. The computational efficiency obtained by the inclusion of such inequalities is next presented in Chapter 5.

Chapter 5

Computational study

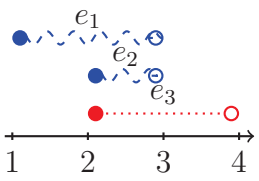
"Computers are useless. They only give you answers."

— Pablo Picasso

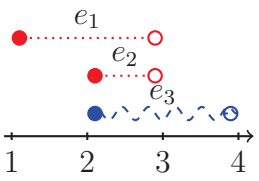
In this chapter a broad computational study on the resolution of U-SNP is conducted and a Branch-and-Cut framework is developed. The formulation $\text{USNP}_{(V,E,C)}$, defined by inequalities (4.1)-(4.6), is taken as an starting point. Following the unsatisfactory performances of such formulation identified in Chapter 4, we propose a series of clever features capable of boosting its performances. These features include i. breaking the symmetry hidden behind U-SNP, ii. eliminating useless variables, iii. relaxing variables, and iv. integrating the reinforcing valid inequalities identified in Section 4.4 into the Branch-and-Cut framework. All the implementations involve the use of the standard MILP solver `ILOG CPLEX 12.8` coupled with `CPLEX Concert Technology` for allowing an interface with CPLEX libraries using C++.

5.1 Symmetry

In Margot [127], a MIP is defined as being symmetric if its variables can be permuted without changing the problem's structure. In our case, since vehicles are considered to be identical, U-SNP hides a complete symmetry with respect to vehicles. In other words, for any given feasible solution, another solution with equivalent objective function value (*i.e.*, the same number of stops) can be obtained by imposing a sequential permutation over any subset of vehicles. This means that the symmetry behind U-SNP results in the existence of $p!$ distinct equivalent solutions for each given assignment of demands to vehicles. Figure 5.1 depicts two symmetric solutions of an instance of U-SNP.



(a) A solution with e_1 and e_2 in vehicle blue and e_3 in vehicle red



(b) A solution with e_1 and e_2 in vehicle red and e_3 in vehicle blue

Figure 5.1: Two symmetric solutions

Numerous authors have highlighted the importance of eliminating – or at least reducing – symmetry when solving MIPs (*e.g.* Sherali and Smith [166], Kaibel and Pfetsch [100], Denton et al. [52], Ostrowski et al. [138]). Indeed, when solving a symmetric MIP through a Branch-and-Bound (or Branch-and-Cut) procedure, one is forced to solve isomorphic subproblems in the enumeration tree which represents an useless replication of efforts. In such scenario, proving optimality of a given solution requires exploring and fathoming each one of its symmetric pairs which can be terribly expensive. Typical combinatorial optimization problems having symmetry issues are Vertex Coloring, Partitioning and Packing problems (see Kaibel and Pfetsch [100]).

5.1.1 Symmetry-breaking methods

Facing symmetry can be done through several ways, each of which having its pros and cons. In this section we present some of the most popular methods capable of (partially) breaking symmetry of a MILP.

Pertubation

One of the first methods that come to mind when tackling symmetry is to perform some perturbation on the coefficients of variables in the objective function. However, according to Margot [127], making small perturbations is often counterproductive when testing for infeasibility or proving optimality. Indeed, if the MILP in question is infeasible, modifying the objective function accomplishes nothing as the Branch-and-Bound enumeration tree remains the same. Moreover, once an optimal solution is known, proving its optimality is equivalent to verifying the nonexistence of a better feasible solution, which requires the same amount of work as verifying a MILP's infeasibility. On the other hand, making huge perturbations can easily lead to numerical instability.

Isomorphism Pruning and Orbital Branching

A more refined approach consists on trying to break symmetry during the optimization process. This can be done either by pruning identified isomorphic branches of the Branch-and-Bound tree or by fixing variables in order to avoid the creation of such isomorphic nodes in the Branch-and-Bound tree. When it comes to pruning identified isomorphic branches, the works of Margot [125, 126] presenting *Isomorphism Pruning* and Ostrowski et al. [139] introducing *Orbital Branching* truly excel. Isomorphism Pruning is a method allowing to detect isomorphic nodes on the enumeration tree - and hence prune them - for a general ILP. Nonetheless, this symmetry detection phase can require a significant computational effort. Orbital Branching is a relaxed version of Isomorphism Pruning, in the sense that it does not guarantee the complete elimination of symmetry for a general ILP, whereas Isomorphism Pruning does. In practice however, it remains competitive with respect to Isomorphism Pruning. In addition, for some families of combinatorial problems (such as job scheduling, see Ostrowski et al. [138]) Orbital Branching does completely eliminate symmetry.

Orbitopal Fixing

In Kaibel et al. [101], a linear-time algorithm capable of completely removing symmetry from partition problems, called *orbitopal fixing*, is proposed. At each node of the enumeration tree, the method fixes variables based on a predefined lexicographic order and on the already fixed variables at the current node. Such lexicographic order consists to say $x_1 \geq x_2 \geq \cdots \geq x_n$ for a given group of symmetric variables. With respect to U-SNP, applying such lexicographic order would amount to impose that

i. the i-th demand must be assigned to one of the first i vehicles;

ii. a demand i cannot be assigned to an empty vehicle j if there exists some empty vehicle j' such that $j' < j$.

To illustrate it, consider the following two matrices of variables x, where the lines indicate the demands and the columns represent the vehicles. That is, the value of x_e^i is given by the entry located on line e and column i. Clearly, the first matrix x^{lex} follows the lexicographic order imposed by conditions i. and ii., while the second

one x^{not} does not.

$$x^{lex} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad x^{not} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

In order to test its performance on U-SNP, the method proposed by Kaibel et al. [101] is hence adapted. At each node of the Branch-and-Bound tree, *Orbitopal Fixing* tries to deduce which demands cannot be assigned to which vehicles (that is, to fix the associated variables x_e^i to zero) so that every feasible solution of the node respects the lexicographic order imposed by conditions i. and ii..

In the first part of Orbitopal Fixing one receives as input the list of fixed variables at the current node. Such list is defined by (F_0, F_1), where F_0 is the set of variables that have been fixed 0 and F_1 the set of variables fixed to 1. The algorithm's objective is to return as output an updated list of fixed variables (F_0^*, F_1), where $F_0 \subseteq F_0^*$. For this, let $a \in \{1, \ldots, p\}^m$ be a vector providing an upper bound on the largest indexed vehicle for each of the m demands. That is to say, if $a_e = j$, then demand $e \in E$ must be assigned to one of the first j vehicles. Consequently, x_e^i can be fixed to zero for any $i > j$. Of course, $a_1 = 1$ and, unless $a_{e-1} = p$, one has that $a_e = a_{e-1} + 1$ for any $e \geq 2$, from condition i.. Notice however that condition ii. ensures that if for some reason (*e.g.*, any previous branching decision), variable $x_e^{a_{e-1}+1}$ has been fixed to 0, then $a_e = a_{e-1}$. A formal description of this first part of Orbitopal Fixing is given by Algorithm 2.

Algorithm 2 Fixing Zeros

Input: (F_0, F_1)
Output: (F_0^*, F_1) such that $F_0 \subseteq F_0^*$
  Set $F_0^* = F_0$ and $a_1 = 1$
  for $e = 2, \ldots, m$ **do**
    if $a_{e-1} \neq p$ **and** $x_e^{a_{e-1}+1} \notin F_0$ **then**
      $a_e = a_{e-1} + 1$
    else
      $a_e = a_{e-1}$
    end if
    Set $F_0^* = F_0^* \cup \{x_e^i : i > a_e\}$
  end for

The second part of Orbitopal Fixing consists of fixing variables to one. For this, Algorithm 2 is employed to search for contradictions. More specifically, given vector a constructed in Algorithm 2, let $F_0' = F_0 \cup x_e^{a_e}$, where $x_e^{a_e}$ is a variable that has not been fixed yet. Then, if Algorithm 2 launched with (F_0', F_1) as input, returns a list

of fixed variables (F_0^*, F_1) such that $F_0^* \cap F_1 \neq \emptyset$, it means that fixing variable $x_e^{a_e}$ to zero yields a contradiction, and hence one may fix $x_e^{a_e}$ to one. A formal description of this second part of Orbitopal Fixing is given by Algorithm 3.

Algorithm 3 Fixing Ones

Input: (F_0, F_1) and vector a constructed in Algorithm 2
Output: (F_0', F_1') such that $F_0 \subseteq F_0'$ and $F_1 \subseteq F_1'$
  Set $F_0' = F_0$ and $F_1' = F_1$
  for $e = 2, \ldots, m$ such that $x_e^{a_e} \notin F_1 \cup F_0$ **do**
    $(F_0^*, F_1^*) =$ Algorithm 2 $(F_0' \cup x_e^{a_e}, F_1')$
    if $F_0^* \cap F_1^* \neq \emptyset$ **then**
      Set $F_1' = F_1' \cup x_e^{a_e}$ and $F_0' = F_0' \cup \{x_e^i : i \neq a_e\}$
    end if
  end for

Symmetry-Breaking Constraints

If one prefers to not intervene in the branching process, a final alternative is to introduce symmetry-breaking inequalities to the ILP formulation. Such inequalities transgress the paradigm of valid inequalities. In fact, symmetry-breaking inequalities should not be valid. On the contrary, their purpose is to cut off part of feasible region, while guaranteeing that at least one optimal solution from the original problem remains feasible. Based on the lexicographic order of variables, the *partitioning orbitope* – the polytope obtained as the convex hull of non-symmetric solutions of a partitioning problem – is characterized in Kaibel and Pfetsch [100]. As expected, the linear description of the partitioning orbitope requires an exponential number of inequalities. In Denton et al. [52], a subclass of the inequalities describing the partitioning orbitope is adapted to tackle symmetry in operating room scheduling problems. Although, the inequalities employed are not sufficient to completely remove symmetry, important results were obtained applying only a polynomial number of such inequalities.

The inequalities presented in Denton et al. [52] are hence adapted to fit U-SNP, giving rise to the following symmetry-breaking inequalities:

$$\sum_{i=1}^{e} x_e^i = 1 \qquad \forall e \in E : e < p, \tag{5.1}$$

$$\sum_{j=i}^{\min\{e,p\}} x_e^j - \sum_{u=i-1}^{e-1} x_u^{i-1} \leq 0 \qquad \forall e \in E, i \in K : e \geq i. \tag{5.2}$$

Notice that constraints (5.1) and (5.2) impose exactly the lexicographic order

previously detailed. Indeed, inequalities (5.1) guarantee that the e-th demand must be assigned to one of the first e vehicles. On the other hand, inequalities (5.2) ensures that demand e can only be assigned to vehicle i if at least one of the first $e-1$ demands is assigned to vehicle $i-1$.

5.1.2 Methods comparison

With the intention of improving the Branch-and-Bound performances, Orbitopal Fixing and Symmetry-Breaking Constraints were adapted to fit U-SNP. Notice however that Orbitopal Fixing requires the user to interfere on the branching process of the chosen MIP solver. In practice, this means that the CPLEX branching function has to be overridden by implementing the *BranchCallback* macro. Anywise, overriding such function unables CPLEX to apply a series of default settings it uses, such as dynamic search and dual reductions. Unfortunately, we are unable to give more details on such CPLEX features as they are trade secrets from IBM (*cf.* Junglas [99]).

Therefore, for a fairer comparison, we have implemented a dummy callback (*i.e.*, an empty callback) employed to deactivate such CPLEX features. Notice that such practice is frequent in the literature for proving the efficiency of branching, cutting planes or node selection methods. Such strategy is used in Carvajal et al. [29] where "[...] dummy callbacks were used when CPLEX was executed with default settings. We think this is necessary in order to get a fair comparison, since the only presence of a user callback function changes the underlying behavior of CPLEX." In Sabharwal et al. [162], the authors justify that "we use CPLEX [...] with node and branch 'callbacks' turned on (using empty callbacks) as our baseline CPLEX solver. [...] Note that this causes some features of CPLEX to be turned off (*e.g.*, dynamic search) but is the only way to enhance CPLEX with a custom node selection strategy without access to the internals of CPLEX". In Fischetti and Monaci [68], the procedure is once again reported: "For a fair comparison, all codes use a (possibly dummy) callback function, thus deactivating IBM ILOG CPLEX's proprietary dynamic search." Finally, even if some CPLEX default features are disabled by the implementation of callbacks, the main CPLEX settings remains untouched such as presolving, cut generation and branching variable choice.

Table 5.1 provides the obtained results for instances with the number of demands m ranging from 30 to 60, the capacity C ranging from 2 to 8 and the graph density ρ from 1.5 to 4.5. For each combination of parameters m, C and ρ, only instances $m\_C\_\rho\_1$ are analysed. For a more exhaustive comparison, please refer to Table A.2 in Appendix A. The three following scenarios have been tested:

- Section `Default` concerns the results obtained by CPLEX using a dummy callback function;
- Section `Constraints` concerns the results obtained by CPLEX with the inclusion of symmetry-breaking constraints to the root node of the enumeration tree;
- Section `Orbitopal` concerns the results obtained by CPLEX with the addition of Orbitopal Fixing method for breaking symmetry;

For each scenario, the total time in seconds required for the optimization is reported under column `CPU`. If the time limit of two hours is exceeded, the remaining gap percentage is displayed under column `gap`. The number of nodes (in thousands) explored in the enumeration tree is given under column `node`.

Table 5.1: Comparison between symmetry-breaking methods

Instances			Default			Constraints			Orbitopal		
m	C	ρ	CPU	gap	node	CPU	gap	node	CPU	gap	node
30	2	1.5	7200	35.3	808	7200	3.0	1623	3388	0.0	891
30	2	3.0	7200	42.1	775	7200	16.4	1598	7200	13.3	1645
30	2	4.5	7200	34.2	1252	7200	16.5	2143	7200	16.6	1947
35	2	1.5	7200	38.9	591	7200	12.0	814	7200	15.9	887
35	2	3.0	7200	59.6	616	7200	27.8	868	7200	25.3	909
35	2	4.5	7200	39.7	254	7200	33.9	925	7200	40.8	793
40	2	1.5	7200	47.4	357	7200	20.7	469	7200	20.5	544
40	2	3.0	7200	56.0	244	7200	33.8	453	7200	33.3	420
40	2	4.5	7200	60.4	284	7200	40.8	578	7200	38.5	489
45	2	1.5	7200	40.4	141	7200	17.0	268	7200	17.1	357
45	2	3.0	7200	60.5	70	7200	36.3	242	7200	37.1	293
45	2	4.5	7200	59.1	162	7200	45.3	316	7200	45.8	293
50	2	1.5	7200	43.6	60	7200	25.3	197	7200	22.9	220
50	2	3.0	7200	60.2	75	7200	39.6	189	7200	42.8	180
50	2	4.5	7200	72.6	55	7200	48.7	165	7200	50.0	166
55	2	1.5	7200	42.8	47	7200	24.9	119	7200	25.7	167
55	2	3.0	7200	63.7	29	7200	43.7	118	7200	45.2	159
55	2	4.5	7200	73.3	28	7200	52.2	134	7200	53.4	157
60	2	1.5	7200	45.2	17	7200	30.7	96	7200	29.4	89
60	2	3.0	7200	66.1	23	7200	45.9	70	7200	47.8	85
60	2	4.5	7200	76.4	23	7200	53.3	75	7200	51.9	108
30	5	1.5	7200	22.1	1319	95	0.0	15	39	0.0	7
30	5	3.0	7200	32.4	1041	193	0.0	35	62	0.0	11
30	5	4.5	7200	27.3	1282	169	0.0	43	250	0.0	76
35	5	1.5	7200	24.6	628	369	0.0	40	237	0.0	28
35	5	3.0	7200	39.8	589	2542	0.0	224	3281	0.0	406
35	5	4.5	7200	39.6	717	7200	4.5	946	7200	13.3	933
40	5	1.5	7200	25.0	573	7200	1.8	290	7200	4.2	594
40	5	3.0	7200	38.4	362	7200	11.2	247	7200	18.1	444
40	5	4.5	7200	43.6	349	7200	13.9	322	7200	22.2	567
45	5	1.5	7200	23.8	338	7200	3.7	159	7200	10.9	413
45	5	3.0	7200	46.8	198	7200	19.5	150	7200	23.0	322
45	5	4.5	7200	51.3	202	7200	23.0	218	7200	26.9	354
50	5	1.5	7200	28.3	193	7200	12.3	105	7200	11.0	254
50	5	3.0	7200	44.2	147	7200	27.9	97	7200	23.2	194
50	5	4.5	7200	56.7	117	7200	34.2	155	7200	31.5	253
55	5	1.5	7200	31.1	69	7200	13.8	63	7200	15.7	138
55	5	3.0	7200	51.7	61	7200	35.1	77	7200	31.7	148
55	5	4.5	7200	58.6	80	7200	38.1	59	7200	37.7	156
60	5	1.5	7200	26.4	66	7200	11.0	35	7200	11.8	91
60	5	3.0	7200	51.5	43	7200	35.8	42	7200	35.4	61
60	5	4.5	7200	56.8	57	7200	36.7	48	7200	34.6	114

Continued on Next Page...

Table 5.1 – Continued

Instances			Default			Constraints			Orbitopal		
m	C	ρ	CPU	gap	node	CPU	gap	node	CPU	gap	node
30	8	1.5	2351	0.0	538	5	0.0	0.3	4	0.0	0.4
30	8	3.0	7200	22.5	983	11	0.0	1	7	0.0	1
30	8	4.5	7200	21.6	971	14	0.0	2	7	0.0	1
35	8	1.5	7200	14.0	701	16	0.0	1	19	0.0	2
35	8	3.0	7200	27.8	665	84	0.0	6	26	0.0	2
35	8	4.5	7200	21.7	732	51	0.0	6	44	0.0	6
40	8	1.5	7200	17.0	395	79	0.0	3	30	0.0	1
40	8	3.0	7200	34.1	402	1010	0.0	42	968	0.0	76
40	8	4.5	7200	27.4	433	75	0.0	2	77	0.0	6
45	8	1.5	7200	17.6	259	384	0.0	12	500	0.0	31
45	8	3.0	7200	34.1	277	5576	0.0	157	5692	0.0	301
45	8	4.5	7200	38.9	291	3685	0.0	123	7200	5.1	304
50	8	1.5	7200	19.3	194	1312	0.0	22	482	0.0	16
50	8	3.0	7200	39.3	156	7200	15.7	88	7200	17.3	203
50	8	4.5	7200	43.8	180	7200	13.8	102	7200	18.2	237
55	8	1.5	7200	21.1	100	7200	4.6	56	7200	6.5	165
55	8	3.0	7200	41.4	100	7200	23.0	61	7200	19.4	157
55	8	4.5	7200	45.8	131	7200	23.5	64	7200	21.2	167
60	8	1.5	7200	25.9	59	7200	16.0	35	7200	12.9	100
60	8	3.0	7200	43.8	58	7200	26.9	42	7200	25.0	104
60	8	4.5	7200	49.0	70	7200	31.2	39	7200	27.9	99

As expected, symmetry-breaking methods have a huge impact on the MILP performances. Notice that when no symmetry breaking method is applied, CPLEX could not solve almost any of tested instances to optimality (the exception is the instance with $m = 30$, $C = 8$ and $\rho = 1.5$). On the other hand, when symmetry methods were applied, some of the non-solved instances (notably instances with $C = 8$) could be solved to optimality. More specifically, the number of solved instances increases from 1 to 19 out of the 63 tested instances. In addition, when optimality could not be proved within time limit using symmetry-breaking methods, the remaining gap obtained for such instances were greatly smaller than the ones obtained with default CPLEX. Such difference of performances is also reflected by the disparity of the enumeration tree sizes. Indeed, for the instances solved to optimality, the number of nodes explored in the enumeration tree is much smaller when symmetry-breaking methods are applied. This shows that such methods allow a much faster progression in the search for optimality.

Given such contrast of performances, the choice of applying such symmetry-breaking methods is therefore straightforward. Nonetheless, the choice of which method to employ is much less trivial. In fact, out the 19 instances solved to optimality, 12 instances were solved faster under scenario `Orbitopal`. Moreover, in all such instances, the number of nodes needed to be explored in the enumeration tree to prove optimality was also smaller under scenario `Orbitopal`. This is explained by the fact that Orbitopal Fixing is capable of fully removing the symmetry from partition problems while the symmetry constraints introduced to the formulation

only perform this job partially. For this reason, Orbitopal Fixing will be the elected method for breaking symmetry in future tests.

Nonetheless, one of the instances (*cf.*, $m = 45$, $C = 8$, $\rho = 4.5$) solved under scenario `Constraints` could not be solved to optimality using Orbitopal Fixing method. Conversely, also one of the tested instances (*cf.*, $m = 30$, $C = 2$, $\rho = 1.5$) solved under scenario `Orbitopal` could not be solved to optimality using symmetry-breaking constraints. Moreover, when instances could not be solved to optimality, the performances of scenario `Constraints` are slightly better when compared to `Orbitopal`. Indeed, by the end of the time limit 24 out of the 44 instances were left with a smaller remaining optimality gap under scenario `Constraints`.

5.2 Eliminating variables

In Section 3.1, it was shown that unlike stated in Pimenta et al. [151], the number of vehicles used in an optimal solution of U-SNP may differ from the minimum number of vehicles needed to obtain a feasible solution, that is, $p_{opt} \neq p_{min}$. For this reason and in the absence of a better bound on p_{opt}, the number of available vehicles p was, up to now, taken to be m. Notice however, that the number of variables in the studied formulation is directly related to the number of available vehicles. More precisely, the formulation uses $(m+n)p$ variables. Therefore, providing a better upper bound on p_{opt} would allow to considerably reduce the number of variables in the formulation. Moreover, recall from the previous section that the symmetry inherent to U-SNP is also closely related to the number of available vehicles. Indeed, for each feasible solution of a given instance of U-SNP, one may come up with $p!$ symmetric solutions. Reducing the value of p would hence reduce the MILP symmetry. The next theorem shows how, without loss of optimality, one may decrease the number of available vehicles for any given instance of U-SNP.

Theorem 5.1 - Upper bound on p_{opt}
For any instance $\mathcal{I} = (V, E, C)$ of U-SNP,

$$p_{opt} \leq \left\lceil \frac{m}{\left\lfloor \frac{C}{2} \right\rfloor + 1} \right\rceil .$$

Proof. Suppose there exists an instance $\mathcal{I}$ of U-SNP for which

$$p_{opt} = \left\lceil \frac{m}{\left\lfloor \frac{C}{2} \right\rfloor + 1} \right\rceil + 1, \tag{5.3}$$

and let $\{E_1^*, \ldots, E_{p_{opt}}^*, \ldots, E_m^*\}$ be an optimal solution for $\mathcal{I}$ using p_{opt} vehicles.

Without loss of generality, assume that $E_i \neq \emptyset$ for $1 \leq i \leq p_{opt}$ and $E_i = \emptyset$ for $p_{opt}+1 \leq i \leq m$. The proof of such upper bound on p_{opt} relies on the fact that there can be no two non-empty vehicles taking, together, less than $C+1$ demands. Notice that if this is not the case, one can just merge such vehicles and find a solution of same cost using one less vehicle.

Therefore, at most one vehicle is allowed to take strictly less than $\left\lfloor \frac{C}{2} \right\rfloor + 1$ demands. Assume such vehicle takes $1 \leq d \leq \left\lfloor \frac{C}{2} \right\rfloor$ demands. Every other non-empty vehicle must take at least $\left\lfloor \frac{C}{2} \right\rfloor + 1$ demands. It follows that the inequality

$$m \geq (p_{opt} - 1)\left(\left\lfloor \frac{C}{2} \right\rfloor + 1\right) + d, \tag{5.4}$$

must hold. Using equality (5.3) to replace p_{opt} in (5.4), one obtains

$$m \geq \left(\left\lceil \frac{m}{\left\lfloor \frac{C}{2} \right\rfloor + 1} \right\rceil\right)\left(\left\lfloor \frac{C}{2} \right\rfloor + 1\right) + d, \tag{5.5}$$

Notice that given two positive integers a and b,

$$a = \left\lceil \frac{a}{b} \right\rceil b - r, \tag{5.6}$$

where $r = b - (a \bmod b)$ is, by definition, a non-negative integer number. Therefore, replacing a by m and b by $\left\lfloor \frac{C}{2} \right\rfloor + 1$ on equation (5.6), one obtains

$$m = \left\lceil \frac{m}{\left\lfloor \frac{C}{2} \right\rfloor + 1} \right\rceil \left(\left\lfloor \frac{C}{2} \right\rfloor + 1\right) - r, \tag{5.7}$$

where $r = m \bmod \left(\left\lfloor \frac{C}{2} \right\rfloor + 1\right)$. By comparing equation (5.7) with inequality (5.5), one finally gets

$$-r \geq d, \tag{5.8}$$

a contradiction, since d is positive and r is non-negative. Therefore,

$$p_{opt} \leq \left\lceil \frac{m}{\left\lfloor \frac{C}{2} \right\rfloor + 1} \right\rceil. \qquad \blacksquare$$

Moreover, the proposed upper bound is tight as there exists examples of instances of U-SNP where such bound is achieved for any value of C.

Proposition 5.1 - Tightness of p_{opt} upper bound
The upper bound on p_{opt} provided by Theorem 5.1 is tight for any capacity C.

Proof. Consider an instance of Intersection U-SNP composed of C sets of $\left\lfloor \frac{C}{2} \right\rfloor + 1$ parallel demands. An example for $C = 5$ is depicted in Figure 5.2.

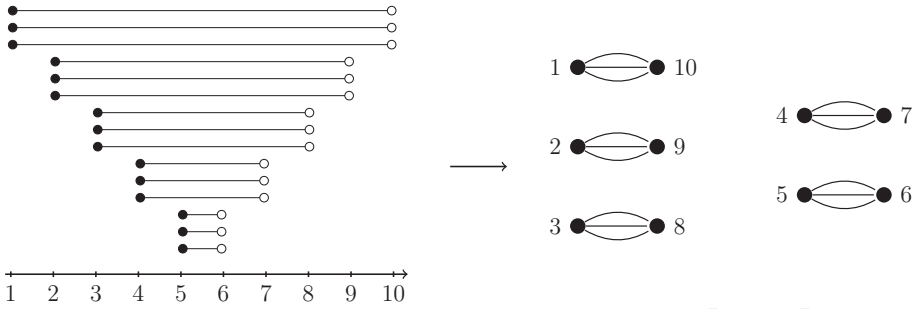


Figure 5.2: Example of instance where $p_{opt} = \left\lceil \frac{m}{\left\lfloor \frac{C}{2} \right\rfloor + 1} \right\rceil$.

In such instance, an optimal solution with n stops can only be obtained if each station is visited by exactly one vehicle. Since each set of parallel demands is composed of $\left\lfloor \frac{C}{2} \right\rfloor + 1$ demands, each set must be in a different vehicle. Therefore, $p_{opt} = \left\lceil \frac{m}{\left\lfloor \frac{C}{2} \right\rfloor + 1} \right\rceil$. ■

In order to understand the impact of setting the number of available vehicles p to the new bound provided by Theorem 5.1 has on the optimization process, Table 5.2 exposes the performances obtained with CPLEX using a dummy callback and default settings. The results displayed under section `Default` account for the case where p is set to m. When p is set to the new upper bound $\left\lceil \frac{m}{\left\lfloor \frac{C}{2} \right\rfloor + 1} \right\rceil$ the results are provided under section `Improved Bound`. A third section `Minimum` is also put to analysis displaying the obtained results for p set to p_{min}.

For each scenario, the number of available vehicles is given under column `p`, and the number of variables in the formulation under column `cols`. Column `best` reports the cost – the total amount of stops – of the best solution found during the optimization process. The number of non-empty vehicles used in such solution is given under column p'. The total time in seconds required for the optimization is reported under column `CPU`. If the time limit of two hours is exceeded, the remaining gap percentage is displayed under column `gap`. The number of nodes (in thousands) explored in the enumeration tree is given under column `node`.

Table 5.2 provides the obtained results for instances with the number of demands m ranging from 30 to 60, the capacity C ranging from 2 to 8 and the graph density ρ from 1.5 to 4.5. For each combination of parameters m, C and ρ, only instances $m\_C\_\rho\_1$ are analysed. For a more exhaustive comparison, the reader is referred to Table A.3 in Appendix A.

As expected, reducing the number of available vehicles has an important impact on the MILP performance. Notice that when the trivial upper bound on p_{opt} (*i.e.*,

Table 5.2: Impact of the new upper bound on p_{opt}

Instances			Default							Improved Bound							Minimum						
m	C	ρ	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
30	2	1.5	30	1500	11	35	7200	35.3	808.0	15	750	10	35	7200	22.7	2233.3	10	500	10	35	7200	16.5	3129.3
30	2	3.0	30	1200	11	31	7200	42.1	775.0	15	600	10	31	7200	35.6	2571.2	10	400	10	31	7200	28.0	3912.1
30	2	4.5	30	1080	7	24	7200	34.2	1251.6	15	540	7	24	7200	25.2	3145.8	7	252	7	24	181	0.0	221.7
35	2	1.5	35	2030	11	41	7200	38.9	591.1	18	1044	11	41	7200	27.9	1253.2	10	580	10	41	7200	19.4	2127.8
35	2	3.0	35	1610	12	35	7200	59.6	616.4	18	828	12	35	7200	51.0	1506.4	12	552	12	35	7200	41.3	2731.2
35	2	4.5	35	1470	12	32	7200	39.7	253.8	18	756	12	32	7200	49.1	1808.3	12	504	12	32	7200	35.7	3653.6
40	2	1.5	40	2640	14	51	7200	47.4	356.9	20	1320	13	50	7200	40.4	854.4	11	726	11	50	7200	29.3	1823.4
40	2	3.0	40	2120	11	38	7200	56.0	244.0	20	1060	11	38	7200	52.8	1087.7	11	583	11	38	7200	38.0	1647.1
40	2	4.5	40	1920	11	35	7200	60.4	284.2	20	960	11	35	7200	49.7	1354.1	11	528	11	35	7200	33.5	2552.8
45	2	1.5	45	3375	12	50	7200	40.4	141.1	23	1725	11	49	7200	35.1	621.4	11	825	11	49	7200	25.2	1574.7
45	2	3.0	45	2700	13	44	7200	60.5	69.8	23	1380	13	45	7200	56.9	617.4	13	780	13	45	7200	46.5	1330.9
45	2	4.5	45	2475	14	42	7200	59.1	162.2	23	1265	14	42	7200	55.7	771.0	14	770	14	42	7200	44.9	1423.3
50	2	1.5	50	4150	17	63	7200	43.6	59.5	25	2075	17	63	7200	39.0	430.4	16	1328	16	62	7200	29.3	683.8
50	2	3.0	50	3300	15	50	7200	60.2	75.1	25	1650	15	50	7200	55.2	402.1	15	990	15	50	7200	46.8	899.0
50	2	4.5	50	3050	14	47	7200	72.6	54.5	25	1525	15	47	7200	70.2	418.4	14	854	14	47	7200	57.1	963.4
55	2	1.5	55	5005	17	65	7200	42.8	47.4	28	2548	16	64	7200	39.3	228.7	14	1274	14	64	7200	34.2	805.8
55	2	3.0	55	4015	16	56	7200	63.7	29.2	28	2044	16	57	7200	57.7	214.0	16	1168	16	57	7200	54.8	517.8
55	2	4.5	55	3685	15	49	7200	73.3	27.7	28	1876	15	49	7200	68.5	276.1	15	1005	15	49	7200	63.0	886.3
60	2	1.5	60	6000	19	74	7200	45.2	17.2	30	3000	19	73	7200	42.7	162.7	19	1900	19	74	7200	38.2	436.3
60	2	3.0	60	4800	19	65	7200	66.1	22.5	30	2400	20	64	7200	62.1	168.9	18	1440	18	64	7200	61.7	282.8
60	2	4.5	60	4380	19	59	7200	76.4	23.0	30	2190	18	58	7200	71.6	177.5	16	1168	16	57	7200	68.1	434.4
30	5	1.5	30	1500	4	25	7200	22.1	1318.6	10	500	4	25	1593	0.0	1253.6	4	200	4	25	5	0.0	9.8
30	5	3.0	30	1200	4	19	7200	32.4	1040.6	10	400	5	19	4829	0.0	3825.6	4	160	4	19	8	0.0	17.3
30	5	4.5	30	1080	4	14	7200	27.3	1282.2	10	360	4	14	131	0.0	140.3	4	144	4	14	1	0.0	1.2
35	5	1.5	35	2030	5	31	7200	24.6	627.6	12	696	5	31	7200	11.4	2601.6	5	290	5	31	112	0.0	135.7
35	5	3.0	35	1610	4	22	7200	39.8	588.6	12	552	4	22	7200	21.6	2429.9	4	184	4	22	80	0.0	131.5
35	5	4.5	35	1470	6	18	7200	39.6	717.0	12	504	6	18	7200	19.2	2411.9	6	252	6	18	837	0.0	1096.0
40	5	1.5	40	2640	5	37	7200	25.0	572.7	14	924	5	37	7200	14.6	2056.6	4	264	4	37	105	0.0	126.1
40	5	3.0	40	2120	5	26	7200	38.4	361.8	14	742	5	26	7200	25.9	1389.8	5	265	5	26	5091	0.0	5092.7
40	5	4.5	40	1920	5	19	7200	43.6	348.9	14	672	5	19	7200	26.4	1409.6	5	240	5	19	342	0.0	369.9
45	5	1.5	45	3375	5	40	7200	23.8	337.9	15	1125	5	40	7200	15.0	1405.2	4	300	4	41	162	0.0	159.2
45	5	3.0	45	2700	6	31	7200	46.8	198.1	15	900	6	31	7200	34.1	819.2	6	360	6	30	7200	13.4	3070.4
45	5	4.5	45	2475	6	25	7200	51.3	201.9	15	825	6	25	7200	40.4	1253.8	6	330	6	25	7200	16.4	3669.9
50	5	1.5	50	4150	7	46	7200	28.3	192.6	17	1411	7	46	7200	20.8	994.4	6	498	6	46	7200	7.4	3294.2
50	5	3.0	50	3300	6	31	7200	44.2	146.6	17	1122	6	31	7200	34.6	706.9	6	396	6	31	7200	13.9	2670.7
50	5	4.5	50	3050	7	29	7200	56.7	117.0	17	1037	7	29	7200	45.0	774.3	7	427	7	29	7200	27.6	2327.4
55	5	1.5	55	5005	8	52	7200	31.1	69.2	19	1729	7	51	7200	23.3	614.2	7	637	7	51	7200	13.8	1933.8
55	5	3.0	55	4015	7	39	7200	51.7	61.2	19	1387	7	39	7200	43.5	449.4	7	511	7	39	7200	28.5	1650.9
55	5	4.5	55	3685	7	31	7200	58.6	79.8	19	1273	6	30	7200	47.5	617.5	6	402	6	30	7200	26.5	2353.6

Continued on Next Page...

Table 5.2 – Continued

Instances			Default							Improved Bound							Minimum						
m	C	ρ	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
60	5	1.5	60	6000	6	50	7200	26.4	66.3	20	2000	6	50	7200	21.1	370.2	5	500	5	50	7200	7.1	2715.8
60	5	3.0	60	4800	7	43	7200	51.5	43.2	20	1600	7	42	7200	44.0	326.9	7	560	7	42	7200	28.6	1261.6
60	5	4.5	60	4380	6	32	7200	56.8	57.3	20	1460	6	31	7200	45.6	437.7	6	438	6	31	7200	24.5	1880.9
30	8	1.5	30	1500	3	24	2351.4	0.0	538.0	6	300	3	24	1	0.0	0.8	3	150	3	24	0.3	0.0	0.1
30	8	3.0	30	1200	3	16	7200	22.5	982.9	6	240	3	16	8	0.0	8.6	3	120	3	16	1	0.0	0.9
30	8	4.5	30	1080	3	11	7200	21.6	970.5	6	216	3	11	3	0.0	3.7	3	108	3	11	0.3	0.0	0.2
35	8	1.5	35	2030	3	29	7200	14.0	701.3	7	406	3	29	16	0.0	13.7	3	174	3	29	1	0.0	1.0
35	8	3.0	35	1610	3	18	7200	27.8	665.1	7	322	3	18	71	0.0	65.6	3	138	3	18	1	0.0	1.3
35	8	4.5	35	1470	3	12	7200	21.7	732.0	7	294	3	12	7	0.0	6.8	3	126	3	12	1	0.0	0.1
40	8	1.5	40	2640	3	32	7200	17.0	394.5	8	528	3	32	89	0.0	48.6	3	198	3	32	1	0.0	1.2
40	8	3.0	40	2120	3	22	7200	34.1	402.4	8	424	3	22	2428	0.0	1477.3	3	159	3	22	3	0.0	7.8
40	8	4.5	40	1920	3	14	7200	27.4	432.8	8	384	3	14	61	0.0	29.3	3	144	3	14	1	0.0	0.6
45	8	1.5	45	3375	3	38	7200	17.6	259.2	9	675	3	38	1327	0.0	511.3	3	225	3	38	2	0.0	3.8
45	8	3.0	45	2700	4	25	7200	34.1	276.9	9	540	4	25	7200	11.0	2590.8	4	240	4	25	54	0.0	66.7
45	8	4.5	45	2475	4	19	7200	38.9	290.6	9	495	4	19	4574	0.0	1789.4	4	220	4	19	13	0.0	18.1
50	8	1.5	50	4150	4	41	7200	19.3	193.7	10	830	4	41	7200	5.1	1691.2	4	332	4	41	19	0.0	16.7
50	8	3.0	50	3300	4	28	7200	39.3	155.6	10	660	4	28	7200	19.9	1553.1	4	264	4	28	501	0.0	667.1
50	8	4.5	50	3050	4	22	7200	43.8	179.5	10	610	4	22	7200	20.6	1554.3	4	244	4	22	212	0.0	215.9
55	8	1.5	55	5005	5	44	7200	21.1	99.9	11	1001	4	44	7200	9.1	1261.1	4	364	4	44	132	0.0	128.3
55	8	3.0	55	4015	5	32	7200	41.4	99.7	11	803	5	32	7200	25.8	1092.3	5	365	5	31	7200	5.2	4197.1
55	8	4.5	55	3685	4	24	7200	45.8	130.5	11	737	4	24	7200	26.6	1171.6	4	268	4	24	251	0.0	260.4
60	8	1.5	60	6000	5	51	7200	25.9	58.5	12	1200	5	51	7200	16.3	829.0	5	500	5	51	7200	5.2	3430.0
60	8	3.0	60	4800	5	37	7200	43.8	57.9	12	960	5	37	7200	30.3	660.5	5	400	5	36	7200	13.1	3275.2
60	8	4.5	60	4380	5	27	7200	49.0	69.8	12	876	5	28	7200	34.9	831.4	5	365	5	27	7200	11.8	3200.9

$p = m$) is applied, CPLEX could only solve one of tested instances to optimality (*cf.* the instance with $m = 30$, $C = 8$ and $\rho = 1.5$). On the other hand, when p is set to the aforementioned new upper bound on p_{opt} (*i.e.*, $p = \left\lceil \frac{m}{\lfloor \frac{C}{2} \rfloor + 1} \right\rceil$), the number of solved instances is raised to 14. In addition, when optimality could not be proved within time limit, the remaining gap obtained was greatly smaller using the new upper bound on p_{opt}. Indeed, the average gap obtained with scenario `Default` was 40.4% and by applying the improved bound this percentage goes down to 27.6%. Such disparity of performances is due to the difference in the size of the LPs. In fact, setting the number of available vehicles to the upper bound provided by Theorem 5.1, reduces the number of variables in the formulation by 60% in average, for the tested instances. Furthermore, in some specific instances (with high capacities and high densities), a reduction of up to 80% of the variables could be achieved. This means that each LP node in the enumeration tree of scenario `Improved Bound` can be solved faster than the correspondent LP node in scenario `Default`, and hence the number of explored nodes within the same amount of time is much greater once the improved bound is applied. Notice that for this same reason, the scenario `Minimum` is capable of obtaining better results than the two previous ones. Nonetheless, this third scenario cannot be seen as a method for solving U-SNP since p_{min} might differ from p_{opt}. Instead, it only provides an upper bound on the optimal solution value.

We insist in highlighting this scenario for a practical analysis of how often one can achieve optimality with only p_{min} vehicles available. In fact, in only 1 out of the 63 tested instances (*cf.* the instance with $m = 45$, $C = 5$ and $\rho = 1.5$), strictly more than p_{min} vehicles were necessary to obtain, within time limit, a better solution than the upper bound provided by scenario `Minimum`. Such behaviour is not a surprise since in Pimenta et al. [151], p_{opt} was empirically claimed to be equivalent to p_{min} based on tests developed with randomly generated instances. In fact, we believe that requiring more vehicles than the strict minimum needed – p_{min} – in order to achieve an optimal solution relies on particular structures of the associated graph G, that may not arise quite often when the instances are randomly generated.

5.3 Relaxing variables

In Section 4.2, the linear relaxation of formulation USNP$_{(V,E,C)}$ is shown to be particularly weak. Such linear relaxation is obtained by replacing the domain of variables $x_e^i \in \{0,1\}$ and $y_v^i \in \{0,1\}$ by $0 \leq x_e^i \leq 1$ and $0 \leq y_v^i \leq 1$ for $e \in E$, $v \in V$, $i \in K$. At the same time, the formulation obtained from the relaxation of variables y_v^i is shown to optimally solve U-SNP. Such formulation is hereafter denoted USNP-X$_{(V,E,C)}$ as only variables x_e^i are constrained to be integers. On the

other hand, keeping the integrality of variables y_v^i and relaxing variables x_e^i does not guarantee integrality as stated in Section 4.2.2. For this reason, let us consider USNP-Y$_{(V,E,C)}$ to be the formulation obtained from USNP$_{(V,E,C)}$ while setting a priority of branching on variables y_v^i. In other words, a variable x_e^i can only be chosen as the branching variable if all variables y_v^i already satisfy the integrality constraints.

Recall that the strength of a formulation is measured by its integrality gap. Since the three aforementioned formulations – USNP$_{(V,E,C)}$, USNP-Y$_{(V,E,C)}$ and USNP-X$_{(V,E,C)}$ – have the same linear relaxation, all three have the same strength. Nonetheless, their performances may differ. Indeed, the choice of which variable to branch on might influence the performance of the Branch-and-Bound procedure used for solving the MILP. With respect to the U-SNP, the information that can be derived from the fixing of a variable y_v^i is much different from that of a variable x_e^i. Fixing a variable y_v^i to 0 implies that $x_e^i = 0$, for all $e \in \delta(v)$. On the other hand, fixing a variable x_e^i to 1 also fixes $y_{o_e}^i$ and $y_{d_e}^i$ to 1, indirectly. Moreover, the denser the associated graph $G = (V, E)$ is, the bigger is the difference between the number of variables x and y. Therefore, it is expected that USNP-Y$_{(V,E,C)}$ should achieve better results when G is dense.

In order to evaluate how these formulations perform in comparison with the original one, Table 5.3 reports the results obtained with each one of the formulations. For each formulation, the number of binary variables is given under column `bin`. The total time in seconds required for the optimization is displayed under column `CPU`. If the time limit of two hours is exceeded, the remaining gap percentage is displayed under column `gap`. The number of nodes (in thousands) explored in the enumeration tree is given under column `node`. In all tested scenarios, the Orbitopal Fixing symmetry-breaking method is applied and the number of available vehicles is set to the upper bound on p_{opt} given by Theorem 5.1.

Table 5.3 provides information for instances with the number of demands m ranging from 30 to 60, the capacity C ranging from 2 to 8 and the graph density ρ from 1.5 to 4.5. For each combination of parameters m, C and ρ, only instances $m\_C\_\rho\_1$ are displayed. For more exhaustive results, the reader is referred to Table A.4 in Appendix A.

Table 5.3: Comparison between relaxations

Instances			USNP$_{(V,E,C)}$				USNP-X$_{(V,E,C)}$				USNP-Y$_{(V,E,C)}$			
m	C	ρ	bin	CPU	gap	node	bin	CPU	gap	node	bin	CPU	gap	node
30	2	1.5	750	6021.7	0	2930.2	450	4696.2	0	2367.2	750	7200	7.3	2127.6
30	5	1.5	500	14.1	0	9.8	300	16.5	0	8.3	500	5	0	3.1
30	8	1.5	300	0.4	0	0.1	180	0.4	0	0.1	300	0.4	0	0.1
35	2	1.5	1044	7200	17.5	1212.6	630	7200	13.3	1526.5	1044	7200	18.3	1097.3

Continued on Next Page...

Table 5.3 – Continued

Instances			USNP$_{(V,E,C)}$				USNP-X$_{(V,E,C)}$				USNP-Y$_{(V,E,C)}$			
m	C	ρ	bin	CPU	gap	node	bin	CPU	gap	node	bin	CPU	gap	node
35	5	1.5	696	522	0	227.1	420	111	0	39.6	696	755.6	0	295.6
35	8	1.5	406	2.1	0	0.9	245	3.3	0	1.7	406	2.5	0	1.5
40	2	1.5	1320	7200	21.2	905.2	800	7200	22.1	949	1320	7200	25.0	748.9
40	5	1.5	924	1895.6	0	546.1	560	2182.5	0	501.3	924	7200	4.8	1637.2
40	8	1.5	528	7.3	0	2.8	320	7.4	0	3.1	528	2.9	0	0.9
45	2	1.5	1725	7200	18	563.3	1035	7200	14.2	748.1	1725	7200	22.3	517.3
45	5	1.5	1125	7200	5.7	1143.9	675	7200	7.1	1205.4	1125	7200	8.4	1062.7
45	8	1.5	675	19.6	0	4.8	405	45	0	10.9	675	64.3	0	19.9
50	2	1.5	2075	7200	20.6	389.1	1250	7200	20.9	418.6	2075	7200	24.7	346.9
50	5	1.5	1411	7200	9.5	858	850	7200	11	780.9	1411	7200	11.8	826.8
50	8	1.5	830	210.3	0	45.6	500	189.9	0	35.6	830	109.4	0	19.6
55	2	1.5	2548	7200	22.5	242	1540	7200	23.8	329.2	2548	7200	26.3	313.8
55	5	1.5	1729	7200	15.3	532.9	1045	7200	12.7	507.1	1729	7200	12.9	416.4
55	8	1.5	1001	1407.3	0	318.2	605	4161.8	0	630.2	1001	5378.7	0	1112.7
60	2	1.5	3000	7200	28.9	163.9	1800	7200	27.5	176.7	3000	7200	30.4	178.9
60	5	1.5	2000	7200	10.6	435.4	1200	7200	9.8	431	2000	7200	9.7	346
60	8	1.5	1200	7200	10.3	740.5	720	7200	11.1	684.4	1200	7200	9.8	665.3
30	2	3	600	7200	15.7	2560.6	450	6259.2	0	3726.9	600	7200	17.4	2140.2
30	5	3	400	92.2	0	60.5	300	39.6	0	20.7	400	167	0	107.3
30	8	3	240	1.4	0	1.1	180	2.3	0	1.7	240	0.7	0	0.4
35	2	3	828	7200	29.1	1406	630	7200	27.5	1401.8	828	7200	29.7	1118.1
35	5	3	552	1701.5	0	882.2	420	906.5	0	281.9	552	1640.4	0	704.3
35	8	3	322	2.2	0	0.9	245	7.9	0	4.1	322	1	0	0.4
40	2	3	1060	7200	33.9	817.8	800	7200	34.3	846.5	1060	7200	34.7	701.6
40	5	3	742	7200	14.7	1362.7	560	7200	10.8	1419.2	742	7200	11.8	1243.9
40	8	3	424	101.7	0	50.8	320	71.7	0	30.9	424	108.2	0	58.9
45	2	3	1380	7200	34.6	478.7	1035	7200	34.3	527.4	1380	7200	36.5	472.7
45	5	3	900	7200	22.3	992.7	675	7200	21.3	899.7	900	7200	23.3	853.8
45	8	3	540	564.5	0	204.5	405	509.3	0	145.9	540	196.1	0	65
50	2	3	1650	7200	38.9	413.7	1250	7200	41.6	363.9	1650	7200	40.9	367.2
50	5	3	1122	7200	23.7	705.1	850	7200	22.8	716.3	1122	7200	20.4	464.7
50	8	3	660	7200.1	16.2	1318.2	500	7200	8.5	1347.7	660	7200	8.5	1341.5
55	2	3	2044	7200	35.7	205.8	1540	7200	42.7	296.8	2044	7200	40.7	177.2
55	5	3	1387	7200	30.7	482	1045	7200	33.4	461.5	1387	7200	29.9	421.6
55	8	3	803	7200	20.5	991.1	605	7200	17.5	957.7	803	7200	17.7	865.9
60	2	3	2400	7200	43.3	142.5	1800	7200	46.6	163	2400	7200	46.1	136.7
60	5	3	1600	7200	29.5	336	1200	7200	34.1	381	1600	7200	33.6	340.3
60	8	3	960	7200	21.6	904.5	720	7200	20.6	775.2	960	7200	18.5	703.9
30	2	4.5	540	4316.2	0	2747.3	450	7200	13.5	3805	540	3479.2	0	2157.9
30	5	4.5	360	1.8	0	0.6	300	68.9	0	48.9	360	1.8	0	0.9
30	8	4.5	216	0.8	0	0.2	180	1.4	0	1.1	216	0.6	0	0.3
35	2	4.5	756	7200	36.7	1340	630	7200	39.2	1362.5	756	7200	32.3	1257.7
35	5	4.5	504	7200	8.2	3069.5	420	4329.6	0	2011.6	504	186	0	67.5
35	8	4.5	294	1.3	0	0.5	245	4	0	2.5	294	0.5	0	0.2
40	2	4.5	960	7200	31.3	1062.3	800	7200	39.7	1022	960	7200	33.9	1020.3
40	5	4.5	672	7200	16	1310.3	560	7200	18.4	1388.2	672	2130.2	0	498.6
40	8	4.5	384	2.8	0	0.9	320	14.5	0	6.8	384	1.8	0	0.5
45	2	4.5	1265	7200	37.3	627.4	1035	7200	41.8	591.1	1265	7200	35.6	456.3
45	5	4.5	825	7200	26.2	1056.3	675	7200	25	1082.2	825	7200	19.5	809
45	8	4.5	495	1025.5	0	365.5	405	919.6	0	300.4	495	220.6	0	77.8
50	2	4.5	1525	7200	46.9	330.6	1250	7200	48.9	368.6	1525	7200	48.3	278.5
50	5	4.5	1037	7200	27.9	755.3	850	7200	30.2	778.1	1037	7200	24.9	603.6
50	8	4.5	610	7200	13.7	1505.7	500	7200	14.6	1328	610	4471.8	0	1311.8
55	2	4.5	1876	7200	53.3	229.1	1540	7200	49.5	343.7	1876	7200	52.4	234.6
55	5	4.5	1273	7200	34.1	493.6	1045	7200	34.1	473.4	1273	7200	30.3	431.1
55	8	4.5	737	7200	17.9	992.7	605	7200	19.9	1150.5	737	7200	12.7	966
60	2	4.5	2190	7200	54.2	168.6	1800	7200	50.7	205.9	2190	7200	54.3	185.3
60	5	4.5	1460	7200	37.8	478.2	1200	7200	32.9	408.1	1460	7200	32.9	391.5
60	8	4.5	876	7200	23.7	895.6	720	7200	30.7	926.1	876	7200	16.1	744.1

Table 5.3 reveals that the difference of performances between the original formulation USNP$_{(V,E,C)}$ and the other two proposed ones is far from being uniform.

Formulation USNP$_{(V,E,C)}$ could solve 22 out of the 63 instances to optimality within time limit, while 23 instances could be solved through formulations USNP-X$_{(V,E,C)}$ and USNP-Y$_{(V,E,C)}$. Notice however, that some of the instances (*e.g.*, $m = 40$, $C = 5$, $\rho = 4.5$) could be solved to optimality with USNP-Y$_{(V,E,C)}$ but not with USNP$_{(V,E,C)}$ and USNP-X$_{(V,E,C)}$ (and vice-versa).

As expected, formulation USNP-Y$_{(V,E,C)}$ clearly outperforms its other two rivals – USNP$_{(V,E,C)}$ and USNP-X$_{(V,E,C)}$ – for instances with dense graphs (that is, $\rho = 4.5$). Considering only instances with $\rho = 4.5$, formulation USNP-Y$_{(V,E,C)}$ was generally either capable of solving the instance to optimality faster or the remaining gap obtained was smaller when compared to the two other formulations. On the other hand, for sparse instances (*i.e.*, $\rho = 1.5$), formulation USNP-Y$_{(V,E,C)}$ could not beat the performances of USNP-X$_{(V,E,C)}$ and USNP$_{(V,E,C)}$.

Even if the results are quite heterogeneous according to the density of the graph, the results obtained from formulation USNP-Y$_{(V,E,C)}$ are, in average, slightly better. In fact, the average remaining gap obtained by the end of the optimization process for USNP-Y$_{(V,E,C)}$ was of 16.1%, while for USNP-X$_{(V,E,C)}$ and USNP$_{(V,E,C)}$ this number raises to 16.8%.

It is worth noting that even a small reduction on the gap may represent a huge computational effort. Figure 5.3 depicts the gap progression through the optimization process for the instance with $m = 60$, $C = 8$, and $\rho = 3$. At the beginning of the optimization, the bounds are constantly updated and the gap reduces quickly. At a certain point – which arises relatively early in the optimization process – the bounds progression hit a barrier and the gap evolution considerably slows down. The final gap obtained with formulation USNP$_{(V,E,C)}$ is 21.6%, while with formulation USNP-Y$_{(V,E,C)}$ a final gap of 18.5% is obtained. Such difference may seem small at first. However, with formulation USNP-Y$_{(V,E,C)}$, the mark of 21.6% of gap is surpassed way before the end of the time limit.

5.4 Branch-and-Cut

A Branch-and-Cut framework consists of embedding a cutting plane algorithm before each branching phase of a Branch-and-Bound algorithm. In this section we show how the valid inequalities presented in Section 4.4 are integrated into our Branch-and-Cut framework used for solving U-SNP. Frequently, the separation problems associated with such valid inequalities require the definition and manipulation of data structures (mostly graphs) as well as the use of state-of-the-art combinatorial algorithms. For such tasks we make use of LEMON, an open source library written

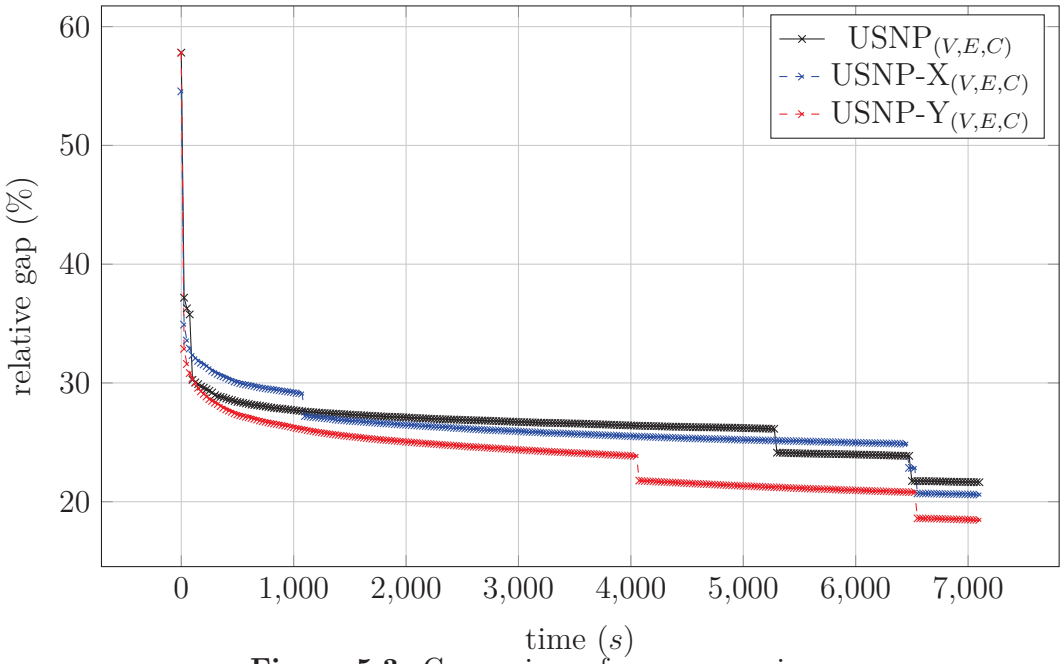


Figure 5.3: Comparison of gap progression

in the C++ language providing efficient implementations of graph and network algorithms (see Dezső et al. [53], Egerváry Combinatorial Optimization Research Group [61]).

5.4.1 Strong capacity inequalities

The strong capacity inequalities (4.34), (4.36) appear in polynomial number. Therefore, stocking such inequalities in a pool and verifying their satisfaction at each node of the enumeration tree remains an efficient way of solving the separation problem. Another reasonable approach would be to directly impose such inequalities to the formulation, that is, introduce all of them in the root node of the enumeration tree. Notice however that if this second approach is chosen, the model image (*i.e.*, its mathematical representation) is changed in the eyes of CPLEX, which might affect – for the better or for the worse – some of the heuristics CPLEX uses for deriving its own cuts and incumbent solutions. For a fairer comparison with the original model, such inequalities are chosen to be added on demand at each node.

5.4.2 k-cardinality tree inequalities

The k-cardinality tree inequalities (4.40), on the other hand, appear in exponential number. Such inequalities are recalled here below:

$$\sum_{e \in S} x_e^i - \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) y_v^i \leq 0 \qquad \forall i \in K, j \in V, S \subseteq \Delta_E(j),$$

such that $G[S]$ is a k-cardinality tree, for $k = C + 1$.

Listing all such inequalities is hence an unreasonable approach. Instead, one is obliged to design more creative ways of adding such inequalities on demand, that is, solving its separation problem. The separation problem associated with k-cardinality tree inequalities (4.40) consists of, given a solution $(\bar{x}, \bar{y})$, decide whether or not there exists an inequality (4.40) that is violated by $(\bar{x}, \bar{y})$. Notice that a k-cardinality tree inequality is defined by a vehicle $i \in K$ and a $(C + 1)$-cardinality tree $G[S]$ such that $S \subseteq \Delta_E(j)$ for some station $j \in V$. Therefore, $(\bar{x}, \bar{y})$ violates a k-cardinality tree inequality if and only if there exists a vehicle $i \in K$, a station $j \in V$, and a $(C + 1)$-cardinality tree $G[S]$ with $S \subseteq \Delta_E(j)$, for which

$$\sum_{e \in S} \bar{x}_e^i - \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1)\bar{y}_v^i > 0.$$

The separation problem associated with k-cardinality tree inequalities is hence equivalent to finding

$$\omega = \max_{i \in K, j \in V} \left\{ \sum_{e \in S} \bar{x}_e^i - \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1)\bar{y}_v^i : S \subseteq \Delta_E(j), G[S] \text{ is a } C + 1 \text{ tree} \right\}, \tag{5.9}$$

and there exists a k-cardinality tree inequality cutting (*i.e.*, violated by) $(\bar{x}, \bar{y})$, if and only if $\omega > 0$. However, before tackling directly the task of designing an algorithm capable of solving this separation problem, let us first analyse the complexity of such problem.

Theorem 5.2 - Separation of k-cardinality tree inequalities
The separation problem associated with k-cardinality tree inequalities (4.40) *is $\mathcal{NP}$-Hard.*

Proof. We give a reduction from the k-Minimum Spanning Tree problem (k-MST) which can be defined as follows. Given a graph $H = (V, E)$ with edge weights w_e and a constant $B \in \mathbb{R}$, find a tree $T = (V', E')$ spanning exactly k edges (or $k + 1$ vertices) with total weight at most B.

When $k = n - 1$, k-MST is nothing but the classic Minimum Spanning Tree problem and therefore can be solved in polynomial time by the famous greedy algorithm proposed by Kruskal [111], Prim [152]. Moreover, if k is a fixed constant, the problem is also polynomially solvable by a brute force enumeration algorithm. In Fischetti et al. [70], k-MST is showed to be polynomially solvable if H is itself a tree. In the general case, however, k-MST is $\mathcal{NP}$-Hard (see Fischetti et al. [70], Ravi et al. [156]). Notice that for proving the $\mathcal{NP}$-Hardness of k-MST, the reduction used in

Fischetti et al. [70] from the Steiner Tree problem preserves bipartiteness. Since Steiner Tree is $\mathcal{NP}$-Hard even for bipartite graphs (see Garey and Johnson [74, p. 208]), k-MST can also be stated as $\mathcal{NP}$-Hard for bipartite graphs.

Next, we show that given a bipartite graph $G = (V, E)$, with negative edge weights w_e, and a constant $B \in \mathbb{R}$, any tree T' spanning k edges in G such that $w(T') \leq B - \epsilon$, corresponds to a violated k-cardinality tree inequality for the following point $(\overline{x}, \overline{y})$: Let $(V, E, k-1)$ be an instance of Intersection U-SNP. Then, given a vehicle $i \in K$, let $\bar{y}^i_v = -\frac{B}{k-1}$ for all $v \in V$ and $\bar{x}^i_e = -w_e$ for all $e \in E$.

Let T denote the minimum cost k-cardinality tree in G. Notice that by definition, $|V(G[T])| = k + 1$. Consequently, the term $\sum_{v \in V(G[T])}(d_{G[T]}(v) - 1)\bar{y}^i_v$ becomes a constant:

$$\begin{aligned}
\sum_{v \in V(G[T])} (d_{G[T]}(v) - 1)\bar{y}^i_v &= \sum_{v \in V(G[T])} d_{G[T]}(i)\bar{y}^i_v - \sum_{v \in V(G[T])} \bar{y}^i_v \\
&= -\sum_{v \in V(G[T])} (d_{G[T]}(v)\frac{B}{k-1}) + \sum_{v \in V(G[T])} \frac{B}{k-1} \\
&= -2k\frac{B}{k-1} + (k+1)\frac{B}{k-1} \\
&= -B
\end{aligned}$$

Therefore, if one is able to find a violated k-cardinality tree inequality (*i.e.*, $\sum_{e \in T} \bar{x}^i_e + B > 0$), then the k-tree defined by T solves the k-MST problem. On the other hand, a k-tree T such that $w(T) \leq B - \epsilon$ defines a violated k-tree inequality. ■

Theorem 5.2 shows that, unless $\mathcal{P} = \mathcal{NP}$, an efficient algorithm capable of solving the separation problem associated with k-cardinality tree inequalities (4.40) is inconceivable. For this reason, one may only hope that a good heuristic scheme will be good enough for generating violated cuts. In this scenario, we propose two heuristics for finding a violated k-cardinality tree inequality.

Minimum Spanning Tree based heuristic

The first heuristic we propose is based on the classic Minimum Spanning Tree problem. For each vehicle $i \in K$ and each station $j \in V$, we consider the graph $G[\Delta_E(j)]$ with edge weights $w_e = -\bar{x}^i_e$ for $e \in \Delta_E(j)$, and node weights $c_v = \bar{y}^i_v$ for $v \in V[\Delta_E(j)]$. Then, a $(C+1)$-cardinality tree is constructed by taking the minimum spanning tree T of $G[\Delta_E(j)]$ obtained with Kruskal's algorithm and pruning its

leaves until it becomes a tree with exactly $C + 1$ edges. Notice that the node weights are not taken into account for constructing the spanning tree T, since it spans all nodes in $G[\Delta_E(j)]$. Instead, they are used for choosing the most suitable leaf to be pruned. Indeed, while T does not have exactly $C + 1$ edges, the leaf edge $e \in E(T)$ with highest impact on (5.9) is chosen to be pruned, that is

$$e \in \arg\max_{e \in E(T)} \{w_e + c_v : e \text{ is a leaf-edge of } T \text{ and } v \text{ is the non-leaf endpoint of } e\}.$$

If the $(C+1)$-cardinality tree obtained by the end of such procedure induces a k-cardinality tree inequality that is violated by the given relaxed solution $(\bar{x}, \bar{y})$, then the inequality is added to the formulation as a cut. Algorithm 4 formally describes such procedure. It takes advantage of function Kruskal – implemented in `LEMON` (see Egerváry Combinatorial Optimization Research Group [63]) – taking as input a graph G and a cost map corresponding to the weights of edges in G, and returning a minimum cost spanning tree. Notice that the complexity of such algorithm is given by the complexity of Kruskal's algorithm (*i.e.*, $\mathcal{O}(E \log V)$) which is executed for each vehicle and each station, that is, $\mathcal{O}(KVE \log V)$.

Without loss of generality, one may only consider stations $j \in V$ for which $|\Delta_E(j)| \geq C+1$, since otherwise no $(C+1)$-cardinality tree can be found. Moreover, if $G[\Delta_E(j)]$ has more than one connected component, then one can independently look for a $(C + 1)$-cardinality tree on each of the connected components.

Greedy k-MST heuristic

Another heuristic for constructing a potentially violated k-cardinality tree inequality is to greedily build a $(C + 1)$-cardinality tree from scratch. For this, given a vehicle $i \in K$ and a station $j \in V$, the graph $G[\Delta_E(j)]$ is considered. Edge weights w_e are set to $\bar{x}^i_e$ for $e \in \Delta_E(j)$, and node weights c_v are set to $-\bar{y}^i_v$ for $v \in V[\Delta_E(j)]$. Next, start with an empty edge set $T = \emptyset$. By the end of the algorithm, T is supposed to define a $(C+1)$-cardinality tree. For this, a single edge is added to T at a turn. The first edge chosen to be part of T is the one with most fractional weight – recall that if $(\bar{x}, \bar{y})$ is integer, then no separation problem is needed. For the next C iterations, the chosen edge $e \in \Delta_E(j) \setminus T$ will be one that is incident to exactly one node of $V[T]$ and maximizes the value of $w_e + c_v$, that is,

$$e = \arg\max_{uv \in \Delta_E(j) \setminus T} \{w_{uv} + c_v : v \in V[T] \text{ and } u \notin V[T]\}.$$

Finally, if the $(C + 1)$-cardinality tree obtained by the end of such procedure

Algorithm 4 Separation of inequalities (4.40) by pruning MST

Input: An instance (V, E, C) of U-SNP, and a fractional solution $(\bar{x}, \bar{y})$ of the linear relaxation of USNP$_{(V,E,C)}$
Output: A "good" candidate k-cardinality tree inequality for cutting $(\bar{x}, \bar{y})$

```
for i = 1, ..., p do
  for j = 1, ..., n do
    for all e ∈ Δ_E(j) do
      w_e = −x̄^i_e
    end for
    for all v ∈ V[Δ_E(j)] do
      c_v = ȳ^i_v
    end for
    T = Kruskal(G[Δ_E(j)], w)
    while |E(T)| > C + 1 do
      prunningEdge = ∅
      maxImprove = −∞
      for all e ∈ E(T) : e is a leaf edge of T do
        nodeCost = c_v : v is the non-leaf endpoint of e
        if (w_e + nodeCost) > maxImprove then
          maxImprove = w_e + nodeCost
          prunningEdge = e
        end if
      end for
      Contract edge prunningEdge in T
    end while
    if Σ_{e∈T} x̄^i_e − Σ_{v∈V[T]} (deg_{G[T]}(v) − 1) ȳ^i_v > 0 then
      Add inequality Σ_{e∈T} x^i_e − Σ_{v∈V[T]} (deg_{G[T]}(v) − 1) y^i_v ≤ 0 as a cut
    end if
  end for
end for
```

induces a k-cardinality tree inequality that is violated by the given relaxed solution $(\bar{x}, \bar{y})$, then the inequality is added to the formulation as a cut. Algorithm 5 formally describes such procedure.

This approach seems less fancy than the aforementioned one taking into account Kruskal's algorithm, but its simplicity is rewarding. Indeed, skipping the construction of a spanning tree saves a lot of time, speeding up the separation process. Moreover, the vehicle's capacity tends to be far smaller than the average number of edges in $G[\Delta_E(j)]$, which makes construction of the $(C+1)$-cardinality tree in Algorithm 5 much faster than in Algorithm 4. Indeed, since the tree T can be constructed in linear time (*i.e.*, $\mathcal{O}(E)$) the complexity of such algorithm is $\mathcal{O}(KVE)$. For these reasons, Algorithm 5 is the chosen heuristic to be applied for trying to solve the separation problem associated with k-cardinality tree inequalities (4.40).

Once again, one may consider only stations $j \in V$ for which $|\Delta_E(j)| \geq C + 1$,

Algorithm 5 Separation of inequalities (4.40) by greedily construction of a $(C+1)$-cardinality tree

Input: An instance (V, E, C) of U-SNP, and a fractional solution $(\bar{x}, \bar{y})$ of the linear relaxation of $\text{USNP}_{(V,E,C)}$
Output: A "good" candidate k-cardinality tree inequality for cutting $(\bar{x}, \bar{y})$

```
for i = 1, ..., p do
  for j = 1, ..., n do
    for all e ∈ Δ_E(j) do
      w_e = x̄^i_e
    end for
    for all v ∈ V[Δ_E(j)] do
      c_v = −ȳ^i_v
    end for
    Choose the edge e with most fractional weight.
    T = {e}
    for k = 1, ..., C do
      Let e_k ∈ arg max_{uv∈Δ_E(j)\T} {w_uv + c_v : v ∈ V[T] and u ∉ V[T]}
      T = T ∪ {e_k}
    end for
    if Σ_{e∈T} x̄^i_e − Σ_{v∈V[T]} (deg_{G[T]}(v) − 1) ȳ^i_v > 0 then
      Add inequality Σ_{e∈T} x^i_e − Σ_{v∈V[T]} (deg_{G[T]}(v) − 1) y^i_v ≤ 0 as a cut
    end if
  end for
end for
```

since otherwise no $(C+1)$-cardinality tree can be found. Moreover, if $G[\Delta_E(j)]$ has more than one connected component, then one can independently look for a $(C+1)$-cardinality tree on each of the connected components.

Impact of k-cardinality tree inequalities

The introduction of k-cardinality tree inequalities into the branch-and-cut framework plays an important role in our constructed Branch-and-Cut framework, notably when dealing with sparse instances. Table 5.4 provides a comparison between the obtained performance results for sparse instances when the separation of such inequalities is active or not. The `inactive` scenario stands for the branch-and-cut framework taking into account the separation of strong capacity inequalities only, while the `active` one also includes the separation of k-cardinality tree inequalities. Columns $\texttt{time}_g$, $\texttt{LB}_g$, $\texttt{UB}_g$, $\texttt{gap}_g$ and `Nodes` provide information on the global optimization process by showing, respectively, the total time spent in seconds, the lower and upper bounds obtained by the end of the optimization, the remaining gap and the number of nodes investigated in the enumeration tree. Columns $\texttt{time}_r$ and $\texttt{LB}_r$ present, respectively, the time spent in seconds for solving the root node

and corresponding lower bound obtained from it. Finally `Cuts` and $\texttt{time}_{cut}$ display, respectively, the number of k-cardinality tree inequalities added to the formulation throughout the optimization process and the time spent on their separation routine.

As expected, in some cases (*e.g.*, instance 60_2_1.5_5) the lower bound obtained from the LP relaxation on the root node is significantly improved by the inclusion of k-tree inequalities. Moreover, it is quite interesting to remark that, in other cases (*e.g.*, instance 30_2_1.5_3), even if the inclusion of such constraints does not have a direct impact on the lower bound value obtained from the LP relaxation, the global performance is still substantially improved when such inequalities are taken into account. Indeed, the addition of k-cardinality tree inequalities greatly reduces the number of nodes in the enumeration tree that have to be verified in order to prove optimality. Figure 5.4 shows the progression of the lower bounds during the optimization process within each scenario for instance 30_2_1.5_3. On the other hand, the activation of the separation routine of k-cardinality tree inequalities greatly increases the time needed for solving the root node. Such behaviour is most perceived within instance 60_2_1.5_3, where no branching could be made within time limit. In other words, solving the root node required more than 2 hours of computational effort. All in all, the inclusion of k-tree inequalities considerably improves the Branch-and-Cut performances as the average final gap is reduced from 9.61% to 6.74%.

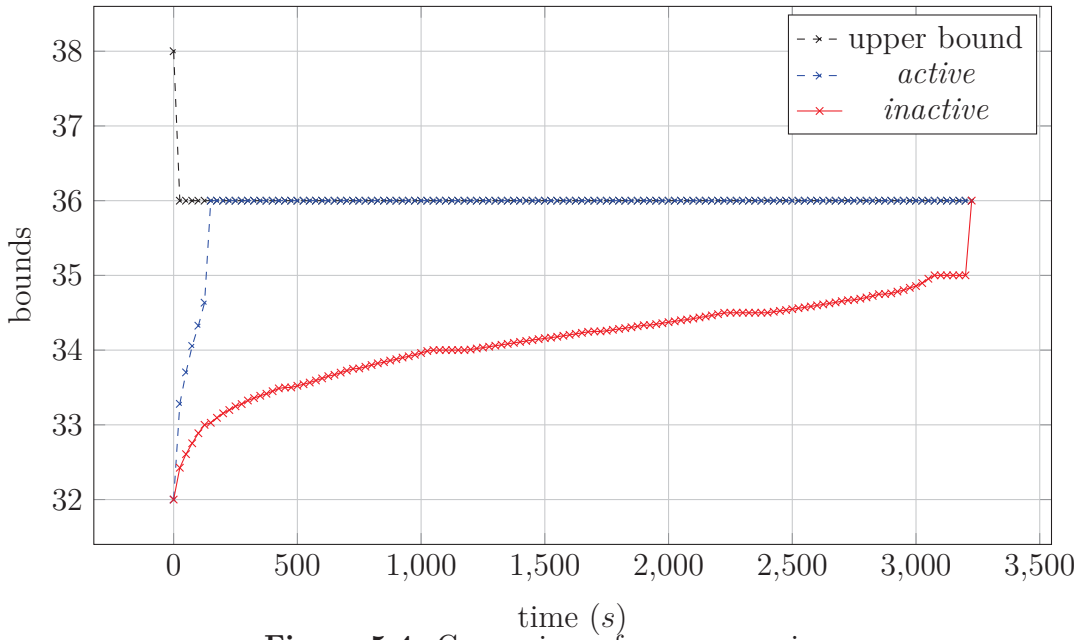


Figure 5.4: Comparison of gap progression

Table 5.4: Impact of the inclusion of k-cardinality tree inequalities

Instances				inactive							active								
m	C	ρ	i	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	time$_r$	LB$_r$	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	Cuts	time$_{cut}$	time$_r$	LB$_r$
30	2	1.5	1	3	35	35	0	0.2	0.9	32	15.1	35	35	0	0.4	420	0.9	9.4	33
30	2	1.5	2	7200	32.7	34	3.92	1917.5	1	32	134.7	34	34	0	24.8	481	1.5	2.3	32
30	2	1.5	3	3213.9	36	36	0	829.7	0.7	32	184.2	36	36	0	24.5	517	1.5	41.6	32
30	2	1.5	4	43	34	34	0	11.9	0.7	30	17.3	34	34	0	0.2	260	0.4	14.2	31
30	2	1.5	5	1163.4	36	36	0	502.3	0.9	30	252.2	36	36	0	82	478	1.4	47.6	30
35	2	1.5	1	7200	36.9	41	9.95	790.5	2.2	34	2068.4	41	41	0	283.5	777	3.3	128.6	35
35	2	1.5	2	7200	39.8	44	9.55	711.3	2.6	39	7200	40.8	44	7.19	695	924	7.8	184.2	39
35	2	1.5	3	1183.7	40	40	0	247.6	1.4	36	88.4	40	40	0	6.4	380	2.3	23.8	36
35	2	1.5	4	7200	38.3	41	6.62	965.7	1.5	37	7200	39.5	41	3.73	765	742	6.6	99.3	37
35	2	1.5	5	7200	37.7	40	5.69	1071	1.8	34	3297.4	40	40	0	409.3	1002	7	84.7	35
40	2	1.5	1	7200	44.5	50	11	594	4.6	41	7200	45.9	50	8.27	350.1	1467	11.7	451.3	42
40	2	1.5	2	7200	44.8	48	6.62	439.8	3	41	7200	45.3	48	5.71	357.1	1054	7	183.2	41.2
40	2	1.5	3	7200	42.2	46	8.19	694.6	2.6	39	4209.6	46	46	0	465.6	1000	4.3	314.4	40
40	2	1.5	4	7200	40.5	46	11.91	573.7	3.2	38	7200	43.1	46	6.25	244.6	1089	6.4	325.9	38
40	2	1.5	5	7200	41.5	49	15.22	669.1	2.2	39	7200	43.8	48	8.7	546.8	1220	7	387.4	39
45	2	1.5	1	7200	44.7	49	8.83	356	8.4	42	7200	45.9	49	6.4	336.6	1046	8.2	420.1	42
45	2	1.5	2	7200	47.5	53	10.38	286	11	46	7200	47.5	53	10.36	180.7	1674	18.9	1170.7	46
45	2	1.5	3	7200	47.3	55	13.95	398.1	7.6	46	7200	49.8	55	9.47	387.4	955	9.4	495	46
45	2	1.5	4	7200	47	54	12.93	290.1	13.7	46	7200	50.3	54	6.85	163.1	2050	16.8	2635.6	46
45	2	1.5	5	7200	47.7	53	9.96	333.8	7.9	46	7200	49.5	53	6.53	139.2	1723	13.4	1661.7	46
50	2	1.5	1	7200	56.6	62	8.71	145.8	11.5	52	7200	57.5	63	8.81	66.3	1892	20.4	1204.9	53
50	2	1.5	2	7200	52.4	59	11.27	152.4	8.4	51	7200	55.2	59	6.52	114.8	1375	13.9	1364.4	51
50	2	1.5	3	7200	52.6	58	9.39	335.7	5.2	48	7200	53.8	58	7.16	111.7	1314	15.4	419.9	50.5
50	2	1.5	4	7200	51.5	60	14.17	148.9	12.4	49	7200	53.6	60	10.59	92.9	1507	16.8	1461.7	49
50	2	1.5	5	7200	52	60	13.33	145.2	12.3	49	7200	54.7	60	8.77	64.3	1825	16.6	1602.8	51
55	2	1.5	1	7200	56.2	64	12.26	91	16.5	53	7200	59.1	64	7.65	32.6	1897	19.3	2266.2	56
55	2	1.5	2	7200	57.4	66	13.11	124.1	11.8	55	7200	61	66	7.64	16.9	2763	35.7	4939.3	57
55	2	1.5	3	7200	56.3	63	10.65	147.9	14.9	54	7200	57.3	63	9.04	46.1	1607	19.7	2175.4	54
55	2	1.5	4	7200	56.8	64	11.24	89.5	12.9	53	7200	58.1	65	10.67	29.9	2001	23.8	1732.2	54.8
55	2	1.5	5	7200	56.4	66	14.52	85.6	16.3	55	7200	58	66	12.1	17.8	2463	29.9	3353.9	55
60	2	1.5	1	7200	63.3	74	14.48	61.7	15.5	62	7200	65.1	73	10.82	2.5	3070	40.9	6452.9	64
60	2	1.5	2	7200	64.8	73	11.19	67.2	18.2	62	7200	64.6	73	11.47	14.1	2600	44.5	3690.7	63
60	2	1.5	3	7200	60.5	72	15.97	50.2	20.6	60	7200	60.4	80	24.5	0	2223	25.8	7200	60.4
60	2	1.5	4	7200	61.1	72	15.19	47.3	30.3	60	7200	63.5	72	11.81	3.2	2712	36	5957.5	63
60	2	1.5	5	7200	60.4	72	16.11	108	17.3	56	7200	63.7	70	9.02	41.2	2087	32.7	2629.6	60.8
AVERAGE				6331.6	47.8	53.4	9.61	385.2	8.6	45.1	5641.9	49.4	53.6	6.74	174.8	1446	15.1	1575.2	46.0

5.4.3 Girth inequalities

Let us first recall the family of girth inequalities (4.53):

$$\sum_{e \in S}(C+1)x_e^i - \sum_{v \in V[S]} C y_v^i \leq 0 \qquad \forall i \in K, j \in V, S \subseteq \Delta_E(j),$$

such that $G[S]$ has girth greater than or equal to $C+1$.

The family of girth inequalities (4.53) also has an exponential number of inequalities, and hence the inclusion of such inequalities to the formulation should be done on demand through the resolution of its associated separation problem. The separation problem associated with girth inequalities (4.53) consists of, given a solution $(\bar{x}, \bar{y})$, decide whether or not there exists an inequality (4.53) that is violated by $(\bar{x}, \bar{y})$. More precisely, one has to decide whether or not there exists a triple (i, j, S), where $i \in K$, $j \in V$ and $S \subseteq \Delta_E(j)$ such that $G[S]$ has girth greater than or equal to $C+1$, for which

$$\sum_{e \in S}(C+1)\bar{x}_e^i - \sum_{v \in V[S]} C \bar{y}_v^i > 0.$$

This means that solving the separation problem associated with girth inequalities is equivalent to finding

$$\omega = \max_{i \in K, j \in V} \left\{ \sum_{e \in S}(C+1)\bar{x}_e^i - \sum_{v \in V[S]} C\bar{y}_v^i : S \subseteq \Delta_E(j), G[S] \text{ has girth greater than } C \right\}, \tag{5.10}$$

and there exists a girth inequality cutting (*i.e.*, violated by) $(\bar{x}, \bar{y})$, if and only if $\omega > 0$. However, it turns out that solving such separation problem is $\mathcal{NP}$-Hard.

Theorem 5.3 - Separation of girth inequalities
The separation problem associated with girth inequalities (4.53) *is $\mathcal{NP}$-Hard.*

Proof. We give a reduction from the Hamiltonian Cycle problem which can be defined as follows. Given a graph $G = (V, E)$, find an Hamiltonian cycle[1] $H = (V, S)$ with $S \subseteq E$. Hamiltonian Cycle is a classic $\mathcal{NP}$-Hard problem (see Karp [103]). Moreover, it has been shown to remain $\mathcal{NP}$-Hard even if graph G is said to be bipartite (see Krishnamoorthy [110]).

Next, we show that given a bipartite graph $G = (V, E)$, any Hamiltonian cycle in G corresponds to a violated girth inequality for the following point $(\overline{x}, \overline{y})$: Let

[1] An Hamiltonian cycle (or Hamiltonian circuit) is a cycle in a graph visiting each vertex exactly once.

(V, E, n) be an instance of Intersection U-SNP. Then, given a vehicle $i \in K$, let $\bar{x}_e^i = \frac{1}{n}$ for all $e \in E$ and $\bar{y}_v^i = \frac{1+\epsilon}{n}$ for all $v \in V$, where $\epsilon \in \mathbb{R}^+$ is an arbitrarily small positive number.

Firstly, notice that since $(V, E, n-1)$ is an instance of Intersection U-SNP, there exists some station $j \in V$ for which $\Delta_E(j) = E$. Therefore, the problem described by (5.10) reduces to

$$\omega = \max_{i \in K} \left\{ \sum_{e \in S} n\bar{x}_e^i - \sum_{v \in V[S]} (n-1)\bar{y}_v^i : S \subseteq E, G[S] \text{ has girth greater than } (n-1) \right\}. \tag{5.11}$$

Moreover, if S is the edge set of an Hamiltonian cycle $H = (V, S)$ in G, then

$$\begin{aligned} \sum_{e \in S} n\bar{x}_e^i - \sum_{v \in V[S]} (n-1)\bar{y}_v^i &= \sum_{e \in S} n\frac{1}{n} - \sum_{v \in V[S]} (n-1)\frac{1+\epsilon}{n} \\ &= |S| - |V[S]|(n-1)\frac{1+\epsilon}{n} \\ &= n - (n-1)(1+\epsilon) \\ &= 1 - \epsilon(n-1) > 0 \end{aligned} \tag{5.12}$$

Therefore, finding an Hamiltonian cycle in G yields a girth inequality violated by $(\overline{x}, \overline{y})$. Conversely, we show that any violated girth inequality corresponds to an Hamiltonian cycle in G. Since $\bar{x}_e^i$ and $\bar{y}_v^i$ have constant values for any $e \in E$ and any $v \in V$, respectively,

$$\sum_{e \in S} n\bar{x}_e^i - \sum_{v \in V[S]} (n-1)\bar{y}_v^i = |S| - |V[S]|(n-1)\frac{1+\epsilon}{n},$$

and such expression is maximized when graph $G[S]$ is the densest possible. However, $G[S]$ is required to have girth greater than or equal to n. Hence, if $G[S]$ contains a cycle, it must be an Hamiltonian cycle. As showed by (5.12), if $G[S]$ is an Hamiltonian cycle, then there exists a girth inequality violated by $(\overline{x}, \overline{y})$. On the other hand, if $G[S]$ is acyclic (a forest), then $n \geq |V[S]| \geq |S| + 1$, and hence

$$\begin{aligned} |S| - |V[S]|(n-1)\frac{1+\epsilon}{n} &\leq |S| - |V[S]|(|V[S]| - 1)\frac{1+\epsilon}{|V[S]|} \\ &\leq |S| - (|S|)(1+\epsilon) = -|S|\epsilon \\ &\leq 0. \end{aligned}$$

Thus if one is able to find a violated girth inequality, then the cycle defined by $G[S]$ solves the Hamiltonian Cycle problem on $G = (V, E)$. ■

Theorem 5.3 shows that, unless $\mathcal{P} = \mathcal{NP}$, an efficient algorithm capable of solving the separation problem associated with girth inequalities (4.53) cannot be achieved. For this reason, such separation problem should be solved heuristically. Next, we propose an heuristic for finding a violated girth inequality.

Given a vehicle $i \in K$ and a station $j \in V$, consider graph $G[\Delta_E(j)]$ with edge weights $w_e = (C+1)\bar{x}_e^i$ for $e \in \Delta_E(j)$. The heuristic proposed for solving the separation problem associated with girth inequalities (4.53) is based on the idea of building a maximum cost spanning tree T of $G[\Delta_E(j)]$ and then including additional edges to T such that the resulting graph does not contain any cycle of size less than or equal to C.

Since a spanning tree is supposed to cover all vertices in the graph, node weights can be ignored in the construction of T. Moreover, notice that

$$\sum_{v \in V[S]} C\bar{y}_v^i \leq \sum_{v \in V[\Delta_E(j)]} C\bar{y}_v^i,$$

for any $S \subseteq \Delta_E(j)$. Therefore, if S is set to $E(T)$, the value of the left-hand side of inequality

$$\sum_{e \in S}(C+1)\bar{x}_e^i - \sum_{v \in V[S]} C\bar{y}_v^i \leq 0,$$

can only be increased by including additional edges to S. An edge uv can only be added to S if it does not creates a cycle in $G[S]$ with size less than or equal to C. This can be verified by computing the shortest path between nodes u and v in $G[S]$ with unit edge lengths. If the shortest path has length greater than or equal to C, then adding uv to S can only create cycles of size strictly greater than C. Algorithm 6 formally describes such heuristic. It takes advantage of functions Kruskal[2] and Dijkstra implemented in `LEMON`[3]. Dijkstra function takes as input a graph G, a cost map corresponding to the edge lengths in G, and two nodes u and v in $V(G)$. It returns the length of the shortest path between such nodes.

Impact of girth inequalities

As expected, the girth inequalities are most impactful when the instance under analysis is dense. However, the influence of the inclusion of girth inequalities into the branch-and-cut framework is much less significant than the inclusion of k-cardinality trees. Table 5.5 provides a comparison between the obtained performance results for dense instances when the separation of girth inequalities is active or not. The

[2]Function Kruskal has been already explained in the description of Algorithm 4 in Section 5.4.2

[3]The documentation of functions Kruskal and Dijkstra can be found in Egerváry Combinatorial Optimization Research Group [63, 62]

Algorithm 6 Separation of girth inequalities (4.53)

Input: An instance (V, E, C) of U-SNP, and a fractional solution $(\bar{x}, \bar{y})$ of the linear relaxation of $\text{USNP}_{(V,E,C)}$
Output: A "good" candidate girth inequality for cutting $(\bar{x}, \bar{y})$

```
for i = 1, ..., p do
  for j = 1, ..., n do
    for all e ∈ Δ_E(j) do
      w_e = (C + 1)x̄^i_e
      length_e = ∞
    end for
    T = Kruskal(G[Δ_E(j)], −w)
    S = E(T)
    for all e ∈ S do
      length_e = 1
    end for
    for all uv ∈ Δ_E(j) \ S do
      if Dijkstra(G[Δ_E(j)], length, u, v) ≥ C then
        length_uv = 1
        S = S ∪ {uv}
      end if
    end for
    if Σ_{e∈S}(C + 1)x̄^i_e − Σ_{v∈V[S]} Cȳ^i_v > 0 then
      Add inequality Σ_{e∈S}(C + 1)x̄^i_e − Σ_{v∈V[S]} Cȳ^i_v ≤ 0 as a cut
    end if
  end for
end for
```

`inactive` scenario stands for the branch-and-cut framework taking into account the separation of strong capacity inequalities and k-cardinality tree inequalities only, while `active` scenario also includes the separation of girth inequalities. Columns $\texttt{time}_g$, $\texttt{LB}_g$, $\texttt{UB}_g$, $\texttt{gap}_g$ and `Nodes` provide information on the global optimization process by showing, respectively, the total time spent in seconds, the lower and upper bounds obtained by the end of the optimization, the remaining gap and the number of nodes investigated in the enumeration tree. Finally, column `Cuts` displays the number of girth inequalities added to the formulation throughout the optimization process.

Table 5.5: Impact of the inclusion of girth inequalities

Instances				inactive					active					
m	C	ρ	i	$\texttt{time}_g$	$\texttt{LB}_g$	$\texttt{UB}_g$	$\texttt{gap}_g$	Nodes	$\texttt{time}_g$	$\texttt{LB}_g$	$\texttt{UB}_g$	$\texttt{gap}_g$	Nodes	Cuts
30	2	4.5	1	12.7	24	24	0	1	8.9	24	24	0	0.5	12
30	2	4.5	2	1002	26	26	0	112.9	638.7	26	26	0	71.4	34
30	2	4.5	3	288.5	26	26	0	29	60	26	26	0	4.7	56
30	2	4.5	4	0.2	27	27	0	0	0.2	27	27	0	0	0
30	2	4.5	5	0.4	27	27	0	0	0.5	27	27	0	0	0
35	2	4.5	1	7200	31	32	3.12	360.5	7200	31	32	3.12	357.3	166
35	2	4.5	2	7200	29	32	9.33	635.2	7200	29	32	9.37	481.2	168

Continued on Next Page...

Table 5.5 – Continued

Instances				inactive					active					
m	C	ρ	i	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	Cuts
35	2	4.5	3	311.2	30	30	0	7.3	2388.8	30	30	0	31.7	252
35	2	4.5	4	28.8	33	33	0	2.1	46.2	33	33	0	1.9	96
35	2	4.5	5	7200	29.1	31	6.23	698.3	7200	29.5	31	4.84	826.4	157
40	2	4.5	1	7200	33.5	35	4.29	337.4	7200	34	35	2.86	404.4	167
40	2	4.5	2	7200	33.5	36	6.94	426.8	7200	33.7	36	6.44	282.3	226
40	2	4.5	3	7200	33	34	2.94	248.4	7200	33	34	2.94	172.4	124
40	2	4.5	4	7200	34	35	2.86	248.3	7200	34	35	2.86	216.6	193
40	2	4.5	5	7200	33	35	5.71	233.2	7200	33	35	5.71	190.2	482
45	2	4.5	1	7200	39	42	7.14	105.9	7200	39	42	7.14	110.4	473
45	2	4.5	2	7200	40.5	42	3.57	126.9	7200	40.2	42	4.29	84.7	764
45	2	4.5	3	7200	39.5	42	5.95	75.5	7200	40	42	4.76	59.7	750
45	2	4.5	4	7200	43	46	6.52	62.5	7200	43	45	4.44	55.5	1450
45	2	4.5	5	7200	36	39	7.58	66.5	7200	36.2	39	7.05	44	240
50	2	4.5	1	7200	44	47	6.38	103.6	7200	44	47	6.38	48.5	835
50	2	4.5	2	7200	41.5	44	5.68	62.7	7200	42	44	4.55	75.5	429
50	2	4.5	3	7200	44.5	48	7.29	67.7	7200	44.5	48	7.29	41.2	476
50	2	4.5	4	7200	44.5	49	9.18	22.9	7200	44.5	48	7.29	7.7	1110
50	2	4.5	5	7200	43	48	10.42	11.7	7200	43	49	12.24	7.3	970
55	2	4.5	1	7200	46	49	6.12	66.5	7200	46	49	6.12	18.1	880
55	2	4.5	2	7200	47.2	51	7.52	29.8	7200	47	51	7.84	19.4	695
55	2	4.5	3	7200	46	51	9.8	32.1	7200	46	51	9.8	17.2	760
55	2	4.5	4	7200	47	53	11.32	7.1	7200	47	53	11.32	0	1
55	2	4.5	5	7200	51	55	7.27	17.9	7200	51	57	10.53	10.5	1211
60	2	4.5	1	7200	52	59	11.86	2.6	7200	52	58	10.34	5.8	316
60	2	4.5	2	7200	49	59	16.95	0	7200	49	59	16.95	0	0
60	2	4.5	3	7200	53	57	7.02	16.8	7200	53	57	7.02	11.9	869
60	2	4.5	4	7200	54	59	8.47	11.9	7200	53.5	59	9.32	6.1	1714
60	2	4.5	5	7200	51	61	16.39	0	7200	51	61	16.39	0	0

The performance results obtained are varied. For some instances (*e.g.*, 30_2_4.5_2 and 30_2_4.5_3) optimality could be proved faster with the inclusion of girth inequalities. On the other hand, there are cases (*e.g.*, 35_2_4.5_3) where the inclusion of such inequalities significantly slows down the performance of the branch-and-cut algorithm. In average, the inclusion of girth inequalities resulted in the reduction of the remaining gap by 0.13%. Moreover, considering only the instances that could be solved to optimality within the time limit, the number of nodes required to be explored in order to prove optimality went down from 21,757 to 15,743 in average.

The relative small number of cuts generated by the separation procedure is explained by the fact that Algorithm 6 is executed only when no violated strong capacity inequality nor k-cardinality tree inequality is identified. This explains the fact that for some instances (*e.g.*, 60_2_4.5_2 and 60_2_4.5_5) no girth inequality was included and the obtained performances are exactly the same for both scenarios.

5.4.4 Further considerations

The separation of other families of valid inequalities identified in Chapter 4 such as the generalized k-cardinality tree inequalities (4.50) or the constructed valid inequalities presented in Section 4.4.5 are not implemented in our Branch-and-Cut

framework. Some work has been dedicated to the inclusion of such additional inequalities, but the preliminary results achieved so far do not yield better computational performances. In Appendix B, some of these results are briefly discussed.

Finally, it is likely that better performances could be achieved by taking into consideration such additional inequalities and including them using more refined heuristic separation procedures. The development of such procedures is by itself a relevant area of research that can be addressed in future works.

5.5 Final results

The final version of the developed Branch-and-Cut framework takes into account the strong capacity inequalities (4.34), (4.36), the k-cardinality tree inequalities (4.40) and the girth inequalities (4.53), (4.54). Such inequalities are added on demand by applying the algorithms presented in the previous section over each LP relaxation solution obtained during the optimization. For this, the algorithms are implemented within the `UserCutCallback` routine from CPLEX that is called after solving each node LP in the enumeration tree. Algorithms 5 and 6 are applied for solving the separation problems associated with inequalities (4.40) and (4.53), respectively. Since such algorithms are relatively slower if compared to the separation problem associated with strong capacity inequalities, they are only called when no additional strong capacity inequality is able of cutting off the given LP solution.

In addition, in order to avoid spending too much time in the search for valid inequalities, we choose to not apply the separation routine on every node LP. Such approach is widely applied in the literature and the strategy for deciding whether or not the separation routine should be called varies according to the problem under analysis. For instance, in Cordeau [37] the separation routine is only applied if a given subset of variables is integer in the current LP solution. In our case, the following scheme is applied. An integer parameter named *pace* is used to guide the strategy. Parameter *pace* indicates the rhythm on which separation routine should be skipped. That is, if $pace = i$, then the separation routine is applied on every i nodes. Such parameter is initialized at 1 at the beginning of the optimization process, meaning that the separation routine should be called on every node at first. Once the separation routine is unable to find any violated cut, skipping separation is stimulated by multiplying[4] *pace* by 2. Finally, whenever the separation routine finds again a violated cut, *pace* is reset to 1. Such strategy allows the optimization process to adapt itself to the instance being solved by ignoring the separation routine if it

[4]The choice of multiplying by 2 is merely empiric. In fact, one may multiply such parameter by any constant $c \geq 1$ in order to obtain a faster or slower progression of *pace*.

is not capable of reinforcing the formulation.

Symmetry is handled by fixing variables on each branching phase, through the Orbitopal Fixing procedure described in Section 5.1. For this, Algorithms 2 and 3 are then implemented within the `BranchCallback` routine from CPLEX that is called after a branch has been selected but before the branch is effectively carried out during the optimization of a MILP. Moreover, a significant amount of variables are eliminated by applying the upper bound on p_{opt} identified in Section 5.2.

In Section 5.3, the difference of performances obtained from the interference in the variable choice for branching is reported. Based on such results, we choose to relax variables y and apply formulation USNP-X$_{(V,E,C)}$ when the instance is sparse (*i.e.*, $\rho = 1.5$). For every other instance, we choose to impose a branching preference on variables y and apply formulation USNP-Y$_{(V,E,C)}$.

The following table provides a comparison between the results achieved by the developed Branch-and-Cut framework (section `Branch-and-Cut`) and the ones achieved by CPLEX applied to the original formulation of Pimenta et al. [151] with no user-cut generation and a dummy callback (section `CPLEX`). Results are given for each one of the 315 generated instances. Columns `time`$_g$, `LB`$_g$, `UB`$_g$, `gap`$_g$ and `Nodes` provide information on the global optimization process by showing, respectively, the total time spent in seconds, the lower and upper bounds obtained by the end of the optimization, the remaining gap and the number of nodes investigated in the enumeration tree. Column `cuts`$_{CP}$ refers to the number of cuts generated by CPLEX itself that were added during the optimization. Columns `cuts`$_{SC}$, `cuts`$_{KT}$ and `cuts`$_G$ stand for the number of violated strong capacity inequalities, k-cardinality tree inequalities and girth inequalities included throughout the optimization, respectively. Finally, columns `cuts`$_{tot}$ and `time`$_{cut}$ provides the total number of user cuts added and the time spent solving the respective separation problems.

Table 5.6: Final results

Instances				CPLEX						Branch-and-Cut										
m	C	ρ	i	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	cuts$_{SC}$	cuts$_{KT}$	cuts$_G$	cuts$_{tot}$	time$_{cut}$
30	2	1.5	1	7200	22.6	35	35.33	808	143	36.5	35	35	0	1.9	0	143	491	81	715	2.5
30	2	1.5	2	7200	23.5	34	30.98	1114.2	123	1970.7	34	34	0	216.7	0	137	596	104	837	4.3
30	2	1.5	3	7200	22.5	36	37.63	782.2	139	234.2	36	36	0	14.2	0	176	507	70	753	2.1
30	2	1.5	4	7200	22.3	34	34.38	910.9	131	105.8	34	34	0	7.8	0	128	359	70	557	1.5
30	2	1.5	5	7200	23.9	36	33.72	782.9	105	3428	36	36	0	839.8	0	128	579	64	771	5
35	2	1.5	1	7200	25.1	41	38.87	591.1	109	6007	41	41	0	433.1	0	207	835	75	1117	7
35	2	1.5	2	7200	27.7	44	36.94	274.9	503	7200	41	44	6.82	401.5	0	247	939	120	1306	9.3
35	2	1.5	3	7200	24.8	40	38.02	666.7	123	55.4	40	40	0	0.6	0	189	380	41	610	2.3
35	2	1.5	4	7200	24.8	41	39.39	354.4	152	7200	39.3	41	4.13	535.8	0	246	872	81	1199	6.3
35	2	1.5	5	7200	25.1	40	37.32	427	417	6452.4	40	40	0	345.7	0	206	1070	95	1371	9.8
40	2	1.5	1	7200	26.8	51	47.42	356.9	15	7200	45.5	50	9.07	308.4	0	252	1662	115	2029	19.4
40	2	1.5	2	7200	27.4	48	42.99	234.1	168	7200	45.3	48	5.56	309.3	0	252	1105	109	1466	13.1
40	2	1.5	3	7200	28.7	46	37.68	323.6	182	3701.5	46	46	0	232.6	0	252	1070	130	1452	6.6
40	2	1.5	4	7200	26.9	46	41.45	236.7	210	7200	43	46	6.57	319.2	0	252	1078	128	1458	10.5
40	2	1.5	5	7200	27.9	48	41.95	229.3	188	7200	44.5	49	9.26	381.3	0	210	1250	124	1584	9
45	2	1.5	1	7200	29.8	50	40.36	141.1	236	7200	45.7	49	6.79	206.4	0	287	1318	168	1773	14.5
45	2	1.5	2	7200	29.8	53	43.69	112.6	181	7200	48.4	53	8.6	112.8	0	359	1679	102	2140	19.4
45	2	1.5	3	7200	32.4	55	41.14	119.2	197	7200	49.8	55	9.36	249.3	0	335	958	142	1435	12.9
45	2	1.5	4	7200	30.3	55	44.87	92.3	163	7200	49.6	54	8.13	111.9	0	431	1909	118	2458	20.2
45	2	1.5	5	7200	31.1	53	41.37	122	209	7200	49.4	53	6.81	89.2	0	359	1737	174	2270	18.5
50	2	1.5	1	7200	35.5	63	43.63	59.5	231	7200	57	62	8	87.5	0	390	2109	335	2834	52.9
50	2	1.5	2	7200	33.8	60	43.65	62.8	224	7200	53.7	59	9	142.2	0	388	1407	195	1990	20.7
50	2	1.5	3	7200	32.7	58	43.66	97.2	201	7200	53.9	58	7.08	104.1	0	257	1234	142	1633	14.6
50	2	1.5	4	7200	32.7	62	47.2	36.8	148	7200	52.8	60	12.06	101.5	0	442	1534	179	2155	26.5
50	2	1.5	5	7200	33.3	61	45.46	83.7	245	7200	54.8	60	8.72	147.7	0	464	1775	95	2334	21.9
55	2	1.5	1	7200	37.2	65	42.8	47.4	290	7200	58.6	64	8.39	36.2	0	435	1913	171	2519	24.9
55	2	1.5	2	7200	38.3	67	42.85	59.8	300	7200	60	66	9.03	10.7	0	435	2725	149	3309	46.3
55	2	1.5	3	7200	36.8	63	41.64	40.9	224	7200	57	62	8.12	101.5	0	406	1652	179	2237	26.6
55	2	1.5	4	7200	36.2	65	44.38	43.8	274	7200	57.7	64	9.81	52.2	0	435	2048	160	2643	34.4
55	2	1.5	5	7200	34.9	66	47.16	52.3	203	7200	58	65	10.85	29.2	0	493	2360	220	3073	47
60	2	1.5	1	7200	40.6	74	45.17	17.2	306	7200	65.1	73	10.84	4.3	0	548	2812	135	3495	43.6
60	2	1.5	2	7200	40.6	74	45.1	37.8	216	7200	65.2	73	10.68	13.2	0	555	2597	329	3481	63.3
60	2	1.5	3	7200	39	72	45.83	24.4	240	7200	60	75	20	0	0	620	2027	0	2647	21.2
60	2	1.5	4	7200	41.3	72	42.58	34.5	369	7200	63	76	17.11	0	0	558	2648	0	3206	30.3
60	2	1.5	5	7200	40	71	43.62	27.3	284	7200	63.4	70	9.36	69.2	0	496	1973	318	2787	49.8
30	5	1.5	1	7200	19.5	25	22.08	1318.6	40	27.3	25	25	0	2.9	0	16	739	101	856	2.1
30	5	1.5	2	7200	22.4	27	16.99	1255.4	8	14	27	27	0	0.9	3	0	398	52	450	0.7
30	5	1.5	3	7200	22.6	27	16.25	1222.2	33	4.4	27	27	0	0.4	1	11	266	39	316	0.8
30	5	1.5	4	7200	21.4	27	20.58	1108.9	12	17.8	27	27	0	4	0	11	247	11	269	0.9

Continued on Next Page...

Table 5.6 – Continued

Instances				CPLEX						Branch-and-Cut										
m	C	ρ	i	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	cuts$_{SC}$	cuts$_{KT}$	cuts$_G$	cuts$_{tot}$	time$_{cut}$
30	5	1.5	5	7200	22.2	28	20.67	1138.5	6	55.5	28	28	0	6.2	2	0	637	28	665	1.1
35	5	1.5	1	7200	23.4	31	24.6	627.6	26	571	31	31	0.01	24.8	0	24	1387	440	1851	11.5
35	5	1.5	2	7200	25.1	31	18.98	711.2	7	185.8	31	31	0	16.8	0	0	711	61	772	2.1
35	5	1.5	3	7200	23.2	29	19.96	712.2	7	78.3	29	29	0.01	9.4	1	12	424	37	473	1.8
35	5	1.5	4	7200	25.2	33	23.71	803.9	7	2590	33	33	0.01	90	0	0	2244	458	2702	16.2
35	5	1.5	5	7200	24.1	31	22.18	828.1	14	309.8	31	31	0	23.8	2	13	973	99	1085	3.7
40	5	1.5	1	7200	27.7	37	25	572.7	6	6112.2	37	37	0.01	281.5	2	15	1283	109	1407	10.1
40	5	1.5	2	7200	27.5	36	23.56	461.3	11	190.5	36	36	0.01	31.5	0	0	112	5	117	0.3
40	5	1.5	3	7200	27.5	37	25.55	441.1	4	7200	33.8	37	8.56	387.3	0	0	1627	215	1842	12
40	5	1.5	4	7200	27.5	37	25.55	503.8	7	2368.9	37	37	0.01	170.2	0	15	871	41	927	4.4
40	5	1.5	5	7200	26.7	33	19.13	552.2	19	310.8	33	33	0	26.3	0	18	483	36	537	2.4
45	5	1.5	1	7200	30.5	40	23.75	337.9	4	7200	36.9	40	7.83	390.4	3	0	1400	187	1587	10.2
45	5	1.5	2	7200	31.7	41	22.74	377.9	34	7200	38	41	7.24	186	0	30	2999	524	3553	41.6
45	5	1.5	3	7200	29.5	41	27.98	204.4	18	7200	37.7	41	7.99	197.7	0	16	1960	435	2411	29.8
45	5	1.5	4	7200	31.6	42	24.73	284.5	13	7200	38.7	42	7.82	319.5	0	16	1768	132	1916	14.8
45	5	1.5	5	7200	29.6	43	31.25	218.4	15	7200	37.4	43	12.93	153.4	1	0	2712	366	3078	49.8
50	5	1.5	1	7200	33	46	28.26	192.6	5	7200	41	46	10.78	132	0	34	3406	727	4167	75.4
50	5	1.5	2	7200	33.9	47	27.84	146.1	27	7200	42.4	46	7.79	158.2	1	16	1525	230	1771	14.1
50	5	1.5	3	7200	34	46	26.09	169	3	7200	42.7	46	7.13	258.4	0	0	1276	138	1414	12
50	5	1.5	4	7200	32	48	33.42	120.8	13	7200	40.5	48	15.72	93.3	0	18	4002	614	4634	93.1
50	5	1.5	5	7200	32.9	48	31.52	133.5	12	7200	41.2	48	14.23	82	0	0	4225	213	4438	68.7
55	5	1.5	1	7200	35.8	52	31.07	69.2	19	7200	45	51	11.73	195.6	1	19	1701	152	1872	33.1
55	5	1.5	2	7200	35.8	51	29.72	58.9	59	7200	45.2	52	13.07	42.7	0	60	6032	499	6591	138.6
55	5	1.5	3	7200	36.9	50	26.28	116.8	37	7200	44.4	50	11.2	56.9	0	20	2831	831	3682	67.5
55	5	1.5	4	7200	33.8	50	32.32	67.7	67	7200	42.2	50	15.52	31.2	1	39	6316	1789	8144	174.2
55	5	1.5	5	7200	36.9	52	29.11	103.2	42	7200	44.8	52	13.91	110.3	4	20	3404	367	3791	54.9
60	5	1.5	1	7200	36.8	50	26.42	66.3	9	7200	44.8	50	10.48	54.8	0	41	3122	353	3516	47.6
60	5	1.5	2	7200	38.8	54	28.11	53.7	11	7200	48.1	53	9.25	47.9	0	40	3254	400	3694	59.9
60	5	1.5	3	7200	38.8	57	31.85	54.8	23	7200	48.5	56	13.47	61.9	0	39	4211	830	5080	92.8
60	5	1.5	4	7200	37.8	56	32.42	59.4	9	7200	46.8	54	13.37	87.6	0	21	3324	180	3525	52.3
60	5	1.5	5	7200	38.9	58	32.99	67.9	15	7200	48.7	57	14.54	33.3	0	0	5635	596	6231	142.6
30	8	1.5	1	2351.4	24	24	0	538	4	2	24	24	0	0.1	1	0	190	9	199	0.2
30	8	1.5	2	3311.8	23	23	0	569.7	5	1.1	23	23	0	0.1	2	0	74	14	88	0.1
30	8	1.5	3	7200	21.2	24	11.7	1037.3	11	0.8	24	24	0	0.3	2	0	32	2	34	0
30	8	1.5	4	7200	20.5	23	10.84	1200.1	6	1	23	23	0	0.6	3	0	15	0	15	0
30	8	1.5	5	3134.4	24	24	0	494.7	6	1.1	24	24	0	0.2	0	0	62	6	68	0.1
35	8	1.5	1	7200	25	29	13.95	701.3	4	5.1	29	29	0	1.1	0	0	129	7	136	0.4
35	8	1.5	2	7200	23.8	25	4.73	821.8	6	2.6	25	25	0	0	1	0	195	5	200	0.3
35	8	1.5	3	7200	24.7	29	14.7	638.2	4	12.4	29	29	0	0.2	2	0	858	87	945	2.3

Continued on Next Page...

Table 5.6 – Continued

Instances				CPLEX						Branch-and-Cut										
m	C	ρ	i	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	cuts$_{SC}$	cuts$_{KT}$	cuts$_G$	cuts$_{tot}$	time$_{cut}$
35	8	1.5	4	7200	24.3	27	9.88	744.5	7	10.8	27	27	0	0.1	2	0	567	144	711	1.7
35	8	1.5	5	7200	24	28	14.14	920.7	11	13	28	28	0	1.4	2	0	356	44	400	0.7
40	8	1.5	1	7200	26.6	32	16.96	394.5	3	9.7	32	32	0	1.9	0	0	147	9	156	0.3
40	8	1.5	2	7200	26.7	31	13.98	504.9	6	26.2	31	31	0	1.1	0	0	745	93	838	1.7
40	8	1.5	3	7200	24.4	31	21.2	457.5	5	466.8	31	31	0	19.7	0	0	1905	135	2040	8.1
40	8	1.5	4	7200	27.7	34	18.55	513.5	5	338	34	34	0	27.3	4	0	917	127	1044	4.8
40	8	1.5	5	7200	27.6	32	13.69	451.2	3	4.3	32	32	0	0.4	2	0	157	18	175	0.3
45	8	1.5	1	7200	31.3	38	17.62	259.2	4	227.6	38	38	0.01	13.4	0	0	1197	103	1300	5.2
45	8	1.5	2	7200	31.4	38	17.47	225.6	9	131.1	38	38	0	6.2	2	0	914	26	940	4.2
45	8	1.5	3	7200	28.4	36	21.22	289.7	7	4217.1	36	36	0.01	60.1	0	0	3156	881	4037	27.5
45	8	1.5	4	7200	29.3	36	18.6	263.7	2	346.3	36	36	0	13.7	0	0	1039	123	1162	6.7
45	8	1.5	5	7200	30.3	36	15.91	366.1	3	26.9	36	36	0	1.5	0	0	384	27	411	1.5
50	8	1.5	1	7200	33.1	41	19.3	193.7	4	1625.2	41	41	0.01	59.6	0	0	1216	129	1345	13.1
50	8	1.5	2	7200	31	38	18.42	191.7	6	1014.1	38	38	0.01	24.7	4	0	2266	67	2333	16.9
50	8	1.5	3	7200	31	39	20.51	176.2	5	835.9	39	39	0	23.5	0	0	1306	206	1512	10
50	8	1.5	4	7200	32	41	21.95	161.5	6	7200	39.6	41	3.41	239	0	0	2300	171	2471	23.7
50	8	1.5	5	7200	33	42	21.43	161.1	8	7200	40.2	42	4.18	158.6	0	0	2595	293	2888	37
55	8	1.5	1	7200	34.7	44	21.06	99.9	3	7200	40.8	44	7.32	162.1	2	0	2983	260	3243	32.7
55	8	1.5	2	7200	35.8	45	20.43	149.7	5	7200	42	45	6.63	194.1	0	0	2502	345	2847	29.7
55	8	1.5	3	7200	35.7	44	18.86	105	6	816.7	44	44	0	53.4	1	0	515	59	574	6.1
55	8	1.5	4	7200	36.7	46	20.16	94.4	4	7200	43.9	46	4.59	295	0	0	1448	81	1529	15.1
55	8	1.5	5	7200	34.8	47	26	64.5	7	7200	42.2	47	10.27	46.8	0	0	3821	2191	6012	130
60	8	1.5	1	7200	37.8	51	25.94	58.5	6	7200	45.2	51	11.35	36.7	0	0	6372	785	7157	111
60	8	1.5	2	7200	38.7	50	22.5	70.4	8	7200	45	50	9.96	34.5	0	0	6832	1865	8697	115.3
60	8	1.5	3	7200	40.8	53	23.06	54.3	4	7200	47.5	52	8.7	65	0	0	4248	1023	5271	98
60	8	1.5	4	7200	39.8	50	20.47	59	6	7200	46.1	50	7.73	71.1	1	0	4715	862	5577	77.9
60	8	1.5	5	7200	36.7	49	25.02	72.7	12	7200	43.8	49	10.68	64	0	11	6946	647	7604	89.1
30	2	3	1	7200	18	31	42.06	775	170	7200	28.7	31	7.55	1155.4	9	160	509	82	751	6.1
30	2	3	2	7200	16.5	29	43.01	834.1	131	1398.6	29	29	0	78.7	3	165	454	47	666	3.1
30	2	3	3	7200	15.2	31	50.94	806.6	169	7200	29.2	31	5.66	706.8	8	150	642	109	901	5.9
30	2	3	4	7200	15.1	27	43.98	636.1	157	111.6	27	27	0	5.4	0	144	482	107	733	2.8
30	2	3	5	7200	15.9	30	46.9	710.6	158	434.8	30	30	0	30.7	3	160	1130	114	1404	2.8
35	2	3	1	7200	14.1	35	59.6	616.4	121	7200	32.6	35	6.83	451.5	3	228	1389	156	1773	9.6
35	2	3	2	7200	14.4	33	56.44	443.6	119	173	33	33	0	7	5	247	818	73	1138	4.3
35	2	3	3	7200	18.8	35	46.43	365.7	189	3447.8	35	35	0	159.2	1	209	1036	244	1489	10.7
35	2	3	4	7200	17.7	32	44.79	590.4	161	25.5	32	32	0	0.2	0	203	420	71	694	1.8
35	2	3	5	7200	16.4	33	50.4	277.2	172	7200	32	33	3.03	841	7	208	680	42	930	6.1
40	2	3	1	7200	16.7	38	55.98	244	161	7200	34.8	38	8.39	168.3	4	294	1861	36	2191	12.2

Continued on Next Page...

Table 5.6 – Continued

Instances				CPLEX						Branch-and-Cut										
m	C	ρ	i	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	cuts$_{SC}$	cuts$_{KT}$	cuts$_G$	cuts$_{tot}$	time$_{cut}$
40	2	3	2	7200	16.1	37	56.52	219.6	187	7200	34.4	37	6.99	123.5	4	315	1583	184	2082	14.2
40	2	3	3	7200	18.5	40	53.85	299.7	199	7200	36.6	40	8.55	115.7	2	315	2340	151	2806	13.4
40	2	3	4	7200	16.8	40	57.9	242	136	7200	37	40	7.5	236.8	2	272	1054	243	1569	20.2
40	2	3	5	7200	15.3	39	60.74	112.1	173	7200	33.9	39	12.96	118	3	315	2344	142	2801	19.6
45	2	3	1	7200	17.4	44	60.49	69.8	289	7200	39.7	44	9.85	125.2	1	360	1848	132	2340	24
45	2	3	2	7200	18.9	47	59.82	129.4	218	7200	42	47	10.55	19	3	384	3221	332	3937	38.7
45	2	3	3	7200	17.7	47	62.44	135.8	159	7200	42.8	47	8.84	64.2	2	336	2244	292	2872	37.7
45	2	3	4	7200	16.9	42	59.67	160.4	188	7200	39.5	42	5.97	147.1	0	427	1515	136	2078	26.7
45	2	3	5	7200	17.3	45	61.56	123.1	179	7200	40	45	11.05	67.5	1	408	1999	107	2514	19
50	2	3	1	7200	19.9	50	60.17	75.1	251	7200	45.5	50	9	34.9	1	490	2134	315	2939	45.4
50	2	3	2	7200	20.5	55	62.71	78	240	7200	48	55	12.73	0	0	390	2938	0	3328	14.2
50	2	3	3	7200	17.9	50	64.3	66.4	167	7200	43	49	12.24	9.8	0	416	2707	229	3352	30.5
50	2	3	4	7200	18.6	49	62.08	52.5	229	7200	45.3	49	7.64	15.7	4	416	2782	419	3617	52
50	2	3	5	7200	18.2	53	65.62	37.1	315	7200	43	52	17.31	0	0	468	2266	0	2734	10.3
55	2	3	1	7200	20.4	56	63.65	29.2	297	7200	48	57	15.79	0	0	609	1827	0	2436	11.5
55	2	3	2	7200	19.9	55	63.89	33.9	283	7200	48	57	15.79	0	0	551	2085	0	2636	12.2
55	2	3	3	7200	20.2	58	65.15	22.9	228	7200	49	61	19.67	0	0	551	1759	0	2310	11.4
55	2	3	4	7200	19.2	55	65.02	37.3	263	7200	48	58	17.24	0	0	609	1745	0	2354	11.8
55	2	3	5	7200	19.2	57	66.37	38.9	293	7200	50	56	10.71	0	0	551	2351	1	2903	16.1
60	2	3	1	7200	22	65	66.08	22.5	303	7200	56	67	16.42	0	0	744	1511	0	2255	13.4
60	2	3	2	7200	21.5	63	65.93	16	295	7200	54	63	14.29	0	0	651	1621	0	2272	13.7
60	2	3	3	7200	22.8	65	64.86	14.8	294	7200	55	64	14.06	0	0	651	1550	0	2201	13.1
60	2	3	4	7200	21.9	64	65.77	32.8	270	7200	54	66	18.18	0	0	651	1403	0	2054	11.3
60	2	3	5	7200	21.3	64	66.75	21.3	357	7200	56	64	12.5	0	0	651	1468	0	2119	12.7
30	5	3	1	7200	12.8	19	32.43	1040.6	59	64.3	19	19	0	14.5	2	43	476	53	572	1.2
30	5	3	2	7200	12.5	19	34.21	865.3	42	23.7	19	19	0	5.8	1	22	391	32	445	1
30	5	3	3	7200	12.8	18	28.76	1064.4	60	62.1	18	18	0	2.8	1	33	1760	390	2183	4.3
30	5	3	4	7200	12.8	18	29	1003.8	51	55	18	18	0	8.9	2	40	568	141	749	1.5
30	5	3	5	7200	12.7	19	32.97	1080.7	24	197.6	19	19	0	34.6	5	33	877	119	1029	1.8
35	5	3	1	7200	13.2	22	39.77	588.6	14	762.7	22	22	0	19.5	6	52	3491	214	3757	9.5
35	5	3	2	7200	13.5	21	35.77	760.5	32	144.8	21	21	0	31.4	3	52	287	49	388	1.8
35	5	3	3	7200	13.7	23	40.6	537.6	76	7200	21.2	23	7.67	438	9	65	1662	304	2031	10.9
35	5	3	4	7200	13.4	22	39.11	627.9	52	7200	19.3	22	12.09	147.8	1	52	3460	1340	4852	28.3
35	5	3	5	7200	13.5	21	35.69	574.4	75	278.3	21	21	0	42.8	5	43	425	63	531	1.9
40	5	3	1	7200	16	26	38.38	361.8	91	7200	23.9	26	7.91	133.1	1	75	3675	975	4725	37.3
40	5	3	2	7200	14.8	25	40.87	366	34	7200	23.1	25	7.75	252.1	2	73	2165	242	2480	15.9
40	5	3	3	7200	15	25	39.93	331.4	84	7200	22.4	25	10.38	223.8	4	74	1815	903	2792	22
40	5	3	4	7200	15	25	39.99	363.8	73	533.1	25	25	0	20.5	2	71	1400	600	2071	12.4
40	5	3	5	7200	15.2	26	41.52	325.5	87	7200	23.3	26	10.28	72.4	2	105	4870	1403	6378	43.8

Continued on Next Page...

Table 5.6 – Continued

Instances				CPLEX						Branch-and-Cut										
m	C	ρ	i	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	cuts$_{SC}$	cuts$_{KT}$	cuts$_G$	cuts$_{tot}$	time$_{cut}$
45	5	3	1	7200	16.5	31	46.79	198.1	87	7200	24.8	31	19.99	23.3	2	64	6931	3147	10142	92.3
45	5	3	2	7200	16.6	31	46.31	206.5	55	7200	24.7	31	20.32	52.8	1	78	4738	1751	6567	79.9
45	5	3	3	7200	16.8	30	44	183.9	91	7200	26.3	30	12.44	61.9	1	95	3375	2139	5609	69.1
45	5	3	4	7200	16.5	32	48.38	186.4	37	7200	25.9	32	19.1	9.9	0	96	9478	6379	15953	160.4
45	5	3	5	7200	16.7	31	46.24	240.5	66	7200	24.7	31	20.17	40.2	1	48	5542	3371	8961	96.2
50	5	3	1	7200	17.3	31	44.19	146.6	54	7200	26.6	33	19.48	37.6	1	108	6303	1650	8061	83.5
50	5	3	2	7200	17	35	51.43	114	77	7200	26.9	38	29.24	5.9	0	108	11041	5407	16556	244.6
50	5	3	3	7200	17	34	50	116	50	7200	26.4	36	26.77	19.2	3	88	7859	2957	10904	146.6
50	5	3	4	7200	17.3	34	49.24	116.2	104	7200	26.7	34	21.57	20.2	5	72	4915	4336	9323	159.6
50	5	3	5	7200	17.3	30	42.43	156.7	67	7200	26.5	30	11.73	107.8	4	125	2848	542	3515	51.5
55	5	3	1	7200	18.9	39	51.66	61.2	115	7200	29.8	41	27.35	7.9	0	140	9647	5309	15096	260.3
55	5	3	2	7200	18.8	36	47.65	70.3	109	7200	28	38	26.44	16.2	1	140	9654	3838	13632	194
55	5	3	3	7200	18.8	38	50.41	73.7	97	7200	29.3	40	26.8	16.5	1	140	8754	2055	10949	151.6
55	5	3	4	7200	18.8	34	44.64	83.1	63	7200	29.4	36	18.34	22.4	0	87	6170	2027	8284	125.8
55	5	3	5	7200	18.9	36	47.63	84.7	75	7200	30.3	36	15.81	54.3	3	114	5331	922	6367	145.6
60	5	3	1	7200	20.9	43	51.5	43.2	175	7200	31.5	45	29.93	14.2	3	116	10455	2848	13419	323.9
60	5	3	2	7200	20.8	42	50.36	45.4	116	7200	31.5	42	25.1	16.7	0	123	6882	2916	9921	214.9
60	5	3	3	7200	20.9	41	49.13	33.9	141	7200	32.2	43	25.02	10.2	0	168	11175	3280	14623	346.1
60	5	3	4	7200	20.8	39	46.54	52.4	82	7200	31.2	40	22.04	31.3	2	63	6749	1593	8405	160.9
60	5	3	5	7200	20.9	44	52.61	46.8	47	7200	31.6	44	28.19	14.5	0	111	6502	4316	10929	234
30	8	3	1	7200	12.4	16	22.45	982.9	32	0.4	16	16	0	0.2	8	7	2	0	9	0
30	8	3	2	233.4	13	13	0	35.3	27	0.2	13	13	0	0	3	0	0	0	0	0
30	8	3	3	1955.5	14	14	0	357	3	0.6	14	14	0	0.1	1	0	69	0	69	0.1
30	8	3	4	7200	12.5	15	16.41	1131	33	0.5	15	15	0	0	1	7	4	0	11	0
30	8	3	5	7200	12.5	15	16.47	1097.6	15	0.5	15	15	0	0.2	5	0	18	0	18	0
35	8	3	1	7200	13	18	27.78	665.1	39	2	18	18	0	0.5	2	13	128	9	150	0.3
35	8	3	2	7200	13.2	18	26.46	949.5	4	17.6	18	18	0	8.2	4	0	169	10	179	0.4
35	8	3	3	201.5	14	14	0	34.4	3	0.3	14	14	0	0	1	0	8	0	8	0
35	8	3	4	7200	12.9	17	24.12	686.2	37	1.3	17	17	0	0.7	2	15	18	0	33	0.1
35	8	3	5	7200	12.8	18	28.89	707.9	7	5.2	18	18	0	2.4	4	0	70	4	74	0.1
40	8	3	1	7200	14.5	22	34.12	402.4	8	423.2	22	22	0	92.5	2	0	628	30	658	2.5
40	8	3	2	7200	14.8	20	25.91	396.9	39	6.3	20	20	0	1.3	3	9	288	9	306	0.9
40	8	3	3	7200	14.5	23	36.97	386.8	5	646.7	23	23	0	59.2	11	0	1184	343	1527	7.1
40	8	3	4	7200	14.7	22	33.27	379.1	6	781.2	22	22	0	17.4	6	0	3155	1132	4287	18.9
40	8	3	5	7200	14.7	22	33.22	390.3	6	385.5	22	22	0	34.3	14	9	1231	129	1369	5.9
45	8	3	1	7200	16.5	25	34.13	276.9	21	416.2	25	25	0	16.1	3	30	1927	352	2309	16.4
45	8	3	2	7200	16.4	26	36.98	251.4	11	5106.7	26	26	0	254.2	4	19	2182	152	2353	16.1
45	8	3	3	7200	16.3	24	32.2	244.2	4	447.3	24	24	0	9.9	1	0	2816	315	3131	14.6
45	8	3	4	7200	16.4	25	34.29	279.3	16	3547.6	25	25	0	25.5	2	20	6503	412	6935	43.8

Continued on Next Page...

Table 5.6 – Continued

Instances				CPLEX						Branch-and-Cut										
m	C	ρ	i	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	cuts$_{SC}$	cuts$_{KT}$	cuts$_G$	cuts$_{tot}$	time$_{cut}$
45	8	3	5	7200	16.5	24	31.44	280.9	78	814.7	24	24	0	38	2	20	1506	609	2135	18
50	8	3	1	7200	17	28	39.29	155.6	6	7200	23.5	28	16.18	50.3	4	11	7749	1387	9147	63.1
50	8	3	2	7200	17.1	28	38.89	170.7	6	7200	23.7	28	15.21	39	2	0	7735	1952	9687	77.5
50	8	3	3	7200	17	28	39.29	170.6	32	7200	24.1	29	17.02	26	1	22	10719	1842	12583	94.9
50	8	3	4	7200	17	28	39.29	163.1	11	7200	25.2	29	13.11	94.7	1	22	5428	373	5823	50.2
50	8	3	5	7200	17.1	29	41.21	152	11	7200	24.5	29	15.39	64.3	0	22	5785	1022	6829	59.1
55	8	3	1	7200	18.8	32	41.35	99.7	10	7200	25.2	32	21.1	62.7	5	24	6806	1554	8384	114.9
55	8	3	2	7200	18.7	31	39.57	97.7	6	7200	23.4	30	21.91	20.5	0	9	9355	1413	10777	109.9
55	8	3	3	7200	18.7	31	39.54	86.8	5	7200	24.7	31	20.24	14.2	2	0	12251	3399	15650	158.9
55	8	3	4	7200	18.8	33	43.15	96	11	7200	26.6	33	19.46	23.4	0	12	10409	1752	12173	118.2
55	8	3	5	7200	18.8	34	44.79	75.6	8	7200	26.5	34	21.93	21.2	3	24	12903	2925	15852	204.8
60	8	3	1	7200	20.8	37	43.8	57.9	47	7200	27.3	37	26.18	17.2	2	13	6835	4053	10901	152.6
60	8	3	2	7200	20.8	36	42.32	56.1	9	7200	26.9	36	25.35	12.6	2	0	11743	3391	15134	201.3
60	8	3	3	7200	19.8	35	43.46	51.1	39	7200	26.8	36	25.62	14	0	13	11601	5844	17458	283.9
60	8	3	4	7200	20.7	34	38.99	63	6	7200	27	34	20.49	16.2	3	0	11422	1723	13145	142.5
60	8	3	5	7200	20.8	36	42.27	63.1	8	7200	27.3	39	29.99	5.8	1	0	18393	7957	26350	334.3
30	2	4.5	1	7200	15.8	24	34.17	1251.6	124	10.7	24	24	0	0.9	1	141	333	18	492	0.8
30	2	4.5	2	7200	15.8	26	39.1	954.9	142	82	26	26	0	14.7	11	142	274	22	438	1.2
30	2	4.5	3	7200	15.5	26	40.35	1064.8	138	36.1	26	26	0	2.8	2	127	370	50	547	1.2
30	2	4.5	4	7200	16.5	27	38.89	1055	154	0.2	27	27	0	0	6	113	0	0	113	0
30	2	4.5	5	7200	17.1	27	36.71	937.6	149	0.4	27	27	0	0	5	104	2	0	106	0
35	2	4.5	1	7200	19.3	32	39.74	253.8	745	7200	31	32	3.12	433.1	6	165	1052	160	1377	9.1
35	2	4.5	2	7200	17.5	32	45.18	621.6	221	7200	29.5	32	7.81	781	4	189	761	47	997	6.8
35	2	4.5	3	7200	17.1	30	43.06	354.9	179	159.7	30	30	0	2.8	2	169	659	169	997	4
35	2	4.5	4	7200	14.9	33	54.74	519	143	850.5	33	33	0	136.5	3	190	696	33	919	5.2
35	2	4.5	5	7200	16.7	31	46.29	525.3	179	2366.3	31	31	0	376.1	10	185	495	72	752	4.1
40	2	4.5	1	7200	13.9	35	60.4	284.2	173	7200	33.5	35	4.29	416.8	6	227	1031	139	1397	10.3
40	2	4.5	2	7200	16.2	36	54.92	287.3	212	7200	34.3	36	4.7	574.6	13	250	685	132	1067	10.6
40	2	4.5	3	7200	16	34	52.98	243.8	196	4454.5	34	34	0	221.6	4	240	374	47	661	3.4
40	2	4.5	4	7200	17.2	35	50.95	419.6	197	5371.7	35	35	0	267.7	14	205	850	85	1140	8.9
40	2	4.5	5	7200	15	35	57.14	393.3	133	5777.2	35	35	0	240.5	4	230	1529	304	2063	16
45	2	4.5	1	7200	17.2	42	59.11	162.2	235	7200	39	42	7.14	220.7	3	286	1171	182	1639	15.9
45	2	4.5	2	7200	18.3	42	56.48	194.8	243	7200	39.6	42	5.82	95.4	0	287	1477	354	2118	21.4
45	2	4.5	3	7200	17	42	59.52	171	253	7200	39.5	42	5.95	219	3	304	1248	171	1723	28.8
45	2	4.5	4	7200	20.5	45	54.46	213.4	260	7200	43	45	4.44	52.8	6	264	2199	757	3220	43.9
45	2	4.5	5	7200	15.3	39	60.82	136.2	207	7200	36.3	39	6.84	68.4	0	336	2313	254	2903	26.4
50	2	4.5	1	7200	12.9	47	72.64	54.5	203	7200	44	47	6.38	56.7	2	389	3198	585	4172	67.4
50	2	4.5	2	7200	15.4	44	64.94	86.9	221	7200	41.2	44	6.47	77.1	4	415	2075	327	2817	48.1

Continued on Next Page...

Table 5.6 – Continued

Instances				CPLEX						Branch-and-Cut										
m	C	ρ	i	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	cuts$_{SC}$	cuts$_{KT}$	cuts$_G$	cuts$_{tot}$	time$_{cut}$
50	2	4.5	3	7200	16.4	47	65.01	114.7	239	7200	44.5	47	5.32	145.4	2	413	1489	244	2146	37.3
50	2	4.5	4	7200	24.4	49	50.11	46.8	1123	7200	44.5	48	7.29	38.2	0	390	3162	513	4065	52.4
50	2	4.5	5	7200	15.6	48	67.49	91.7	227	7200	42.5	47	9.57	19.9	4	390	3927	486	4803	62.7
55	2	4.5	1	7200	13.1	49	73.34	27.7	201	7200	46	49	6.12	50.3	1	548	1754	223	2525	41.9
55	2	4.5	2	7200	14.1	51	72.41	29.7	311	7200	47	51	7.84	48.9	1	463	2319	538	3320	72.1
55	2	4.5	3	7200	15.2	51	70.29	57.3	290	7200	46	51	9.8	34	3	434	2109	331	2874	42.1
55	2	4.5	4	7200	14	52	73.17	36.9	211	7200	47	53	11.32	8	0	464	3995	792	5251	79.4
55	2	4.5	5	7200	17.2	56	69.36	52.6	255	7200	51	56	8.93	20.3	1	377	4446	1164	5987	129.7
60	2	4.5	1	7200	13.9	59	76.37	23	291	7200	52	57	8.77	13.1	0	558	2600	148	3306	33.2
60	2	4.5	2	7200	15.4	57	72.9	17.4	314	7200	49	59	16.95	0	0	527	1842	0	2369	11.2
60	2	4.5	3	7200	16.2	58	72.07	25.4	320	7200	53	57	7.02	20.7	1	527	4276	991	5794	138.9
60	2	4.5	4	7200	15.8	59	73.26	18.1	294	7200	53.5	58	7.76	17.9	1	455	3593	804	4852	108.6
60	2	4.5	5	7200	15.4	57	72.94	28	273	7200	51	61	16.39	0	0	527	1680	0	2207	10.4
30	5	4.5	1	7200	10.2	14	27.29	1282.2	73	1.3	14	14	0	0.6	1	49	0	0	49	0
30	5	4.5	2	7200	9.7	13	25.38	1151.3	83	1	13	13	0	0.1	1	53	0	1	54	0
30	5	4.5	3	7200	10.2	13	21.23	1078.9	76	0.6	13	13	0	0.1	1	33	0	0	33	0
30	5	4.5	4	7200	9.7	14	30.55	1500.5	74	7.6	14	14	0	3.8	2	53	0	1	54	0
30	5	4.5	5	7200	9.9	13	23.66	1222.3	76	0.7	13	13	0	0.1	2	36	0	2	38	0
35	5	4.5	1	7200	10.9	18	39.58	717	99	13.3	18	18	0	2.6	4	69	6	1	76	0.1
35	5	4.5	2	7200	10.2	16	36.37	614.5	95	16.3	16	16	0	3.2	1	86	16	14	116	0.3
35	5	4.5	3	7200	10.1	17	40.78	547.3	96	27.9	17	17	0	6.6	1	67	1	2	70	0.1
35	5	4.5	4	7200	10.7	16	33.21	689.2	104	11.2	16	16	0	2.2	5	56	4	1	61	0.1
35	5	4.5	5	7200	10.9	17	36.17	687.2	96	13.4	17	17	0	2.3	1	93	3	4	100	0.2
40	5	4.5	1	7200	10.7	19	43.62	348.9	108	1983.6	19	19	0	206.9	2	103	743	78	924	3.6
40	5	4.5	2	7200	9.9	20	50.48	324.1	80	7200	18.6	20	7.17	962.8	4	110	240	77	427	3.8
40	5	4.5	3	7200	11.4	18	36.39	512.2	68	5	18	18	0	0.3	4	80	9	2	91	0.1
40	5	4.5	4	7200	10.4	20	47.85	475.7	56	444.9	20	20	0	56.4	5	89	283	23	395	1
40	5	4.5	5	7200	10.7	20	46.29	371.5	78	117.1	20	20	0	13.7	0	105	98	9	212	0.4
45	5	4.5	1	7200	12.2	25	51.33	201.9	96	7200	20.7	25	17.29	215.9	3	121	2418	482	3021	23.8
45	5	4.5	2	7200	11.6	23	49.57	275.7	52	4797.4	23	23	0	332.3	0	106	1290	190	1586	9.4
45	5	4.5	3	7200	12.3	24	48.93	149	128	7200	21.5	24	10.25	119.7	2	122	3471	892	4485	36.1
45	5	4.5	4	7200	11.6	25	53.42	286.2	49	7200	20.9	25	16.5	27.6	1	112	9082	2346	11540	82.8
45	5	4.5	5	7200	12.2	24	48.96	205.7	137	7200	21.7	24	9.43	181.6	3	93	2117	362	2572	17.2
50	5	4.5	1	7200	12.6	29	56.69	117	127	7200	23.3	30	22.2	31.8	3	153	5319	3881	9353	128.1
50	5	4.5	2	7200	13.1	30	56.47	157	96	7200	24.2	31	21.94	16	2	126	8259	5186	13571	161.2
50	5	4.5	3	7200	12.4	27	54.07	147	116	7200	22.5	30	25.11	21.6	2	136	10157	2344	12637	117.4
50	5	4.5	4	7200	12.4	26	52.44	144	121	7200	22.4	26	13.78	170.7	1	140	2655	516	3311	36.3
50	5	4.5	5	7200	12.3	29	57.51	148.4	99	7200	22.5	30	25.06	15.1	2	162	8904	4826	13892	141.9
55	5	4.5	1	7200	12.8	31	58.61	79.8	98	7200	23	32	28.12	10.7	0	199	12641	2448	15288	146.1

Continued on Next Page...

Table 5.6 – Continued

Instances				CPLEX						Branch-and-Cut										
m	C	ρ	i	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	time$_g$	LB$_g$	UB$_g$	gap$_g$	Nodes	cuts$_{CP}$	cuts$_{SC}$	cuts$_{KT}$	cuts$_G$	cuts$_{tot}$	time$_{cut}$
55	5	4.5	2	7200	13	30	56.67	92	114	7200	24.5	32	23.57	17.4	2	171	9573	3228	12972	152.5
55	5	4.5	3	7200	13	32	59.37	61	108	7200	22.2	32	30.67	13.2	1	176	13270	4544	17990	226.7
55	5	4.5	4	7200	12.8	31	58.57	102.1	66	7200	24	33	27.27	9.3	1	176	9531	4243	13950	138
55	5	4.5	5	7200	12.9	33	61.04	80	85	7200	24.3	33	26.43	15.7	0	160	7056	5648	12864	176.8
60	5	4.5	1	7200	13.8	32	56.81	57.3	125	7200	25.6	33	22.52	13.4	0	209	9986	2012	12207	124.4
60	5	4.5	2	7200	13.9	35	60.41	43.9	150	7200	26	37	29.64	9.9	0	176	8868	4767	13811	224.9
60	5	4.5	3	7200	13.9	33	57.98	48.5	172	7200	26.6	34	21.87	11.2	2	230	8036	4516	12782	211.9
60	5	4.5	4	7200	13.8	34	59.27	44.9	147	7200	27	35	22.92	7.5	1	251	10020	3115	13386	166.1
60	5	4.5	5	7200	13.8	33	58.08	48.3	154	7200	26.2	33	20.66	7.3	0	210	9388	4864	14462	185.2
30	8	4.5	1	7200	8.6	11	21.59	970.5	94	0.3	11	11	0	0.1	1	18	0	0	18	0
30	8	4.5	2	7200	9.9	11	10.33	1291.2	123	0.1	11	11	0	0	2	14	0	0	14	0
30	8	4.5	3	7200	9.1	12	24.27	1346.4	47	0.5	12	12	0	0.4	1	13	0	0	13	0
30	8	4.5	4	43.2	9	9	0	6	54	0.3	9	9	0	0	7	5	0	0	5	0
30	8	4.5	5	7200	9.1	12	23.78	1356.8	55	0.3	12	12	0	0.1	4	18	0	0	18	0
35	8	4.5	1	7200	9.4	12	21.71	732	99	0.5	12	12	0	0.2	4	15	0	0	15	0
35	8	4.5	2	7200	9.3	13	28.67	800.3	17	0.6	13	13	0	0.2	4	22	0	0	22	0
35	8	4.5	3	7200	9.5	14	32.42	803.5	54	2.5	14	14	0	1.2	8	16	0	0	16	0
35	8	4.5	4	7200	9.4	12	21.97	711.5	48	0.3	12	12	0	0	2	19	0	0	19	0
35	8	4.5	5	7200	9.2	13	29.58	842.4	74	1.5	13	13	0	0.8	3	19	0	0	19	0
40	8	4.5	1	7200	10.2	14	27.42	432.8	96	0.9	14	14	0	0.2	1	16	0	0	16	0
40	8	4.5	2	7200	9.7	17	42.74	476.4	21	21.2	17	17	0	8.9	5	26	0	0	26	0.1
40	8	4.5	3	7200	9.6	14	31.39	483.8	55	3.6	14	14	0	1.4	4	17	0	0	17	0
40	8	4.5	4	7200	9.7	14	31.06	405.7	68	1	14	14	0	0.2	4	11	0	0	11	0
40	8	4.5	5	7200	10.1	17	40.78	447.9	54	61.3	17	17	0	31.1	5	33	0	0	33	0.1
45	8	4.5	1	7200	11.6	19	38.86	290.6	60	42.9	19	19	0	11.4	2	40	21	0	61	0.2
45	8	4.5	2	7200	11.6	18	35.45	262.3	96	19.3	18	18	0	3.3	0	29	327	0	356	1.1
45	8	4.5	3	7200	11.4	18	36.56	231.7	58	84	18	18	0	20.3	4	26	122	0	148	0.5
45	8	4.5	4	7200	11.4	21	45.68	321.5	11	1848.6	21	21	0	658.6	2	30	12	0	42	1
45	8	4.5	5	7200	11.5	20	42.3	299.4	52	1968.2	20	20	0	576.5	4	26	113	0	139	1.4
50	8	4.5	1	7200	12.4	22	43.75	179.5	68	7200	18.7	22	15.11	510.3	3	33	1051	23	1107	8.4
50	8	4.5	2	7200	12.6	22	42.87	210.4	40	7200	19.6	22	10.73	287.7	7	42	1876	134	2052	11.9
50	8	4.5	3	7200	12.4	22	43.71	208	50	7200	20	22	8.93	270.9	5	27	3007	136	3170	18.8
50	8	4.5	4	7200	12.5	22	43.3	179.7	68	5383.1	22	22	0	863.8	5	33	276	0	309	3.4
50	8	4.5	5	7200	12.5	23	45.52	168.3	58	7200	20.9	23	9.11	370.9	9	43	1597	5	1645	8.6
55	8	4.5	1	7200	13	24	45.83	130.5	20	7200	21.1	24	11.9	22.2	1	36	14472	517	15025	91.3
55	8	4.5	2	7200	13	23	43.48	119.1	54	7200	20	23	13.06	50.2	3	45	7475	249	7769	42.1
55	8	4.5	3	7200	13	26	50	112.1	41	7200	22	27	18.66	136.2	3	35	3676	182	3893	35.8
55	8	4.5	4	7200	12.9	23	44.08	123	55	7200	19.1	23	16.96	474.9	2	21	1282	28	1331	12.4
55	8	4.5	5	7200	12.9	25	48.24	124	32	7200	20.4	25	18.3	104.9	5	48	6374	71	6493	46

Continued on Next Page...

Table 5.6 – Continued

Instances				CPLEX						Branch-and-Cut										
m	C	ρ	i	$\texttt{time}_g$	$\texttt{LB}_g$	$\texttt{UB}_g$	$\texttt{gap}_g$	Nodes	$\texttt{cuts}_{CP}$	$\texttt{time}_g$	$\texttt{LB}_g$	$\texttt{UB}_g$	$\texttt{gap}_g$	Nodes	$\texttt{cuts}_{CP}$	$\texttt{cuts}_{SC}$	$\texttt{cuts}_{KT}$	$\texttt{cuts}_G$	$\texttt{cuts}_{tot}$	$\texttt{time}_{cut}$
60	8	4.5	1	7200	13.8	27	49.04	69.8	61	7200	22	28	21.36	14.3	0	57	17065	1312	18434	121.5
60	8	4.5	2	7200	13.8	27	48.97	67.1	20	7200	22.1	29	23.69	10.4	1	52	17930	723	18705	118.5
60	8	4.5	3	7200	13.8	29	52.41	64.1	49	7200	23.1	29	20.18	28	1	61	9043	620	9724	86.9
60	8	4.5	4	7200	13.8	26	47.12	81.2	74	7200	21.8	26	16.2	35.2	4	37	9785	348	10170	61.6
60	8	4.5	5	7200	13.8	28	50.74	63.2	67	7200	21	30	30.15	7.9	0	52	22753	1456	24261	167.2

The results achieved with the Branch-and-Cut framework clearly outperform the ones obtained by CPLEX. Indeed, while CPLEX could only solve 7 out of 315 instances to optimality within the time limit, our developed framework could solve up to 129 instances to optimality. All instances solved by CPLEX could also be solved within the Branch-and-Cut framework. Considering only the 7 instances that were solved to optimality in both scenarios, the average CPU time required for CPLEX was 1604 seconds ($\approx$ 26 minutes), while Branch-and-Cut needed in average less than 1 second (precisely, 0.8 seconds). Moreover, the average number of nodes explored to prove optimality went down from 290,728 to 71. It is worth noting that 3 of these 7 instances did not required any branching (*i.e.*, were solved to optimality in the root node) once the cuts were added to the formulation. In addition, 6 more instances that could not be solved by CPLEX within the time limit were solved by the Branch-and-Cut algorithm in the root node (*e.g.* instances 35_8_1.5_2, 30_8_3.0_4, 30_2_4.5_4, 30_2_4.5_5, 30_8_4.5_2 and 35_8_4.5_4).

Even if the number of explored nodes required to prove optimality is greatly reduced in the Branch-and-Cut framework, the time spent within each LP node is considerably increased. Despite the fact that the LP size (*i.e.*, the number of variables and constraints) is significantly reduced by the elimination of variables and the integration of symmetry-breaking methods described in Sections 5.2 and 5.1, such behaviour is not a surprise since many violated cuts are identified and introduced to the formulation, which requires the re-optimization of the node LP. Such fact can be spotted by analyzing the difference between the number of nodes explored within the two hours time limit in each approach for the instances that could not be solved to optimality. While CPLEX was able to explore, in average, 138,931 nodes, this number is reduced to 122,147 for the Branch-and-Cut algorithm. Notice that for some unsolved instances (*e.g.*, 55_2_3.0_1), the optimization process was entirely performed in the root node. In other words, the Branch-and-Cut algorithm spent the totality of the two hours time limit searching for violated cuts and re-optimizing the root LP with such new cuts.

Such behaviour reveals a trade-off between quickly solving LP nodes and hence further exploring the enumeration tree, or instead, refining the LP node formulation through the continuous search for violated cuts. We estimate to have found a good balance between these two aspects. In fact, for the 186 instances that could not be solved within the time limit by any of the two approaches, the average lower bound obtained by CPLEX was 21.99 compared with 36.94 for the Branch-and-Cut. Moreover, the average remaining gap went down from 47.67% to 13.66%.

It is important to highlight that even if the CPLEX automatic cut generation was not disabled in the Branch-and-Cut framework, very few cuts were introduced

through this feature, when compared with standard CPLEX. In average, 106 cuts were identified and introduced using CPLEX, while only 1.9 cuts were provided by CPLEX in the Branch-and-Cut algorithm. This indicates that the vast majority of the cuts automatically added by CPLEX might be, in fact, dominated by the ones added through our separation routine. Such presumption is enhanced by the fact that the heuristic procedures presented in Section 5.4 proved to be quite effective in the search for violated cuts. Indeed, the number of k-cardinality tree inequalities and girth inequalities added during the optimization process was, in average, of 2933.1 and 694.9 cuts, respectively. The total number of user cuts added was, in average, of 3772.8 cuts.

Conclusion

> "We can only see a short distance ahead, but we can see plenty there that needs to be done."
>
> — Alan Turing

In this dissertation, we have explored several aspects of a combinatorial problem emerging from the management of a Dial-a-Ride system involving a fleet of electric autonomous vehicles. We first presented the so-called Unit Stop Number Problem (U-SNP), a variant of the Stop Number Problem introduced by Pimenta et al. [151]. In addition, a variant of particular interest, called Intersection U-SNP, is also presented. A literature review on Dial-a-Ride problems was then provided and other problems arising from telecommunications were shown to have close relations with U-SNP.

We approached the U-SNP from three main complementary angles: theoretical complexity, polyhedra and computational test.

Firstly, we investigated the some theoretical properties of the problem. Such study yielded new bounds on the optimal solution value of the U-SNP as well as allowed to understand how optimality may be lost by adopting some "intuitive" greedy policies. Polynomial-time combinatorial algorithms were then proposed to solve some specific classes of instances (*e.g.*, U-SNP with $C = 1$, Intersection U-SNP with $C = 2$). On the other hand, the $\mathcal{NP}$-Hardness of U-SNP was proved for any given fixed capacity $C \geq 2$, answering the conjecture proposed by Pimenta et al. [151]. Moreover, Intersection U-SNP was showed to be also $\mathcal{NP}$-Hard for any fixed capacity $C \geq 5$ and for $C = 3$. We conjecture that the problem remains $\mathcal{NP}$-Hard for $C = 4$. It is worth noting that our proofs of complexity applies even when the associated graph is restricted to the class of planar bipartite graphs. Such property was then used to extend our results to other related problems (*e.g.*, Traffic Grooming) improving their current state-of-the-art complexity classification.

Once U-SNP was proved to be $\mathcal{NP}$-Hard, the idea of searching for a polynomial-time algorithm capable of optimally solving the problem had to be abandoned.

Nevertheless, the need for exact methods remained. A MILP formulation adapted from Pimenta et al. [151] was therefore studied and quickly showed to be particularly weak. A polyhedral study was then conducted in order to better understand how such formulation could be improved. The dimension of the integer polyhedron is then provided as well as the necessary and sufficient conditions under which the inequalities composing the formulation are facet-defining. Families of valid inequalities (*e.g.*, strong capacity inequalities, k-cardinality tree inequalities, generalized k-cardinality tree inequalities, girth inequalities) capable of reinforcing the formulation were finally introduced. Necessary and sufficient facet-defining conditions for some of these families were presented.

Finally, the introduced valid inequalities were put to test within a Branch-and-Cut framework. The separation problems associated with each family of valid inequalities were investigated and when $\mathcal{NP}$-hardness was detected, heuristic procedures were developed to efficiently search for violated cuts. In order to boost computational performances, symmetry-breaking methods known in the literature were studied and adapted to the U-SNP. A comparison between such methods was carried out and the Orbitopal Fixing algorithm due to Kaibel et al. [101] was chosen to be the one implemented in our Branch-and-Cut algorithm. Furthermore, two relaxations preserving optimality have been proposed and compared. Our tests allow to conclude that the choice of the most suited relaxation to be applied depends on the density of the associated graph. A new upper bound on the number of vehicles needed to achieve an optimal solution is also proposed, leading to significant variable eliminations.

Computational results have proved that our approach largely outperforms CPLEX traditional Branch-and-Cut algorithm. The remaining gap of all tested instances was reduced when applying our method. Moreover, a considerable set of instances that could not be solved to optimality within the time limit of two hours by CPLEX, could be solved using our developed Branch-and-Cut framework.

Certainly, there remains many unexplored aspects with significant scientific potential to be studied over the U-SNP. Future research lines may involve a deeper investigation of generalized k-cardinality tree inequalities, whose necessary and sufficient conditions for being facet-defining are still missing. Moreover, efficient separation procedures for taking into account such inequalities may improve even further the performance of our Branch-and-Cut algorithm. Considering such inequalities and based on empirical tests with `PORTA` software over small instances, we believe that the characterization of the integer polyhedron for the polynomial case Intersection U-SNP with $C = 2$ is not far from being achieved. We also believe that an extended formulation with tighter linear relaxations might be the answer to escape

from the need of so many additional cuts in order to reinforce the formulation. For the complexity aspect, the answer to our conjecture stating that the Intersection U-SNP is NP-Hard for capacity $C = 4$ (even when G is restricted to the class of planar bipartite graphs) remains an open question that deserves attention.

The valid inequalities introduced by our study can be easily generalized for the general SNP where demands are allowed to reserve multiple seats within a single vehicle. Considering the impact such inequalities had on the performance of our Branch-and-Cut framework, computational tests over the general SNP (and other related problems such as k-Edge Partitioning and Traffic Grooming) seems appropriate. An even more rewarding challenge would be trying to adapt such inequalities for the general Dial-a-Ride Problem.

Appendix A

Detailed computational results

All tests were run on a machine provided with Intel Xeon E5, 3.10GHz and 32 Gb of RAM using CPLEX 12.8. A set of different scenarios based on the number of demands m, the capacity C and the density ρ is thus provided with $m = \{30, 35, 40, 45, 50, 55, 60\}$, $C = \{2, 5, 8\}$ and $\rho = \{1.5, 3.0, 4.5\}$. For each combination of parameters, 5 instances were randomly generated.

Table A.1 shows detailed computational results based on formulation $\text{USNP}_{(V,E,C)}$ using CPLEX with all default settings applied. No callback function is implemented. The number of variables and constraints are given for each instance under columns *cols* and *rows*, respectively. The minimum number of vehicles needed is provided under column p_{min}, and the number of vehicles used on the best solution found under column p'. Columns $time_r$, LB_r, UB_r and gap_r provide information about the performance on the root node by showing, respectively, the time spent in seconds, the lower and upper bound obtained, and the remaining gap at root node. Information on the global optimization process is depicted under column *Global*. Columns $time_g$ and *timeBest* provide the total time in seconds spent in the optimization process and the time needed for detecting the best solution found, respectively. Columns LB_r and UB_r present the best lower and upper bounds obtained by the end of the optimization process and column gap_r shows the associated remaining gap. The number of nodes explored and unexplored, by the end of the optimization, in the enumeration tree are given under columns *Nodes* and *NLeft*, respectively. Finally *CCuts* presents the number of CPLEX cuts added to the formulation throughout the optimization process.

Table A.2 provides detailed computational results in order to compare symmetry-breaking methods. All tests were run using CPLEX with default settings and an implemented (possibly empty) callback function. Column `Default` stands for the results obtained with formulation $\text{USNP}_{(V,E,C)}$. Column *Constraints* stands for the

results obtained with formulation $\text{USNP}_{(V,E,C)}$ reinforced by the symmetry-breaking constraints described in Section 5.1. Column *Orbitopal* stands for the results obtained with formulation $\text{USNP}_{(V,E,C)}$ using Orbitopal Fixing as the symmetry-breaking method. The information provided follows the structure described for Table A.1.

Table A.3 provides detailed computational results in order to show the impact of setting the number of available vehicles p to the new bound provided by Theorem 5.1. All tests were run using CPLEX with default settings and an implemented (possibly empty) callback function. Column *Default* stands for the case where p is set to m. Column `Improved Bound` stands for the results obtained with formulation $\text{USNP}_{(V,E,C)}$ when p is set to the new upper bound $\left\lceil \frac{m}{\lfloor \frac{C}{2} \rfloor + 1} \right\rceil$. Column `Minimum` stands for the results obtained when p is set to p_{min}. The information provided follows the structure described for Table 5.2 in Section 5.2.

Table A.4 provides detailed computational results of the formulations proposed in Section 4.2 in order to evaluate the impact of changing the branching policy. All tests were run using CPLEX with default settings together with the Orbitopal Fixing symmetry-breaking method. Moreover, the number of available vehicles is set to the upper bound on p_{opt} given by Theorem 5.1. Column $\texttt{USNP}_{(V,E,C)}$ stands for the results obtained with the original formulation $\text{USNP}_{(V,E,C)}$. Column $\texttt{USNP-X}_{(V,E,C)}$ stands for the results obtained from formulation $\text{USNP}_{(V,E,C)}$ by relaxing variables y_v^i and keeping the integrality constraints on variables x_e^i. Column $\texttt{USNP-Y}_{(V,E,C)}$ stands for the results obtained from formulation $\text{USNP}_{(V,E,C)}$ by setting a priority of branching on variables y_v^i. For each case, the number of binary variables in the formulation is displayed under column `bin`. The total time in seconds employed in the optimization process is given under `CPU`. The best lower bound and the best solution found within the time limit is given under `lb` and `ub`, respectively. The remaining gap associated with such bounds is then proposed under column `gap`. Since the variable choice for branching impacts the construction of the enumeration tree, the number of nodes explored in such tree is displayed under column `node`. Finally, this difference in the construction might impact the CPLEX heuristics for generating cuts. The number of CPLEX cuts added through the optimization is provided under `cuts`.

Table A.1: Detailed results of formulation $\text{USNP}_{(V,E,C)}$

Instances								*Root*				*Global*							
m	C	ρ	i	*cols*	*rows*	p_{min}	p'	$time_r$	LB_r	UB_r	gap_r	$time_g$	*timeBest*	LB_g	UB_g	gap_g	*Nodes*	*NLeft*	*CCuts*
30	2	1.5	1	1500	2130	10	11	1.3	21.9	39	43.96	42.4	31.1	35	35	0	4.2	0	24
30	2	1.5	2	1500	2220	8	8	1	22.8	35	34.95	74.2	2.7	34	34	0	5.7	0	47
30	2	1.5	3	1500	2130	9	9	1.2	21.7	44	50.71	571	324.2	36	36	0	62.8	0	35
30	2	1.5	4	1500	2190	8	8	1.1	22.6	45	49.76	41.7	35.4	34	34	0	7.7	0	19
30	2	1.5	5	1500	2220	9	10	1.7	23.7	45	47.38	115.4	23.1	36	36	0	19.6	0	37
30	2	3.0	1	1200	2040	10	10	1.3	13.8	40	65.39	184.9	21.8	31	31	0	23.2	0	45
30	2	3.0	2	1200	2040	10	10	1.3	13.9	38	63.49	66	55.6	29	29	0	9.9	0	42
30	2	3.0	3	1200	2070	10	10	1.1	14	41	65.85	445.5	16.9	31	31	0	96.4	0	25
30	2	3.0	4	1200	2070	8	8	1.4	13.9	39	64.44	74.4	67.9	27	27	0	7.2	0	28
30	2	3.0	5	1200	2010	10	10	1.6	13.7	39	64.8	256	105	30	30	0	26.1	0	44
30	2	4.5	1	1080	1980	7	7	1.3	10	35	71.43	24	21.3	24	24	0	2	0	46
30	2	4.5	2	1080	1980	9	9	1.2	10	32	68.75	58.4	25.8	26	26	0	8.6	0	34
30	2	4.5	3	1080	1980	9	9	1.2	10	28	64.29	19.8	14.7	26	26	0	1.9	0	32
30	2	4.5	4	1080	1980	9	9	1.3	9.8	29	66.08	25.5	4.2	27	27	0	3.5	0	36
30	2	4.5	5	1080	1980	10	10	1.7	9.9	33	70.05	39.4	35.8	27	27	0	3.6	0	37
35	2	1.5	1	2030	2905	10	10	1.7	24.7	56	55.84	1947.8	243.3	41	41	0	200.8	0	85
35	2	1.5	2	2030	2940	13	13	1.6	25.8	58	55.56	614.8	37.3	44	44	0	50.8	0	41
35	2	1.5	3	2030	2940	11	12	2.1	25.8	46	43.9	157.1	146	40	40	0	11.4	0	13
35	2	1.5	4	2030	2975	11	11	1.9	25.8	53	51.38	783.7	153.1	41	41	0	76.6	0	11
35	2	1.5	5	2030	2905	12	12	1.7	23.8	51	53.32	288.2	136.9	40	40	0	21.9	0	11
35	2	3.0	1	1610	2695	12	12	1.6	14.8	50	70.4	2043.1	50	35	35	0	232.3	0	45
35	2	3.0	2	1610	2765	10	10	1.6	14.8	46	67.9	4509.5	4138.4	33	33	0	563.8	0	21
35	2	3.0	3	1610	2800	12	12	2.8	19.3	41	52.88	613.1	247.3	35	35	0	25.6	0	51
35	2	3.0	4	1610	2765	11	11	2	14	37	62.16	259.2	106.6	32	32	0	14.6	0	45
35	2	3.0	5	1610	2800	10	10	1.8	14.8	50	70.46	605	276.1	33	33	0	59.7	0	30
35	2	4.5	1	1470	2660	12	12	2.4	13.8	38	63.66	1923.3	60.2	32	32	0	281.7	0	26
35	2	4.5	2	1470	2695	11	11	2	11	43	74.42	663.1	80.1	32	32	0	125.9	0	43
35	2	4.5	3	1470	2695	10	10	2.1	10.9	36	69.75	153.5	131.6	30	30	0	9.8	0	45
35	2	4.5	4	1470	2660	12	12	2.4	15.8	38	58.47	696.3	52.9	33	33	0	55.6	0	83
35	2	4.5	5	1470	2695	10	10	1.5	10.8	40	73.06	293.4	24.7	31	31	0	67.6	0	39
40	2	1.5	1	2640	3800	11	13	2.3	27.8	64	56.55	7200	747.1	46	50	8.02	406.1	244.1	18
40	2	1.5	2	2640	3800	11	12	2.9	28.8	63	54.29	7200	187	45.3	48	5.69	498.7	250	16
40	2	1.5	3	2640	3640	12	14	2.6	27.9	63	55.77	1282.1	112.9	46	46	0	88.4	0	59
40	2	1.5	4	2640	3920	11	12	3	27.8	64	56.56	5450.2	699.9	46	46	0	301.1	0	23
40	2	1.5	5	2640	3680	11	13	3.8	28.9	61	52.63	3886	318.8	48	48	0	325.8	0	22
40	2	3.0	1	2120	3600	11	11	2.6	16.9	52	67.5	7200	574.6	33.3	38	12.31	249	132	22
40	2	3.0	2	2120	3640	10	10	3.7	16.7	46	63.71	7200	419.2	32.9	37	11.14	337	202.6	18

Continued on Next Page...

Table A.1 – Continued

Instances								*Root*				*Global*							
m	C	ρ	i	*cols*	*rows*	p_{min}	p'	$time_r$	LB_r	UB_r	gap_r	$time_g$	*timeBest*	LB_g	UB_g	gap_g	*Nodes*	*NLeft*	*CCuts*
40	2	3.0	3	2120	3640	12	12	2.2	16.9	55	69.35	7200	116.9	35.8	40	10.44	308	182.6	23
40	2	3.0	4	2120	3640	13	13	4.1	16.8	51	67	7200	78	36.1	40	9.69	344.6	172.2	22
40	2	3.0	5	2120	3640	11	11	2.4	16.8	54	68.97	7200	107.2	33.9	39	13.06	325.4	216.8	20
40	2	4.5	1	1920	3520	11	11	3.4	12	42	71.43	2158.7	461.4	35	35	0	147.2	0	74
40	2	4.5	2	1920	3520	11	11	3.1	11.9	50	76.22	2193	89	36	36	0	143.9	0	39
40	2	4.5	3	1920	3520	11	11	3	12	51	76.47	7200	187.4	31.2	34	8.1	549.4	291.3	28
40	2	4.5	4	1920	3520	11	11	3.2	11.9	44	72.94	2008	1186.9	35	35	0	215.1	0	29
40	2	4.5	5	1920	3520	11	11	3.5	11.9	42	71.67	7200	6	32.8	35	6.29	550.1	215.2	29
45	2	1.5	1	3375	4815	11	11	4.1	31.7	66	51.94	3717.7	708.3	49	49	0	161	0	11
45	2	1.5	2	3375	4815	13	14	3.6	31.8	71	55.17	7200	3130.3	49.8	53	6.1	300.1	161.7	18
45	2	1.5	3	3375	4770	13	17	3.8	33.8	71	52.35	7200	199.3	53.8	55	2.15	434.9	19.7	21
45	2	1.5	4	3375	4590	13	14	3.7	31.8	72	55.8	7200	2087.1	50.6	54	6.28	205.3	93	31
45	2	1.5	5	3375	4860	12	13	3.3	31.8	71	55.17	7200	237.6	49.3	53	6.89	224.6	123.1	58
45	2	3.0	1	2700	4590	13	13	3.9	18.8	54	65.12	7200	1915.1	35.8	45	20.46	171.9	122	16
45	2	3.0	2	2700	4545	15	15	3.4	19	64	70.31	7200	2987.2	39.1	47	16.84	154.1	92.3	24
45	2	3.0	3	2700	4590	15	15	4.2	18.9	62	69.6	7200	3287.6	38.2	46	16.93	166.9	102.4	21
45	2	3.0	4	2700	4545	11	12	3.7	18.8	59	68.13	7200	478.9	35.3	42	15.88	177.9	117.3	23
45	2	3.0	5	2700	4545	13	14	3.3	18.8	59	68.09	7200	2490	34.2	45	24.01	163.3	116.5	13
45	2	4.5	1	2475	4455	14	14	4.3	14	58	75.86	7200	677	39.2	42	6.65	293.6	129.8	25
45	2	4.5	2	2475	4455	15	15	3.2	13.9	55	74.68	7200	336.4	37	42	11.91	209.6	116.8	30
45	2	4.5	3	2475	4500	13	15	4.1	14	60	76.67	7200	126.3	36.4	42	13.29	211.2	128	38
45	2	4.5	4	2475	4500	17	17	6.7	21.9	55	60.26	7200	6650.3	41.7	45	7.39	200.3	69.7	142
45	2	4.5	5	2475	4410	12	12	3.3	13.9	57	75.61	7200	1352.5	33.9	39	13.08	225.5	143.1	20
50	2	1.5	1	4150	5900	16	18	5.9	36.9	78	52.74	7200	4440.6	53.6	63	14.88	104.4	59.3	57
50	2	1.5	2	4150	6000	14	14	5.7	35.8	85	57.85	7200	880.4	54	59	8.42	76.3	31.9	56
50	2	1.5	3	4150	6150	12	14	6.5	33.8	73	53.66	7200	3171.5	48.8	58	15.89	116.8	75.4	54
50	2	1.5	4	4150	5950	15	15	5.7	34.9	75	53.52	7200	1290.6	51	60	15	83.5	29.5	62
50	2	1.5	5	4150	6050	14	15	6.6	34.4	79	56.51	7200	1305.9	49.1	60	18.11	76.9	32	67
50	2	3.0	1	3300	5700	15	16	4.8	20	75	73.33	7200	700.9	39.2	51	23.16	97.4	64.7	34
50	2	3.0	2	3300	5500	19	20	5.3	19.9	76	73.83	7200	551.1	39.4	55	28.38	84.2	53.6	57
50	2	3.0	3	3300	5600	14	14	5.1	19.8	66	69.94	7200	1563.9	38	50	24.03	89.8	12.7	29
50	2	3.0	4	3300	5750	15	16	5.5	20	66	69.7	7200	531.3	39.5	49	19.3	71.4	38	56
50	2	3.0	5	3300	5750	14	15	5	19.8	70	71.75	7200	1781.6	36.7	52	29.52	77.5	34.5	49
50	2	4.5	1	3050	5400	14	15	4.6	14.8	62	76.05	7200	2933.5	32.4	47	31.05	121.4	80.3	8
50	2	4.5	2	3050	5450	13	13	4.8	14.9	61	75.55	7200	980.6	36.7	44	16.66	88.6	34.5	48
50	2	4.5	3	3050	5450	15	15	5.4	14.9	65	77.03	7200	3951.1	35.5	47	24.38	105.1	50.4	27
50	2	4.5	4	3050	5400	17	17	5.9	17.9	61	70.6	7200	288.2	37.1	49	24.19	115.2	74.6	30

Continued on Next Page...

Table A.1 – Continued

Instances								*Root*				*Global*							
m	C	ρ	i	*cols*	*rows*	p_{min}	p'	$time_r$	LB_r	UB_r	gap_r	$time_g$	*timeBest*	LB_g	UB_g	gap_g	*Nodes*	*NLeft*	*CCuts*
50	2	4.5	5	3050	5450	15	15	4.7	14.9	64	76.67	7200	3300.6	34.9	48	27.29	88.7	54	64
55	2	1.5	1	5005	7205	14	16	6.7	38.9	91	57.3	7200	2660.7	53	64	17.17	67.1	38.8	55
55	2	1.5	2	5005	7095	17	18	11.2	37.9	79	52.08	7200	3870.4	53.8	66	18.54	83.4	47.2	37
55	2	1.5	3	5005	7315	16	16	31.5	38.8	83	53.22	7200	6605.9	53.8	62	13.18	55.8	22.9	66
55	2	1.5	4	5005	7425	14	14	6.8	37.8	85	55.48	7200	1825.6	52.3	64	18.26	53.9	19.2	56
55	2	1.5	5	5005	7095	16	18	29.4	35	89	60.67	7200	2271.6	50.5	66	23.44	62.6	53	57
55	2	3.0	1	4015	6875	16	16	7	21.9	80	72.64	7200	833.9	39.7	58	31.47	64.7	40.4	43
55	2	3.0	2	4015	6655	16	16	6	21.9	73	70.07	7200	946.4	37	56	33.86	69.8	28	64
55	2	3.0	3	4015	6710	18	18	7.5	21.9	78	71.95	7200	6179.9	38.4	58	33.84	75	43.8	25
55	2	3.0	4	4015	6765	15	16	7	21.9	80	72.69	7200	6036.9	36.5	55	33.62	49.5	23.6	39
55	2	3.0	5	4015	6930	15	16	6.4	21.9	77	71.62	7200	3557.8	38.7	56	30.91	66.7	54.1	24
55	2	4.5	1	3685	6600	15	15	6.5	15.9	74	78.58	7200	2104.4	32.4	49	33.88	69.6	40.8	23
55	2	4.5	2	3685	6710	15	15	6.6	15.9	78	79.58	7200	7005.9	38.2	50	23.52	49.8	6.4	72
55	2	4.5	3	3685	6710	16	16	6.7	15.9	73	78.26	7200	6130.2	39.3	51	22.96	46.8	19	65
55	2	4.5	4	3685	6600	17	18	6.9	15.9	77	79.41	7200	2882	34.6	53	34.68	73.5	57.3	21
55	2	4.5	5	3685	6600	20	20	5.3	15.9	80	80.13	7200	759	32.8	56	41.5	84	56	6
60	2	1.5	1	6000	8280	19	20	9	41.9	104	59.67	7200	3432.1	60.5	73	17.14	38.1	28.1	18
60	2	1.5	2	6000	8640	19	20	31.5	41.9	93	54.9	7200	7000.7	57.8	73	20.86	33.8	9.8	34
60	2	1.5	3	6000	8580	17	18	26.2	40.9	94	56.52	7200	6652.1	55.2	71	22.26	45.4	19.4	34
60	2	1.5	4	6000	8700	16	18	27.7	42	89	52.85	7200	496.1	58.8	72	18.27	35.9	12.5	75
60	2	1.5	5	6000	8700	16	17	53.1	42.9	99	56.7	7200	6762.3	56.5	71	20.45	45.1	30.9	35
60	2	3.0	1	4800	8100	18	18	21.2	23.9	78	69.33	7200	5909.6	40.7	65	37.38	35.3	29	88
60	2	3.0	2	4800	8160	17	19	9.1	24	90	73.33	7200	6589.8	42.9	63	31.85	43.2	19	44
60	2	3.0	3	4800	8100	20	20	17.4	23.9	96	75.05	7200	6792.3	41.2	64	35.67	43.7	19.8	49
60	2	3.0	4	4800	8160	18	18	10	23.9	86	72.16	7200	6443.2	40.8	63	35.2	31	23.6	99
60	2	3.0	5	4800	8340	16	19	9.8	23.9	91	73.7	7200	7048	44.7	64	30.1	46.2	33.7	85
60	2	4.5	1	4380	7860	16	19	8.8	16.8	85	80.22	7200	4722.5	31.4	58	45.81	50.2	25.3	27
60	2	4.5	2	4380	7920	17	17	9	16.9	81	79.09	7200	532.3	37.1	57	34.84	46.7	22.9	51
60	2	4.5	3	4380	7860	19	19	13.3	17	83	79.52	7200	1138.4	37.7	57	33.95	39.3	15.3	81
60	2	4.5	4	4380	7860	19	20	8.5	16.9	82	79.33	7200	1581.2	35.5	59	39.87	41.2	17.3	81
60	2	4.5	5	4380	7800	18	18	9.2	17	85	80	7200	3005.2	34.8	57	38.89	42.3	18.3	61
30	5	1.5	1	1500	2190	4	4	2.2	19.5	27	27.74	15.8	3.3	25	25	0	1.4	0	36
30	5	1.5	2	1500	2250	4	4	2	21.4	27	20.62	4.1	2	27	27	0	0.1	0	8
30	5	1.5	3	1500	2100	5	5	1.7	22.4	30	25.4	6.5	6.2	27	27	0	0.5	0	9
30	5	1.5	4	1500	2130	4	4	1.7	21.2	28	24.38	9.9	5.7	27	27	0	0.9	0	6
30	5	1.5	5	1500	2160	4	4	1.9	22.2	29	23.55	17.4	8.9	28	28	0	1.6	0	8
30	5	3.0	1	1200	2040	4	4	1.8	12.5	20	37.54	20.7	6.1	19	19	0	1.9	0	15

Continued on Next Page...

Table A.1 – Continued

Instances								*Root*				*Global*							
m	C	ρ	i	*cols*	*rows*	p_{min}	p'	$time_r$	LB_r	UB_r	gap_r	$time_g$	*timeBest*	LB_g	UB_g	gap_g	*Nodes*	*NLeft*	*CCuts*
30	5	3.0	2	1200	2070	4	4	1.7	12.4	20	38.19	23.8	12.3	19	19	0	2	0	20
30	5	3.0	3	1200	2070	4	4	1.7	12.9	21	38.44	13.2	13.2	18	18	0	0.9	0	25
30	5	3.0	4	1200	2040	4	4	1.9	12.5	19	34.34	6.6	4.3	18	18	0	0.4	0	12
30	5	3.0	5	1200	2010	4	4	1.9	12.3	20	38.42	16.6	9.6	19	19	0	1.4	0	10
30	5	4.5	1	1080	1950	4	4	1.9	9.4	16	40.97	4	3.1	14	14	0	0.2	0	15
30	5	4.5	2	1080	1980	4	4	1.4	9.3	14	33.79	2.4	2.1	13	13	0	0	0	24
30	5	4.5	3	1080	1980	4	4	1.6	8.2	14	41.38	6.1	6.1	13	13	0	0.3	0	21
30	5	4.5	4	1080	1980	4	4	1.9	9	14	35.71	8.9	1.9	14	14	0	0.7	0	30
30	5	4.5	5	1080	1980	3	3	1.8	8.1	13	37.49	4.4	1.8	13	13	0	0.2	0	18
35	5	1.5	1	2030	2905	5	5	3.6	23.4	34	31.05	111.3	67.4	31	31	0	3.9	0	20
35	5	1.5	2	2030	2940	4	4	3.6	24.5	32	23.44	17	12.4	31	31	0	0.8	0	10
35	5	1.5	3	2030	3010	4	4	3.5	23.5	32	26.65	25.6	19.5	29	29	0	1.2	0	8
35	5	1.5	4	2030	2905	5	5	2.7	25.4	37	31.4	117.2	116	33	33	0	7	0	19
35	5	1.5	5	2030	3010	5	5	3.3	24.3	38	36.05	63.1	60.7	31	31	0	3.9	0	14
35	5	3.0	1	1610	2765	4	4	2.3	13.4	23	41.93	225	63.5	22	22	0	12	0	22
35	5	3.0	2	1610	2730	5	5	3.3	13.4	21	36.28	94.9	3.3	21	21	0	5.5	0	19
35	5	3.0	3	1610	2765	5	5	2.8	13.5	30	55.05	538.8	251.3	23	23	0	35.5	0	24
35	5	3.0	4	1610	2765	5	5	2.8	13.4	26	48.42	360.3	351.3	22	22	0	20.4	0	19
35	5	3.0	5	1610	2765	4	4	2.3	13.4	23	41.86	161.8	14.1	21	21	0	5.8	0	31
35	5	4.5	1	1470	2695	6	6	2.1	10.3	21	50.92	22.6	16.7	18	18	0	1.1	0	62
35	5	4.5	2	1470	2695	4	4	2.2	9.5	16	40.47	12.4	2.2	16	16	0	0.5	0	32
35	5	4.5	3	1470	2695	4	4	2.6	9.5	17	44.27	45.3	2.6	17	17	0	2.5	0	70
35	5	4.5	4	1470	2695	5	5	2.4	9.4	19	50.48	13.9	8	16	16	0	0.5	0	33
35	5	4.5	5	1470	2695	5	5	2.9	9.6	23	58.18	18.9	5.7	17	17	0	0.9	0	45
40	5	1.5	1	2640	3840	4	5	5.4	28.3	49	42.27	445.8	113.6	37	37	0	23.8	0	15
40	5	1.5	2	2640	4080	4	4	5.8	27.8	39	28.85	64.9	42.9	36	36	0	2	0	5
40	5	1.5	3	2640	3840	5	5	3.3	28.3	41	30.93	376.2	362.6	37	37	0	11.8	0	18
40	5	1.5	4	2640	3880	5	5	4	27.7	43	35.59	312.3	18.9	37	37	0	13.4	0	18
40	5	1.5	5	2640	4000	4	4	4.2	27.1	36	24.63	132.8	119.1	33	33	0	2.7	0	20
40	5	3.0	1	2120	3680	5	5	3.7	15.7	29	46.02	1193.3	352	26	26	0	28.8	0	27
40	5	3.0	2	2120	3640	5	5	3.4	15.4	32	51.81	486.8	390.2	25	25	0	15.8	0	12
40	5	3.0	3	2120	3640	6	6	5	15.5	31	49.86	374.5	309.5	25	25	0	12.4	0	32
40	5	3.0	4	2120	3560	5	5	3.3	15.5	33	53.07	588.3	463	25	25	0	16.5	0	17
40	5	3.0	5	2120	3640	5	5	4.4	15.5	28	44.55	1928.1	1334.9	26	26	0	50.2	0	38
40	5	4.5	1	1920	3520	5	5	4.8	10.8	24	55.2	218.2	146.9	19	19	0	6.1	0	27
40	5	4.5	2	1920	3480	5	5	4.1	10.9	22	50.66	564.5	89.3	20	20	0	19.9	0	31
40	5	4.5	3	1920	3480	4	4	3	10.3	22	53.26	49.1	23.8	18	18	0	1.8	0	29

Continued on Next Page...

Table A.1 – Continued

Instances								*Root*				*Global*							
m	C	ρ	i	*cols*	*rows*	p_{min}	p'	$time_r$	LB_r	UB_r	gap_r	$time_g$	*timeBest*	LB_g	UB_g	gap_g	*Nodes*	*NLeft*	*CCuts*
40	5	4.5	4	1920	3440	6	6	2.7	11.4	27	57.93	162.4	71.1	20	20	0	7.2	0	19
40	5	4.5	5	1920	3520	5	5	3.2	10.9	22	50.5	438.8	57.8	20	20	0	13.9	0	26
45	5	1.5	1	3375	4860	4	5	6	30.6	42	27.19	711	323.5	40	40	0	15.3	0	14
45	5	1.5	2	3375	4590	7	7	4.6	32.5	49	33.65	402.6	89.6	41	41	0	11	0	30
45	5	1.5	3	3375	4770	6	6	5.2	30.5	47	35.15	1391	685.4	41	41	0	29.3	0	22
45	5	1.5	4	3375	4815	6	7	4	32.5	49	33.71	938.3	787.3	42	42	0	26.7	0	20
45	5	1.5	5	3375	4725	7	7	4.2	29.7	45	33.91	2513.8	285.5	43	43	0	71.2	0	38
45	5	3.0	1	2700	4545	6	6	4.2	17.5	47	62.87	7200	1069.8	27	31	12.8	111.3	49.6	22
45	5	3.0	2	2700	4545	6	6	3.7	17.5	50	64.96	7200	287.8	27.9	31	10	101	44	46
45	5	3.0	3	2700	4545	6	6	4.5	17.5	53	66.98	7200	287.7	27.9	30	7.1	152.9	55.4	40
45	5	3.0	4	2700	4545	6	6	4.9	17.5	38	53.85	7200	864.8	27.1	31	12.66	115.7	52.4	34
45	5	3.0	5	2700	4500	6	6	5	17.6	39	55	7200	563.4	27.9	31	10	137.9	69.7	43
45	5	4.5	1	2475	4410	6	6	4.3	12.6	34	62.94	7200	1854.9	23.1	25	7.53	176.6	51.2	35
45	5	4.5	2	2475	4365	5	5	4.2	13.3	34	60.87	1148.3	339.1	23	23	0	31.9	0	14
45	5	4.5	3	2475	4500	6	6	4.2	12.5	32	60.88	5081.8	1271.6	24	24	0	108.3	0	22
45	5	4.5	4	2475	4455	6	6	4.2	12.5	36	65.38	7200	314.7	23.6	25	5.42	210.1	27	17
45	5	4.5	5	2475	4500	6	6	4.1	13.5	30	54.95	1052.9	230.4	24	24	0	22.7	0	48
50	5	1.5	1	4150	5850	6	7	5.8	34.5	52	33.59	4903.2	1211.1	46	46	0	126.3	0	30
50	5	1.5	2	4150	5900	5	5	8.5	35.4	54	34.41	7200	3375	43.8	46	4.68	106.4	27.3	27
50	5	1.5	3	4150	6100	5	6	10.9	34.7	50	30.67	1917.8	301.9	46	46	0	42.5	0	25
50	5	1.5	4	4150	5800	7	7	8.3	33.6	63	46.68	7200	4040.8	44.1	48	8.13	112.7	57.3	26
50	5	1.5	5	4150	6050	6	6	7.3	33.9	62	45.3	7200	322.7	44.8	48	6.77	78.5	33	37
50	5	3.0	1	3300	5550	6	6	5.8	18.5	40	53.66	7200	2520.8	28.3	31	8.71	96.8	42.2	17
50	5	3.0	2	3300	5650	7	7	6.4	18.6	61	69.52	7200	4894.8	28.7	35	18.02	66.4	30	21
50	5	3.0	3	3300	5750	6	6	5.7	18.5	56	66.96	7200	5889.2	27.8	34	18.3	74.6	39	33
50	5	3.0	4	3300	5650	7	7	6.7	18.8	63	70.21	7200	3795.3	28.5	34	16.2	70.4	34.7	35
50	5	3.0	5	3300	5750	6	6	6.2	18.5	38	51.32	4148.9	740.4	30	30	0	62.8	0	18
50	5	4.5	1	3050	5500	7	7	5.5	13.9	45	69.18	7200	6715.6	23.8	30	20.55	85.2	43.2	80
50	5	4.5	2	3050	5450	8	8	5.5	13.8	51	73.01	7200	2414.4	24.9	30	16.88	101.2	57.4	45
50	5	4.5	3	3050	5500	6	6	5.6	13.8	46	69.94	7200	1246.3	23	28	17.88	77	40.8	54
50	5	4.5	4	3050	5500	6	6	5.7	13.6	46	70.52	7200	2020.5	23.1	26	10.96	69.7	30.6	16
50	5	4.5	5	3050	5400	7	7	6.1	13.6	51	73.32	7200	1989.1	24	30	20	107.2	66.5	33
55	5	1.5	1	5005	7315	7	7	67	37.6	76	50.55	7200	2140.2	47.1	51	7.7	77.7	31	33
55	5	1.5	2	5005	7260	7	8	78.4	37.7	59	36.09	7200	4784.4	47.2	51	7.44	53.6	18.5	42
55	5	1.5	3	5005	6875	8	9	9.7	38.6	73	47.08	4133.7	386.9	50	50	0	76	0	34
55	5	1.5	4	5005	7315	7	7	36.3	35.7	72	50.43	7200	6756.8	45.2	49	7.66	52	16.1	34
55	5	1.5	5	5005	7040	7	7	19.5	38.6	74	47.86	7200	1437.4	48.1	52	7.41	70.2	19.5	41

Continued on Next Page...

Table A.1 – Continued

Instances								*Root*				*Global*							
m	C	ρ	i	*cols*	*rows*	p_{min}	p'	$time_r$	LB_r	UB_r	gap_r	$time_g$	*timeBest*	LB_g	UB_g	gap_g	*Nodes*	*NLeft*	*CCuts*
55	5	3.0	1	4015	6765	7	7	9.4	20.7	65	68.13	7200	6516.5	30.1	40	24.72	58.3	30	27
55	5	3.0	2	4015	6985	6	6	8.3	20.7	50	58.68	7200	578.4	29.1	37	21.31	41.2	17.1	29
55	5	3.0	3	4015	6820	6	7	8.3	20.5	60	65.77	7200	1981.3	29	40	27.4	51.9	30.8	45
55	5	3.0	4	4015	6930	6	7	9.2	20.5	63	67.52	7200	6761.3	29	36	19.51	57.9	16.3	20
55	5	3.0	5	4015	6820	7	7	9.2	20.6	65	68.37	7200	6745.5	30.8	37	16.89	60.3	25.2	37
55	5	4.5	1	3685	6600	6	6	7.4	14.5	54	73.08	7200	109.9	22.7	32	29.12	57.1	33	19
55	5	4.5	2	3685	6600	7	7	8.2	14.9	51	70.85	7200	6858.6	24.3	30	19.1	49.1	17.7	28
55	5	4.5	3	3685	6600	7	7	6.8	14.7	59	75.04	7200	1197.1	25	32	21.74	52.2	26.5	52
55	5	4.5	4	3685	6545	7	7	7.4	14.6	56	73.96	7200	6525.8	23.7	31	23.57	70.1	23.2	42
55	5	4.5	5	3685	6545	7	7	8.3	14.6	61	76.05	7200	5071.2	24.4	33	26.09	79.5	44.3	14
60	5	1.5	1	6000	8880	5	6	90.1	38.5	74	47.91	7200	1756.4	45.7	50	8.64	33.6	9.4	20
60	5	1.5	2	6000	8640	6	7	86	40.7	65	37.34	7200	1919.3	48.8	53	7.87	37.8	12.3	50
60	5	1.5	3	6000	8340	7	7	70.8	40.6	92	55.89	7200	2054	49.7	57	12.87	47.3	21.6	28
60	5	1.5	4	6000	8640	7	7	107.2	39.7	83	52.13	7200	3026.2	47.2	56	15.72	33.2	25.6	24
60	5	1.5	5	6000	8340	8	9	86.8	40.7	83	51.01	7200	3910.7	48.1	57	15.63	36.8	25.8	46
60	5	3.0	1	4800	8340	7	7	20.8	22.6	65	65.2	7200	6825.3	30.9	44	29.8	28.5	9	70
60	5	3.0	2	4800	8100	7	8	9.4	22.6	72	68.66	7200	6630.5	30.4	42	27.68	31	10.3	41
60	5	3.0	3	4800	8220	7	8	9	22.7	72	68.41	7200	6631.1	31.3	42	25.43	33.7	12	39
60	5	3.0	4	4800	8040	7	7	9.1	22.7	67	66.09	7200	6780	31.9	39	18.25	26.2	14.8	63
60	5	3.0	5	4800	8040	7	8	9.8	22.7	65	65.13	7200	6812.5	30.9	43	28.11	33	22.4	21
60	5	4.5	1	4380	7920	6	6	79	16.6	55	69.82	7200	7146.6	24	33	27.38	17	11.8	28
60	5	4.5	2	4380	7920	7	7	9.8	15.7	62	74.69	7200	71.5	25.5	35	27.22	31.6	10.1	23
60	5	4.5	3	4380	7920	8	8	8.4	15.7	61	74.29	7200	136.3	25.3	34	25.6	30.9	8.8	99
60	5	4.5	4	4380	7920	7	7	8.7	15.6	59	73.57	7200	6763.5	23.7	35	32.27	32.5	12.1	73
60	5	4.5	5	4380	7920	6	6	8.3	16	61	73.77	7200	6928.5	24.7	35	29.31	29.8	22.1	28
30	8	1.5	1	1500	2130	3	3	1.3	21.4	24	10.94	1.7	1.3	24	24	0	0	0	5
30	8	1.5	2	1500	2130	3	3	1.8	20.4	23	11.52	2	1.8	23	23	0	0	0	6
30	8	1.5	3	1500	2130	3	3	1.9	20.5	24	14.39	2.7	1.9	24	24	0	0	0	6
30	8	1.5	4	1500	2190	3	3	2	19.6	23	14.87	2.9	2	23	23	0	0.1	0	3
30	8	1.5	5	1500	2160	3	3	2	21.4	24	10.94	2.4	2	24	24	0	0	0	7
30	8	3.0	1	1200	2070	3	3	1.7	11.9	16	25.87	3.7	1.7	16	16	0	0.1	0	13
30	8	3.0	2	1200	2070	2	2	1.1	10.8	13	17.31	1.4	1.1	13	13	0	0	0	2
30	8	3.0	3	1200	2040	2	2	0.9	10.9	14	22.36	1.3	0.9	14	14	0	0	0	2
30	8	3.0	4	1200	2010	2	2	1	11.7	15	21.75	1.9	1	15	15	0	0	0	2
30	8	3.0	5	1200	2040	3	3	2.2	11.4	15	23.96	3.1	2.2	15	15	0	0	0	5
30	8	4.5	1	1080	1980	3	3	1.3	7.9	11	28.53	2.1	1.3	11	11	0	0.1	0	12
30	8	4.5	2	1080	1980	3	3	1.9	7.7	11	30.27	3.3	1.9	11	11	0	0.1	0	14

Continued on Next Page...

Table A.1 – Continued

Instances								*Root*				*Global*							
m	C	ρ	i	*cols*	*rows*	p_{min}	p'	$time_r$	LB_r	UB_r	gap_r	$time_g$	*timeBest*	LB_g	UB_g	gap_g	*Nodes*	*NLeft*	*CCuts*
30	8	4.5	3	1080	1980	3	3	1.3	8	12	33.68	3	1.3	12	12	0	0.2	0	13
30	8	4.5	4	1080	1950	2	2	0	0	0	0	0.9	0.8	9	9	0	0	0	5
30	8	4.5	5	1080	1980	3	3	1.9	8.4	12	29.78	2.9	1.9	12	12	0	0.1	0	10
35	8	1.5	1	2030	2940	3	3	3.3	24.7	29	14.94	4.9	3.3	29	29	0	0	0	7
35	8	1.5	2	2030	3010	3	3	2.8	22.2	25	11.29	3.6	2.8	25	25	0	0	0	7
35	8	1.5	3	2030	2975	3	3	3.7	24.6	29	15.2	8.7	3.7	29	29	0	0.2	0	8
35	8	1.5	4	2030	2835	3	3	3.4	23.4	27	13.37	4.4	3.4	27	27	0	0	0	5
35	8	1.5	5	2030	2800	4	4	2.7	23.5	28	15.91	6.8	2.7	28	28	0	0.2	0	6
35	8	3.0	1	1610	2835	3	3	2.6	12.9	18	28.55	11.3	2.6	18	18	0	0.4	0	9
35	8	3.0	2	1610	2730	3	3	2.9	12.2	19	35.59	11.8	7.9	18	18	0	0.4	0	5
35	8	3.0	3	1610	2800	2	2	1.7	12.9	14	7.5	1.7	1.7	14	14	0	0	0	2
35	8	3.0	4	1610	2800	3	3	2.9	12.9	17	24.34	5.9	2.9	17	17	0	0.1	0	9
35	8	3.0	5	1610	2765	3	3	3.1	13	18	28.05	12.2	3.1	18	18	0	0.5	0	3
35	8	4.5	1	1470	2695	3	3	2.9	8.8	12	26.54	4.2	2.9	12	12	0	0	0	9
35	8	4.5	2	1470	2660	3	3	1.9	9	13	30.63	4	1.9	13	13	0	0.1	0	6
35	8	4.5	3	1470	2660	3	3	2	8.9	14	36.15	6.6	2	14	14	0	0.2	0	11
35	8	4.5	4	1470	2660	3	3	2.5	8.9	12	25.66	3.6	2.5	12	12	0	0	0	9
35	8	4.5	5	1470	2695	3	3	2.3	8.7	13	33.03	7	2.3	13	13	0	0.2	0	9
40	8	1.5	1	2640	3920	3	3	6.3	26.7	32	16.46	28	6.3	32	32	0	0.6	0	6
40	8	1.5	2	2640	3640	3	3	4.5	26.7	35	23.76	34.5	34.5	31	31	0	1.1	0	6
40	8	1.5	3	2640	3800	4	4	5.8	24.4	34	28.19	32.6	27.4	31	31	0	0.8	0	8
40	8	1.5	4	2640	3760	4	4	5.6	27.4	36	23.78	35.1	30.3	34	34	0	1	0	4
40	8	1.5	5	2640	3960	3	3	5.7	27.3	33	17.27	12.8	12.1	32	32	0	0.2	0	7
40	8	3.0	1	2120	3600	3	3	6.4	14.4	22	34.71	98.9	6.4	22	22	0	3	0	4
40	8	3.0	2	2120	3640	3	3	3.8	15.1	20	24.4	17.4	3.8	20	20	0	0.3	0	12
40	8	3.0	3	2120	3600	4	4	3.1	15.1	25	39.61	257.6	15.4	23	23	0	6.6	0	9
40	8	3.0	4	2120	3600	4	4	3.3	15.3	25	38.97	140.5	109	22	22	0	2.7	0	9
40	8	3.0	5	2120	3600	4	4	7.7	15.3	23	33.33	204.9	195	22	22	0	4.3	0	15
40	8	4.5	1	1920	3520	3	3	3.1	10.2	14	27.06	9	3.1	14	14	0	0.1	0	20
40	8	4.5	2	1920	3480	4	4	2.4	10.4	17	39.04	68.9	2.4	17	17	0	2.3	0	6
40	8	4.5	3	1920	3520	3	3	3.3	9.7	14	30.44	8.3	3.3	14	14	0	0.1	0	7
40	8	4.5	4	1920	3520	3	3	3.9	9.9	15	34.05	18.7	18.7	14	14	0	0.3	0	16
40	8	4.5	5	1920	3480	4	4	3.3	10.4	18	42.48	67.2	5.3	17	17	0	2.5	0	23
45	8	1.5	1	3375	4905	3	3	5.8	31.4	42	25.32	107	107	38	38	0	2.2	0	7
45	8	1.5	2	3375	4860	4	4	6.6	31.4	39	19.44	78.5	78.5	38	38	0	1.5	0	9
45	8	1.5	3	3375	4815	4	4	9	28.4	38	25.22	101.5	54.3	36	36	0	1.9	0	7
45	8	1.5	4	3375	4950	3	3	6.3	29.3	38	22.83	59.4	10.6	36	36	0	0.7	0	4

Continued on Next Page...

Table A.1 – Continued

Instances								*Root*				*Global*							
m	C	ρ	i	*cols*	*rows*	p_{min}	p'	$time_r$	LB_r	UB_r	gap_r	$time_g$	*timeBest*	LB_g	UB_g	gap_g	*Nodes*	*NLeft*	*CCuts*
45	8	1.5	5	3375	4905	3	3	6.8	30.3	38	20.18	20.7	12.5	36	36	0	0.2	0	5
45	8	3.0	1	2700	4500	4	4	4	17.4	30	42.15	322.5	225.3	25	25	0	5.1	0	13
45	8	3.0	2	2700	4590	4	4	4.4	17.1	29	40.94	894.4	48.6	26	26	0	17	0	13
45	8	3.0	3	2700	4680	3	3	5	16.3	26	37.41	225.9	207.3	24	24	0	3.5	0	3
45	8	3.0	4	2700	4545	4	4	4.4	17.2	27	36.35	568.2	557.6	25	25	0	9.1	0	13
45	8	3.0	5	2700	4590	4	4	9.7	16.7	25	33.3	388.8	358.1	24	24	0	5.1	0	23
45	8	4.5	1	2475	4455	4	4	9	12.2	19	35.57	83.1	9	19	19	0	1.5	0	34
45	8	4.5	2	2475	4500	3	3	6.4	12	20	40.16	53.6	24.8	18	18	0	0.7	0	30
45	8	4.5	3	2475	4500	3	3	9.8	12	19	36.89	52.8	26.4	18	18	0	0.5	0	9
45	8	4.5	4	2475	4410	4	4	4.2	12.2	23	47.16	1050.9	30.3	21	21	0	26.3	0	9
45	8	4.5	5	2475	4410	4	4	3.8	12.2	21	42.02	363.5	188.3	20	20	0	6.1	0	11
50	8	1.5	1	4150	6000	4	5	16	33.8	47	28	286.1	132	41	41	0	2.1	0	20
50	8	1.5	2	4150	5800	4	4	18	31.4	41	23.51	183	183	38	38	0	1.6	0	6
50	8	1.5	3	4150	6100	4	4	13.8	31.4	44	28.62	262.6	232.6	39	39	0	2.5	0	11
50	8	1.5	4	4150	5800	4	4	7.2	32.5	47	30.75	596.8	596.8	41	41	0	6.8	0	14
50	8	1.5	5	4150	5850	5	5	16.7	33.6	48	30.08	1159.7	1158.7	42	42	0	9.5	0	15
50	8	3.0	1	3300	5550	4	4	6.5	18.2	29	37.4	6076.2	2919.1	28	28	0	59.4	0	11
50	8	3.0	2	3300	5700	4	4	6	17.8	31	42.55	3109.5	959.7	28	28	0	38	0	19
50	8	3.0	3	3300	5650	4	4	6.7	18.2	31	41.38	6300.1	2981.7	28	28	0	81.4	0	6
50	8	3.0	4	3300	5600	4	4	7.2	18.3	31	41.05	3283.6	1246.9	28	28	0	34.2	0	9
50	8	3.0	5	3300	5600	4	4	10.2	18.7	31	39.6	7200	547.9	27.3	29	5.98	88.9	19.7	12
50	8	4.5	1	3050	5550	4	4	6	13.3	24	44.55	761.6	338.3	22	22	0	7.1	0	19
50	8	4.5	2	3050	5300	5	5	5.3	13.4	24	44.28	1192.9	1109.6	22	22	0	13.6	0	8
50	8	4.5	3	3050	5500	4	4	7	13.2	27	51	1571.3	243.2	22	22	0	17.4	0	12
50	8	4.5	4	3050	5500	4	4	7.9	13.8	24	42.52	1017.8	727.3	22	22	0	8.3	0	17
50	8	4.5	5	3050	5550	5	5	5.5	13.5	26	48.07	2000.2	518.3	23	23	0	23.8	0	19
55	8	1.5	1	5005	7260	4	5	31.7	35.6	49	27.33	1478.9	1178	44	44	0	12.4	0	5
55	8	1.5	2	5005	7150	4	5	65.1	36.6	51	28.18	892.5	301.7	45	45	0	5	0	3
55	8	1.5	3	5005	7205	4	4	48.3	36.4	49	25.63	574.6	471	44	44	0	3.5	0	7
55	8	1.5	4	5005	7095	4	4	15.3	37.4	51	26.57	1675.8	249.5	46	46	0	13.8	0	12
55	8	1.5	5	5005	7205	5	5	68.4	35.6	52	31.54	7200	169	44.9	47	4.42	70.8	23.9	25
55	8	3.0	1	4015	6820	5	5	8.6	20.3	39	47.86	7200	6834.6	28.4	31	8.5	53.9	12.1	10
55	8	3.0	2	4015	6765	4	4	8.6	20.2	37	45.37	3615.9	866.8	30	30	0	20	0	12
55	8	3.0	3	4015	6820	4	4	7.7	19.5	51	61.76	5055.8	836.8	31	31	0	36.9	0	11
55	8	3.0	4	4015	6655	5	5	7.8	20.5	53	61.41	7200	516.7	27.1	33	17.76	58.4	43.2	9
55	8	3.0	5	4015	6820	5	5	7.3	20.4	41	50.36	7200	2254.2	28	34	17.59	45.3	30.7	37
55	8	4.5	1	3685	6600	4	4	9.9	14.4	33	56.46	7200	747	21.9	25	12.47	45.6	13.7	14

Continued on Next Page...

Table A.1 – Continued

Instances								*Root*				*Global*							
m	C	ρ	i	*cols*	*rows*	p_{min}	p'	$time_r$	LB_r	UB_r	gap_r	$time_g$	*timeBest*	LB_g	UB_g	gap_g	*Nodes*	*NLeft*	*CCuts*
55	8	4.5	2	3685	6710	4	4	7.5	14.1	29	51.26	4238.9	362.7	23	23	0	29.1	0	13
55	8	4.5	3	3685	6600	5	5	6.3	14.3	34	57.84	7200	1926.6	22	27	18.57	57.3	16.6	12
55	8	4.5	4	3685	6600	4	4	7.2	14.1	27	47.74	1546.7	540.8	23	23	0	10.3	0	19
55	8	4.5	5	3685	6710	5	5	6.9	14.3	43	66.7	7200	405.5	22.8	25	8.76	70.1	19.6	19
60	8	1.5	1	6000	8520	5	5	43.5	38.9	76	48.78	7200	1510.3	46.8	52	9.94	30	9.7	35
60	8	1.5	2	6000	8640	4	4	26.2	39.9	70	42.99	6014.2	984.1	50	50	0	33	0	12
60	8	1.5	3	6000	8580	5	5	94.2	41.6	64	35.07	5526.5	1171.6	52	52	0	31.5	0	12
60	8	1.5	4	6000	8460	5	5	69.8	40.6	55	26.23	2941.7	2824.1	50	50	0	16.6	0	11
60	8	1.5	5	6000	8580	4	4	79.1	38.6	54	28.59	7200	5842.1	45.6	49	6.86	39.2	17.8	7
60	8	3.0	1	4800	7980	5	5	80	22.5	41	45.04	7200	2471.1	29.5	37	20.33	21.9	10.6	15
60	8	3.0	2	4800	8160	5	5	9.6	21.6	48	54.97	7200	640.9	30.2	35	13.64	23.2	9.7	4
60	8	3.0	3	4800	8100	5	5	10.2	21.4	61	64.97	7200	1366.9	28.5	35	18.61	24.2	11.7	19
60	8	3.0	4	4800	8160	4	4	8.1	21.5	59	63.5	7200	4524.8	30.2	34	11.16	24.5	13	11
60	8	3.0	5	4800	7980	5	5	13	21.7	45	51.76	7200	6843.5	30.4	36	15.56	23.1	7.7	29
60	8	4.5	1	4380	7920	5	5	8.6	15.4	33	53.31	7200	1638.8	22.5	28	19.5	22	9.1	16
60	8	4.5	2	4380	7740	5	5	12.4	15.5	33	52.97	7200	4139.4	23.7	27	12.27	38	16.7	12
60	8	4.5	3	4380	7920	5	5	8.1	15.4	40	61.39	7200	3154.2	24.1	29	16.87	27.6	16.7	12
60	8	4.5	4	4380	7920	4	4	7.6	15.6	33	52.79	7200	5865.2	24.2	26	6.86	34.1	6.8	17
60	8	4.5	5	4380	7920	5	5	6.8	15.6	51	69.47	7200	715.2	23.8	28	15.15	34.6	21.8	20

Table A.2: Detailed results for the comparison of symmetry-breaking methods

Instances				Default						Constraints						Orbitopal					
m	*C*	ρ	#	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*
30	2	1.5	1	7200	22.6	35	35.3	808	16.3	7200	33.9	35	3	1623.4	173.9	3387.7	35	35	0	891.1	11.2
30	2	1.5	2	7200	23.5	34	31	1114.2	10.7	6044.4	34	34	0	1508.7	205.9	7200	30.4	34	10.5	1604.7	267.2
30	2	1.5	3	7200	22.5	36	37.6	782.2	56.7	7200	34.5	36	4.2	1392.8	876.7	4079.7	36	36	0	1082.9	106.2
30	2	1.5	4	7200	22.3	34	34.4	910.9	582.4	7200	31.5	34	7.4	1327.8	682.2	6224.6	34	34	0	1960.8	631
30	2	1.5	5	7200	23.9	36	33.7	782.9	1.9	7200	33.8	36	6.1	1817.3	34	6696.5	36	36	0	2169.7	12.3
30	2	3.0	1	7200	18	31	42.1	775	15.8	7200	25.9	31	16.4	1597.8	646.8	7200	26.9	31	13.3	1645.2	36.1
30	2	3.0	2	7200	16.5	29	43	834.1	13.5	7200	24.3	29	16.3	1506.7	2632.7	7200	20.4	29	29.6	1121.9	50.9
30	2	3.0	3	7200	15.2	31	50.9	806.6	58	7200	24.8	31	19.9	1408	199	7200	22.6	31	27	1254.6	38.9
30	2	3.0	4	7200	15.1	27	44	636.1	78	7200	25	27	7.3	1315.8	2369.6	7200	25.4	27	6	1451.7	75.4
30	2	3.0	5	7200	15.9	30	46.9	710.6	67.5	7200	24.5	30	18.5	1679.8	6552.9	7200	24.9	30	17.1	1347.7	6750
30	2	4.5	1	7200	15.8	24	34.2	1251.6	8.7	7200	20	24	16.5	2142.5	107.7	7200	20	24	16.6	1946.5	2.7
30	2	4.5	2	7200	15.8	26	39.1	954.9	8.9	7200	19.3	26	25.9	2000.3	570.3	7200	18.4	26	29.2	1425.4	400.6
30	2	4.5	3	7200	15.5	26	40.4	1064.8	10.4	7200	20.8	26	20	1920.3	439.2	7200	21.9	26	15.9	1997.3	16.2
30	2	4.5	4	7200	16.5	27	38.9	1055	9.3	7200	20	27	25.9	1938.4	909.1	7200	19.8	27	26.8	1797.9	1.1
30	2	4.5	5	7200	17.1	27	36.7	937.6	7.7	7200	21.6	27	20.1	1517.3	40	7200	23.3	27	13.7	2067.8	30.4
35	2	1.5	1	7200	25.1	41	38.9	591.1	286.6	7200	36.1	41	12	813.6	1325.8	7200	34.5	41	15.9	887.1	387.7
35	2	1.5	2	7200	27.7	44	36.9	274.9	51	7200	38.5	44	12.5	849.6	60.6	7200	37	44	15.9	906.8	30.3
35	2	1.5	3	7200	24.8	40	38	666.7	17.2	7200	37.3	40	6.7	855.4	7040.1	7200	36.6	40	8.5	984.6	33.2
35	2	1.5	4	7200	24.8	41	39.4	354.4	26.4	7200	36.1	41	12	774.3	6595.1	7200	36.9	41	9.9	871.4	289.5
35	2	1.5	5	7200	25.1	40	37.3	427	40.6	7200	35.4	40	11.5	771.3	942.7	7200	32.4	40	19.1	770.7	148.3
35	2	3.0	1	7200	14.1	35	59.6	616.4	18.8	7200	26	36	27.8	867.5	241.9	7200	26.1	35	25.3	909.3	252.6
35	2	3.0	2	7200	14.4	33	56.4	443.6	46.2	7200	26.2	34	22.8	736.5	4901.1	7200	23.7	33	28.1	742.1	6702.7
35	2	3.0	3	7200	18.8	35	46.4	365.7	143.4	7200	26.8	36	25.5	985.5	6701.9	7200	26.8	35	23.5	733	328.5
35	2	3.0	4	7200	17.7	32	44.8	590.4	26.2	7200	25.3	33	23.3	813.6	50.5	7200	24.2	32	24.4	771.4	52.8
35	2	3.0	5	7200	16.4	33	50.4	277.2	144.5	7200	26.1	33	20.8	741.3	5882.6	7200	24.7	33	25	720.8	38
35	2	4.5	1	7200	19.3	32	39.7	253.8	37.8	7200	21.2	32	33.9	925.1	460.4	7200	18.9	32	40.8	792.7	49
35	2	4.5	2	7200	17.5	32	45.2	621.6	14.6	7200	20.8	32	35	1078.2	6748.8	7200	20.3	32	36.5	885.2	34.5
35	2	4.5	3	7200	17.1	30	43.1	354.9	26.7	7200	21.6	31	30.4	994.5	2664.9	7200	20.6	30	31.2	845.5	416.2
35	2	4.5	4	7200	14.9	33	54.7	519	24.7	7200	22.4	33	32.1	1023.8	993.5	7200	22.2	33	32.8	868.4	56.5
35	2	4.5	5	7200	16.7	31	46.3	525.3	17.3	7200	21.1	31	32.1	1060	6488	7200	22.2	31	28.3	1078.8	15.4
40	2	1.5	1	7200	26.8	51	47.4	356.9	188.2	7200	40.4	51	20.7	468.6	3696.3	7200	39.7	50	20.5	544.1	266.5
40	2	1.5	2	7200	27.4	48	43	234.1	396.2	7200	37.4	48	22.1	456	6070.5	7200	36.6	48	23.8	416	27.2
40	2	1.5	3	7200	28.7	46	37.7	323.6	34.4	7200	39	46	15.3	539.3	1988.4	7200	38.1	46	17.2	617.4	34.6
40	2	1.5	4	7200	26.9	46	41.5	236.7	322.7	7200	39.2	46	14.8	333.2	6990.5	7200	40.2	46	12.6	573.9	6300.4
40	2	1.5	5	7200	27.9	48	42	229.3	6776.9	7200	40.3	49	17.7	430.6	6566.6	7200	40.5	48	15.7	488.8	2114.9
40	2	3.0	1	7200	16.7	38	56	244	172.7	7200	25.8	39	33.8	452.6	6992.2	7200	25.4	38	33.3	420.4	1937.8
40	2	3.0	2	7200	16.1	37	56.5	219.6	77.6	7200	26	37	29.7	383	2990	7200	25.4	37	31.4	436.4	118.4
40	2	3.0	3	7200	18.5	40	53.9	299.7	259.2	7200	27.2	42	35.3	433.9	1068.2	7200	25.2	40	37	364.3	389.9
40	2	3.0	4	7200	16.8	40	57.9	242	169.6	7200	26.7	40	33.1	533.3	6853.2	7200	23.6	40	40.9	514.7	4050.2

Continued on Next Page...

Table A.2 – Continued

Instances				`Default`						`Constraints`						`Orbitopal`					
m	*C*	*ρ*	#	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*
40	2	3.0	5	7200	15.3	39	60.7	112.1	1191.7	7200	27.2	40	32.1	417.9	6527.5	7200	26.2	39	32.9	492.8	175.2
40	2	4.5	1	7200	13.9	35	60.4	284.2	61.4	7200	20.7	35	40.8	578	6828.5	7200	21.5	35	38.5	488.7	85.4
40	2	4.5	2	7200	16.2	36	54.9	287.3	178	7200	21.8	36	39.5	453.2	2898.6	7200	21.3	36	40.9	484.6	288.6
40	2	4.5	3	7200	16	34	53	243.8	36.1	7200	22.1	34	34.9	470.9	6844.2	7200	23.3	34	31.5	530.3	113.7
40	2	4.5	4	7200	17.2	35	51	419.6	1242	7200	21.9	35	37.4	512.2	117.3	7200	20.9	35	40.3	477.4	4476.8
40	2	4.5	5	7200	15	35	57.1	393.3	465.1	7200	22.1	35	37	532.7	6527.2	7200	23.1	35	34	519.8	104
45	2	1.5	1	7200	29.8	50	40.4	141.1	464.6	7200	40.7	49	17	267.9	6604.6	7200	40.6	49	17.1	356.8	6524.7
45	2	1.5	2	7200	29.8	53	43.7	112.6	102.2	7200	40.7	53	23.2	265.5	4253.1	7200	41.5	53	21.8	314.5	349
45	2	1.5	3	7200	32.4	55	41.1	119.2	101.5	7200	42.8	56	23.6	291.9	6695.1	7200	42.5	55	22.8	348.1	1248.5
45	2	1.5	4	7200	30.3	55	44.9	92.3	502.4	7200	43.5	54	19.5	244.5	6635.7	7200	42.3	54	21.7	334.1	2684.4
45	2	1.5	5	7200	31.1	53	41.4	122	1211.4	7200	43	53	18.9	259.4	6684	7200	42.9	53	19.1	344	277.4
45	2	3.0	1	7200	17.4	44	60.5	69.8	6721.1	7200	29.3	46	36.3	242.2	1220.8	7200	28.3	45	37.1	293.1	2795.4
45	2	3.0	2	7200	18.9	47	59.8	129.4	1709.4	7200	30.8	49	37.2	304	6832.5	7200	30.1	47	35.9	265.4	383.2
45	2	3.0	3	7200	17.7	47	62.4	135.8	547.5	7200	28.4	47	39.5	317.8	6956.9	7200	27.7	46	39.8	284.7	6881.7
45	2	3.0	4	7200	16.9	42	59.7	160.4	6735.9	7200	28.4	44	35.4	263.7	473.3	7200	28.8	42	31.4	288.2	7033.1
45	2	3.0	5	7200	17.3	45	61.6	123.1	324	7200	30.1	46	34.6	236.8	528.5	7200	30	44	31.9	314.8	6665
45	2	4.5	1	7200	17.2	42	59.1	162.2	305.9	7200	23.5	43	45.3	315.5	6566.4	7200	22.7	42	45.8	292.5	5471.1
45	2	4.5	2	7200	18.3	42	56.5	194.8	1607.1	7200	24.3	42	42.2	313.5	6574.5	7200	23.4	42	44.4	333.5	182.4
45	2	4.5	3	7200	17	42	59.5	171	80.7	7200	24.4	43	43.3	314.5	6507.2	7200	24.2	42	42.4	305.5	138.3
45	2	4.5	4	7200	20.5	45	54.5	213.4	72.7	7200	25.3	47	46.2	329.9	4902.5	7200	24.8	46	46.2	333.7	55
45	2	4.5	5	7200	15.3	39	60.8	136.2	635.9	7200	23.6	41	42.6	309.7	6544.7	7200	22.3	39	42.9	287.6	1145.1
50	2	1.5	1	7200	35.5	63	43.6	59.5	1675	7200	47.1	63	25.3	196.8	3625.2	7200	48.6	63	22.9	219.6	320.5
50	2	1.5	2	7200	33.8	60	43.7	62.8	425.8	7200	45.7	59	22.6	201.5	5236.5	7200	45.5	59	22.9	246.7	1453.4
50	2	1.5	3	7200	32.7	58	43.7	97.2	4101.3	7200	43.3	58	25.4	148	4225.6	7200	43.5	58	25.1	240.1	445.1
50	2	1.5	4	7200	32.7	62	47.2	36.8	3667	7200	44.3	60	26.1	188.6	6817.3	7200	45	60	25	255.5	50.2
50	2	1.5	5	7200	33.3	61	45.5	83.7	968.2	7200	45.6	62	26.4	195.3	6567.4	7200	44.8	60	25.4	203.4	6546.9
50	2	3.0	1	7200	19.9	50	60.2	75.1	2101.8	7200	30.2	50	39.6	188.6	3220.4	7200	29.2	51	42.8	180.3	103.8
50	2	3.0	2	7200	20.5	55	62.7	78	387.2	7200	32.3	56	42.4	223.2	7173	7200	32.2	55	41.4	226.1	258.2
50	2	3.0	3	7200	17.9	50	64.3	66.4	6558.9	7200	29.7	50	40.5	166.2	6729.4	7200	29.3	50	41.5	235.9	2733.7
50	2	3.0	4	7200	18.6	49	62.1	52.5	6743.1	7200	29.5	51	42.1	193.2	6955.3	7200	29.6	49	39.5	196.8	6544.3
50	2	3.0	5	7200	18.2	53	65.6	37.1	176	7200	30.2	52	41.9	158.6	6494.7	7200	30.6	52	41.1	198.7	236.8
50	2	4.5	1	7200	12.9	47	72.6	54.5	6769.3	7200	25.1	49	48.7	165.2	1245.4	7200	23.5	47	50	165.6	1964.4
50	2	4.5	2	7200	15.4	44	64.9	86.9	3535.1	7200	24.4	44	44.6	194.3	6529.1	7200	24.4	44	44.6	197	508.6
50	2	4.5	3	7200	16.4	47	65	114.7	3686.9	7200	25.9	48	46	191.8	6744.9	7200	23.8	47	49.4	198	2188
50	2	4.5	4	7200	24.4	49	50.1	46.8	267.7	7200	25.4	49	48.2	176.8	6669.3	7200	25.4	49	48.1	198.1	197.4
50	2	4.5	5	7200	15.6	48	67.5	91.7	754.8	7200	25.7	48	46.5	216.6	4674.7	7200	24.7	48	48.5	191.4	936.5
55	2	1.5	1	7200	37.2	65	42.8	47.4	273	7200	48.8	65	24.9	119.1	6524.9	7200	47.6	64	25.7	167.3	675.4
55	2	1.5	2	7200	38.3	67	42.9	59.8	6415	7200	48.4	70	30.8	149.8	6693.8	7200	47.6	66	27.9	135.2	3639.1
55	2	1.5	3	7200	36.8	63	41.6	40.9	6656.4	7200	48.9	64	23.7	126.2	5922.6	7200	47.9	63	24	116.9	6727

Continued on Next Page...

Table A.2 – Continued

Instances				`Default`						`Constraints`						`Orbitopal`					
m	*C*	*ρ*	#	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*
55	2	1.5	4	7200	36.2	65	44.4	43.8	6940.6	7200	46.4	66	29.7	111.5	6582.8	7200	46.1	64	28	132	7043.8
55	2	1.5	5	7200	34.9	66	47.2	52.3	6571	7200	46.2	66	30	127.1	6898.5	7200	46.6	65	28.4	151.8	6918.7
55	2	3.0	1	7200	20.4	56	63.7	29.2	6802.2	7200	33.2	59	43.7	117.9	6935.1	7200	31.8	58	45.2	159	1583.1
55	2	3.0	2	7200	19.9	55	63.9	33.9	186	7200.2	32.4	57	43.1	115.9	1437.1	7200	32.3	55	41.3	141.3	1746.6
55	2	3.0	3	7200	20.2	58	65.2	22.9	7060.1	7200	32.9	60	45.2	128.8	6792.4	7200	33.5	57	41.2	128	910.7
55	2	3.0	4	7200	19.2	55	65	37.3	7086	7200	32	56	42.8	107	6721	7200	31.3	55	43.1	139.4	6517.3
55	2	3.0	5	7200	19.2	57	66.4	38.9	142.8	7200	30.3	58	47.8	99.2	6706.4	7200	30.2	55	45.1	140.2	6607.7
55	2	4.5	1	7200	13.1	49	73.3	27.7	314.6	7200	23.9	50	52.2	133.8	2219.6	7200	22.8	49	53.4	157.2	6556.8
55	2	4.5	2	7200	14.1	51	72.4	29.7	154.1	7200	25	52	51.9	143.5	6878.4	7200	24.4	52	53.2	135	1217.3
55	2	4.5	3	7200	15.2	51	70.3	57.3	1153.5	7200	24.7	51	51.6	121.5	6669.8	7200	24.9	51	51.1	148.6	1821.1
55	2	4.5	4	7200	14	52	73.2	36.9	3477.8	7200	26.3	53	50.4	147.6	5812.9	7200	26.3	52	49.4	149.8	858.9
55	2	4.5	5	7200	17.2	56	69.4	52.6	3019.1	7200	26.4	59	55.2	134.8	6541.5	7200	27	55	50.9	156.7	6674.5
60	2	1.5	1	7200	40.6	74	45.2	17.2	6615.3	7200	52.7	76	30.7	95.9	7084.3	7200	52.3	74	29.4	88.5	1155.1
60	2	1.5	2	7200	40.6	74	45.1	37.8	4944.3	7200	50.8	75	32.2	100.6	3706	7200	50.1	73	31.4	103.2	1088.1
60	2	1.5	3	7200	39	72	45.8	24.4	6724.5	7200	48.8	72	32.3	89.3	2227.6	7200	48.9	71	31.1	88.9	5785.1
60	2	1.5	4	7200	41.3	72	42.6	34.5	2971.3	7200	49.5	74	33.1	73.4	6749.1	7200	50.3	72	30.1	91.9	2068.4
60	2	1.5	5	7200	40	71	43.6	27.3	6610.9	7200	52.2	74	29.5	84.3	7103.8	7200	52.3	72	27.4	90.6	3979.2
60	2	3.0	1	7200	22	65	66.1	22.5	2426.5	7200	34.6	64	45.9	70.4	6950.6	7200	34	65	47.8	85.1	1666.8
60	2	3.0	2	7200	21.5	63	65.9	16	4509.2	7200	34.3	63	45.5	80.3	6737.3	7200	33.5	62	46	89.4	683.1
60	2	3.0	3	7200	22.8	65	64.9	14.8	5394.1	7200	35.7	67	46.7	92.7	5911.9	7200	35.5	64	44.6	95.9	6296.5
60	2	3.0	4	7200	21.9	64	65.8	32.8	6807.8	7200	34.4	66	47.9	69.6	7026.5	7200	32.9	63	47.9	79.6	7015.1
60	2	3.0	5	7200	21.3	64	66.8	21.3	5721.8	7200	34.1	65	47.5	70.5	6720	7200	33	63	47.7	90.8	3160.9
60	2	4.5	1	7200	13.9	59	76.4	23	6619.6	7200	27.6	59	53.3	74.9	6841.3	7200	27.9	58	51.9	107.6	1600.9
60	2	4.5	2	7200	15.4	57	72.9	17.4	190.8	7200	26.7	60	55.5	75.6	6741	7200	26.3	56	53.1	103.5	6911
60	2	4.5	3	7200	16.2	58	72.1	25.4	2373.7	7200	27.5	58	52.5	83.6	6664.2	7200	25.9	57	54.6	95.1	1214.7
60	2	4.5	4	7200	15.8	59	73.3	18.1	7060.6	7200	27.9	59	52.7	81.3	6514.9	7200	27.2	59	53.9	90.7	4507.7
60	2	4.5	5	7200	15.4	57	72.9	28	4209.2	7200	26.9	57	52.9	85.6	6559	7200	26.6	57	53.4	88	7198.3
30	5	1.5	1	7200	19.5	25	22.1	1318.6	2.3	95.2	25	25	0	15.4	6.4	38.7	25	25	0	7	2.5
30	5	1.5	2	7200	22.4	27	17	1255.4	2	20.7	27	27	0	3.1	1.1	31.8	27	27	0	5.9	2.1
30	5	1.5	3	7200	22.6	27	16.3	1222.2	23.3	28.3	27	27	0	4.6	28.3	4.1	27	27	0	0.5	2.9
30	5	1.5	4	7200	21.4	27	20.6	1108.9	2	45.2	27	27	0	7.3	31.8	36.5	27	27	0	7.3	2.5
30	5	1.5	5	7200	22.2	28	20.7	1138.5	2.4	81.2	28	28	0	11.2	33.9	32.2	28	28	0	6.4	2.9
30	5	3.0	1	7200	12.8	19	32.4	1040.6	2.5	193	19	19	0	34.9	31.7	62.1	19	19	0	11	1.9
30	5	3.0	2	7200	12.5	19	34.2	865.3	13.3	161.4	19	19	0	28.3	78.8	173.4	19	19	0	33.1	27.2
30	5	3.0	3	7200	12.8	18	28.8	1064.4	2.8	85.2	18	18	0	12.5	49.8	121.7	18	18	0	22.2	81.8
30	5	3.0	4	7200	12.8	18	29	1003.8	2.6	130.6	18	18	0	18.1	64.9	140.6	18	18	0	25.4	2.6
30	5	3.0	5	7200	12.7	19	33	1080.7	2.6	272	19	19	0	53.9	1.7	502.6	19	19	0	97.4	2.4
30	5	4.5	1	7200	10.2	14	27.3	1282.2	3	169.1	14	14	0	43.2	127.5	249.6	14	14	0	75.9	4.8
30	5	4.5	2	7200	9.7	13	25.4	1151.3	2.1	28.5	13	13	0	5.7	8.5	58.1	13	13	0	10.1	18.8

Continued on Next Page...

Table A.2 – Continued

Instances				`Default`						`Constraints`						`Orbitopal`					
m	*C*	*ρ*	#	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*
30	5	4.5	3	7200	10.2	13	21.2	1078.9	2.2	54.2	13	13	0	11.8	14.1	168.6	13	13	0	39.1	115.8
30	5	4.5	4	7200	9.7	14	30.6	1500.5	2.9	471.2	14	14	0	120	2.5	1086.8	14	14	0	245.6	8.4
30	5	4.5	5	7200	9.9	13	23.7	1222.3	2.8	35.4	13	13	0	7.1	4.3	35.1	13	13	0	6.6	1.7
35	5	1.5	1	7200	23.4	31	24.6	627.6	4.9	368.8	31	31	0	40.1	109.8	236.7	31	31	0	28	64
35	5	1.5	2	7200	25.1	31	19	711.2	75.4	214.2	31	31	0	18.7	127.6	118.1	31	31	0	14.4	65.4
35	5	1.5	3	7200	23.2	29	20	712.2	47.2	169.4	29	29	0	14.8	118.9	178	29	29	0	24.2	115
35	5	1.5	4	7200	25.2	33	23.7	803.9	4	1317.4	33	33	0	129.8	376.9	2448	33	33	0	325.7	846.8
35	5	1.5	5	7200	24.1	31	22.2	828.1	16.6	882.1	31	31	0	89.1	610.6	1928.1	31	31	0	262	82.9
35	5	3.0	1	7200	13.2	22	39.8	588.6	4.3	2541.7	22	22	0	224.3	78.6	3281.3	22	22	0	405.5	85.3
35	5	3.0	2	7200	13.5	21	35.8	760.5	5.1	853.2	21	21	0	106.3	311.5	855.8	21	21	0	90.5	26.1
35	5	3.0	3	7200	13.7	23	40.6	537.6	181.5	7200	21.2	24	11.7	632.3	601.7	7200	20.8	23	9.6	831	237.7
35	5	3.0	4	7200	13.4	22	39.1	627.9	48.1	7200	20.8	22	5.3	600.3	369.6	7200	18.5	22	15.7	775.6	6359.9
35	5	3.0	5	7200	13.5	21	35.7	574.4	4.2	864	21	21	0	81.7	123.7	1129.9	21	21	0	120.3	112.2
35	5	4.5	1	7200	10.9	18	39.6	717	19.4	7200	17.2	18	4.5	946.3	2406.7	7200	15.6	18	13.3	932.9	88.7
35	5	4.5	2	7200	10.2	16	36.4	614.5	4.3	721.9	16	16	0	81.1	61.9	2547.3	16	16	0	318.8	52.6
35	5	4.5	3	7200	10.1	17	40.8	547.3	4.7	7200	16.4	17	3.5	638.2	49.7	7200	14.5	17	14.5	875.7	71
35	5	4.5	4	7200	10.7	16	33.2	689.2	5.2	821.5	16	16	0	91.5	107	2840.7	16	16	0	357.2	91.3
35	5	4.5	5	7200	10.9	17	36.2	687.2	4.9	2683.5	17	17	0	333.1	27.9	7200	15.4	17	9.3	835.8	3.8
40	5	1.5	1	7200	27.7	37	25	572.7	107.9	7200	36.3	37	1.8	290	1675.8	7200	35.5	37	4.2	593.5	248.5
40	5	1.5	2	7200	27.5	36	23.6	461.3	5.1	2016.1	36	36	0	88.9	318.1	389.3	36	36	0	29.7	95.4
40	5	1.5	3	7200	27.5	37	25.6	441.1	2515.1	7200	35.9	37	3.1	234.3	3038.8	7200	34.9	37	5.8	477.3	5916.7
40	5	1.5	4	7200	27.5	37	25.6	503.8	4	6040.3	37	37	0	341.5	101.5	6451.1	37	37	0	647.2	105.4
40	5	1.5	5	7200	26.7	33	19.1	552.2	142.5	1181.9	33	33	0	53.5	1016.5	675.1	33	33	0	64	464.1
40	5	3.0	1	7200	16	26	38.4	361.8	18	7200	23.1	26	11.2	246.9	517.2	7200	21.3	26	18.1	443.9	3403
40	5	3.0	2	7200	14.8	25	40.9	366	5.7	7200	23.6	25	5.7	317.4	222.9	7200	23.9	25	4.4	559.7	3.2
40	5	3.0	3	7200	15	25	39.9	331.4	101.9	7200	23.6	25	5.6	362.8	89	7200	21.1	25	15.6	539.1	77
40	5	3.0	4	7200	15	25	40	363.8	76.9	7200	22.8	26	12.5	329.5	96.9	7200	22.9	25	8.3	496.1	2829.1
40	5	3.0	5	7200	15.2	26	41.5	325.5	3070.9	7200	23.2	27	14.1	310.2	278.6	7200	23.2	26	10.6	505.1	5153.5
40	5	4.5	1	7200	10.7	19	43.6	348.9	20	7200	17.2	20	13.9	322.2	108.3	7200	15.6	20	22.2	567.3	22.1
40	5	4.5	2	7200	9.9	20	50.5	324.1	7.4	7200	17.9	20	10.6	341.1	1225.4	7200	15.9	20	20.7	551.5	23.7
40	5	4.5	3	7200	11.4	18	36.4	512.2	55.9	7200	16.6	18	7.9	398	318	7200	15.1	18	16.3	536.8	1242.9
40	5	4.5	4	7200	10.4	20	47.9	475.7	18.1	7200	17.4	20	12.8	379.1	1233.3	7200	15.3	20	23.3	500.7	76.5
40	5	4.5	5	7200	10.7	20	46.3	371.5	6.7	7200	17.2	20	13.8	391.8	3839.9	7200	16.4	20	18	566.4	746.6
45	5	1.5	1	7200	30.5	40	23.8	337.9	365.1	7200	38.5	40	3.7	159.2	230.1	7200	35.7	40	10.9	413.1	85.5
45	5	1.5	2	7200	31.7	41	22.7	377.9	52	7200	38.7	41	5.6	188	3445.5	7200	37	41	9.8	441	121.5
45	5	1.5	3	7200	29.5	41	28	204.4	330.9	7200	38.5	41	6.1	174.1	766	7200	37.8	41	7.8	372	1419
45	5	1.5	4	7200	31.6	42	24.7	284.5	138	7200	38.9	43	9.5	182.9	1883	7200	38.4	42	8.6	351.5	2239.6
45	5	1.5	5	7200	29.6	43	31.3	218.4	2528.8	7200	37.5	43	12.8	214	1537.6	7200	37.2	43	13.4	356	280.2
45	5	3.0	1	7200	16.5	31	46.8	198.1	137.1	7200	24.1	30	19.5	150.3	2816.2	7200	23.9	31	23	321.8	265.5

Continued on Next Page...

Table A.2 – Continued

Instances				Default						Constraints						Orbitopal					
m	*C*	ρ	#	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*
45	5	3.0	2	7200	16.6	31	46.3	206.5	278.9	7200	25.1	31	19.1	158	1903.8	7200	24.7	31	20.4	271.4	2133.1
45	5	3.0	3	7200	16.8	30	44	183.9	1476.3	7200	24.3	32	24.2	162	724.1	7200	24.3	30	18.9	282.6	6499.3
45	5	3.0	4	7200	16.5	32	48.4	186.4	47.2	7200	24.9	32	22.1	185.6	4414.1	7200	25.5	32	20.3	333.2	1732.9
45	5	3.0	5	7200	16.7	31	46.2	240.5	124.1	7200	24.2	31	22	171.5	6342.7	7200	23	32	28.2	329.6	1742.5
45	5	4.5	1	7200	12.2	25	51.3	201.9	358	7200	19.3	25	23	217.9	6588.6	7200	18.3	25	26.9	354.4	487.7
45	5	4.5	2	7200	11.6	23	49.6	275.7	9.2	7200	19.3	23	16	169.8	1798	7200	17.7	23	23	330.7	193.1
45	5	4.5	3	7200	12.3	24	48.9	149	53.4	7200	19.5	24	18.6	212.8	6575	7200	19.1	24	20.4	357.7	53.3
45	5	4.5	4	7200	11.6	25	53.4	286.2	28.8	7200	18.7	25	25.2	178.3	5185.9	7200	18.3	25	26.8	320.8	228.8
45	5	4.5	5	7200	12.2	24	49	205.7	55.7	7200	19.1	25	23.8	200.9	114	7200	18.6	24	22.6	366.3	226.9
50	5	1.5	1	7200	33	46	28.3	192.6	3576.7	7200	41.2	47	12.3	105.1	2687.7	7200	40.9	46	11	253.8	1048.9
50	5	1.5	2	7200	33.9	47	27.8	146.1	77	7200	41.2	47	12.3	109.8	391.6	7200	41	47	12.8	266.9	125.6
50	5	1.5	3	7200	34	46	26.1	169	86.7	7200	41.7	46	9.2	99.9	868.9	7200	40.9	46	11.1	256.5	725.4
50	5	1.5	4	7200	32	48	33.4	120.8	3012.1	7200	40.5	49	17.3	98.8	4182	7200	38.5	48	19.8	193	1683.7
50	5	1.5	5	7200	32.9	48	31.5	133.5	593.5	7200	40.4	48	15.8	90.9	3020.1	7200	40.6	48	15.4	186.6	6624.9
50	5	3.0	1	7200	17.3	31	44.2	146.6	882.7	7200	24.5	34	27.9	96.7	1392.1	7200	24.6	32	23.2	193.9	1102.4
50	5	3.0	2	7200	17	35	51.4	114	6619.7	7200	24.9	36	30.9	103.9	6882	7200	24.7	37	33.2	204.2	148.5
50	5	3.0	3	7200	17	34	50	116	6609.2	7200	24.4	35	30.2	99	572.7	7200	25	35	28.5	201.6	3023.1
50	5	3.0	4	7200	17.3	34	49.2	116.2	1859.3	7200	24.9	35	28.9	118.4	4569.8	7200	25.6	34	24.7	182	1651.3
50	5	3.0	5	7200	17.3	30	42.4	156.7	253.2	7200	24.9	31	19.5	88	3876.8	7200	24.6	30	18.1	210	171.5
50	5	4.5	1	7200	12.6	29	56.7	117	5037.3	7200	20.4	31	34.2	154.6	7080.1	7200	20.6	30	31.5	252.6	4805.5
50	5	4.5	2	7200	13.1	30	56.5	157	296.7	7200	19.8	31	36	139.4	6953.6	7200	20.1	31	35	260.2	204.9
50	5	4.5	3	7200	12.4	27	54.1	147	6953.8	7200	19.7	28	29.7	111.2	6705.7	7200	19.5	28	30.4	244.3	1352.3
50	5	4.5	4	7200	12.4	26	52.4	144	169.8	7200	19.1	27	29.1	101.3	7110	7200	18.8	26	27.8	234.2	666.5
50	5	4.5	5	7200	12.3	29	57.5	148.4	915.1	7200	19.2	31	37.9	125.2	6705.4	7200	20.3	30	32.3	231.8	2803.1
55	5	1.5	1	7200	35.8	52	31.1	69.2	251.2	7200	44	51	13.8	62.5	4239.6	7200	43	51	15.7	138	2282.7
55	5	1.5	2	7200	35.8	51	29.7	58.9	5360.7	7200	44.1	51	13.5	68.2	554.3	7200	43.6	51	14.6	143.8	2049.4
55	5	1.5	3	7200	36.9	50	26.3	116.8	254.7	7200	44	50	12	77.7	2080.4	7200	42.8	50	14.5	174.8	131.6
55	5	1.5	4	7200	33.8	50	32.3	67.7	388.7	7200	40.8	51	20.1	71.3	652.9	7200	40	50	20.1	103.2	565.5
55	5	1.5	5	7200	36.9	52	29.1	103.2	4968.2	7200	44.2	53	16.5	82	131.7	7200	43.7	52	16	136.6	2459.4
55	5	3.0	1	7200	18.9	39	51.7	61.2	6541.4	7200	26.6	41	35.1	76.6	6685	7200	26.6	39	31.7	147.7	7192.4
55	5	3.0	2	7200	18.8	36	47.7	70.3	7092.1	7200	26	37	29.8	76.4	6825.9	7200	25.8	38	32.1	150.7	444.2
55	5	3.0	3	7200	18.8	38	50.4	73.7	5850.3	7200	26.2	40	34.6	55.6	1615.3	7200	25.9	39	33.6	166.4	469.6
55	5	3.0	4	7200	18.8	34	44.6	83.1	3490.9	7200	26.9	37	27.4	58.6	7024.2	7200	26.9	34	20.8	137.7	685.2
55	5	3.0	5	7200	18.9	36	47.6	84.7	108.5	7200	27.2	37	26.6	67.3	6798.6	7200	25.6	37	30.9	168.1	315.8
55	5	4.5	1	7200	12.8	31	58.6	79.8	4981.9	7200	19.8	32	38.1	58.9	3799.8	7200	19.3	31	37.7	155.6	372.7
55	5	4.5	2	7200	13	30	56.7	92	1169.8	7200	20.4	32	36.4	94.9	1592.1	7200	20.7	30	31.2	154.2	3467.7
55	5	4.5	3	7200	13	32	59.4	61	356	7200	20	33	39.5	67.7	6814	7200	20	33	39.3	152.2	546.9
55	5	4.5	4	7200	12.8	31	58.6	102.1	136.6	7200	20	32	37.5	77	1092.1	7200	20.3	32	36.6	158	826
55	5	4.5	5	7200	12.9	33	61	80	153.1	7200	21.3	33	35.3	75.3	917.6	7200	20.6	32	35.5	165	6693.9

Continued on Next Page...

Table A.2 – Continued

Instances				`Default`						`Constraints`						`Orbitopal`					
m	*C*	ρ	#	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*
60	5	1.5	1	7200	36.8	50	26.4	66.3	409.1	7200	44.5	50	11	35.2	4006.1	7200	44.1	50	11.8	91.4	3359.9
60	5	1.5	2	7200	38.8	54	28.1	53.7	229.4	7200	45.3	56	19.2	44.3	1558	7200	46.2	53	12.8	113.8	6622.3
60	5	1.5	3	7200	38.8	57	31.9	54.8	6147.9	7200	46.6	56	16.8	48.3	6690.5	7200	46.7	56	16.6	109.7	902.2
60	5	1.5	4	7200	37.8	56	32.4	59.4	1847.4	7200	44.9	55	18.3	44.6	6708.9	7200	46	55	16.4	90.3	6712.3
60	5	1.5	5	7200	38.9	58	33	67.9	7096.6	7200	46.5	60	22.5	57.9	6607	7200	46.5	58	19.8	91.5	1114.4
60	5	3.0	1	7200	20.9	43	51.5	43.2	251.8	7200	28.3	44	35.8	42.3	6579	7200	28.4	44	35.4	61	6765.8
60	5	3.0	2	7200	20.8	42	50.4	45.4	5608.3	7200	28.1	42	33.1	40.6	2493.8	7200	28.3	41	30.9	77.8	1417.5
60	5	3.0	3	7200	20.9	41	49.1	33.9	7101.8	7200	28.7	42	31.6	49.5	6725.1	7200	29.7	42	29.4	80.3	6742.4
60	5	3.0	4	7200	20.8	39	46.5	52.4	6687.9	7200	28.6	41	30.2	40.5	2869	7200	28.8	38	24.2	86.5	736.3
60	5	3.0	5	7200	20.9	44	52.6	46.8	6559.8	7200	28.5	45	36.8	52.9	6876.1	7200	29.8	44	32.4	70.2	2337.4
60	5	4.5	1	7200	13.8	32	56.8	57.3	2977.9	7200	20.9	33	36.7	47.9	3943.5	7200	20.9	32	34.6	113.9	1159.6
60	5	4.5	2	7200	13.9	35	60.4	43.9	753.5	7200	20.6	36	42.7	46.6	3203	7200	21.3	35	39.2	89	7018.6
60	5	4.5	3	7200	13.9	33	58	48.5	7038.7	7200	21.3	35	39.2	43.6	3344.2	7200	20.6	34	39.4	95.6	6540
60	5	4.5	4	7200	13.8	34	59.3	44.9	6777.1	7200	20.3	35	41.9	44.7	3491.9	7200	21	34	38.1	93	2960.6
60	5	4.5	5	7200	13.8	33	58.1	48.3	4221.1	7200	20.8	34	38.9	45.2	7131.8	7200	20.8	34	38.9	87.3	7179.5
30	8	1.5	1	2351.4	24	24	0	538	1.1	4.5	24	24	0	0.3	0.9	3.5	24	24	0	0.4	1.2
30	8	1.5	2	3311.8	23	23	0	569.7	1.6	3.3	23	23	0	0.1	1.2	3	23	23	0	0	1.9
30	8	1.5	3	7200	21.2	24	11.7	1037.3	1.5	5.5	24	24	0	0.4	3	3.5	24	24	0	0.2	1.6
30	8	1.5	4	7200	20.5	23	10.8	1200.1	1.8	6	23	23	0	0.7	3	8.2	23	23	0	0.9	1.7
30	8	1.5	5	3134.4	24	24	0	494.7	1.3	4.3	24	24	0	0.3	1.2	3.3	24	24	0	0.1	1.5
30	8	3.0	1	7200	12.4	16	22.5	982.9	1.8	10.5	16	16	0	1.4	9.2	6.6	16	16	0	0.8	5.8
30	8	3.0	2	233.4	13	13	0	35.3	1.1	1.4	13	13	0	0	1.2	1.9	13	13	0	0	1.2
30	8	3.0	3	1955.5	14	14	0	357	1.2	5	14	14	0	0.3	5	2.3	14	14	0	0	1.3
30	8	3.0	4	7200	12.5	15	16.4	1131	1.1	6.1	15	15	0	0.5	3.2	5.5	15	15	0	0.4	1.2
30	8	3.0	5	7200	12.5	15	16.5	1097.6	2.1	9.1	15	15	0	0.6	4.2	3.4	15	15	0	0.1	1.7
30	8	4.5	1	7200	8.6	11	21.6	970.5	1.6	13.7	11	11	0	2.3	5.6	7	11	11	0	0.5	1.7
30	8	4.5	2	7200	9.9	11	10.3	1291.2	1.7	18.5	11	11	0	5.7	2.4	107.6	11	11	0	27.2	1.5
30	8	4.5	3	7200	9.1	12	24.3	1346.4	1.7	46.6	12	12	0	11.1	1.8	425.1	12	12	0	102.2	1.8
30	8	4.5	4	43.2	9	9	0	6	0.1	2	9	9	0	0.1	1.5	2.2	9	9	0	0	1.1
30	8	4.5	5	7200	9.1	12	23.8	1356.8	2.3	35.3	12	12	0	6.2	3.2	154.4	12	12	0	38.3	3.9
35	8	1.5	1	7200	25	29	14	701.3	3.7	16.1	29	29	0	0.9	1.9	19.3	29	29	0	1.8	3.3
35	8	1.5	2	7200	23.8	25	4.7	821.8	1.9	5.9	25	25	0	0.1	3	4.8	25	25	0	0.1	2.4
35	8	1.5	3	7200	24.7	29	14.7	638.2	3	23.6	29	29	0	1.2	1.7	7.8	29	29	0	0.3	2.4
35	8	1.5	4	7200	24.3	27	9.9	744.5	2.6	12.8	27	27	0	0.5	12.7	4.8	27	27	0	0.1	2.6
35	8	1.5	5	7200	24	28	14.1	920.7	3.7	63.4	28	28	0	5.7	2.2	54.9	28	28	0	7.3	2.3
35	8	3.0	1	7200	13	18	27.8	665.1	3.5	83.9	18	18	0	5.9	14.5	25.5	18	18	0	2.3	2.5
35	8	3.0	2	7200	13.2	18	26.5	949.5	4.2	73.7	18	18	0	6.1	17.9	72	18	18	0	8.6	2.6
35	8	3.0	3	201.5	14	14	0	34.4	0.2	3.8	14	14	0	0	3.2	3.1	14	14	0	0	3
35	8	3.0	4	7200	12.9	17	24.1	686.2	2.9	22.8	17	17	0	1.5	9.9	12.3	17	17	0	0.8	2.8

Continued on Next Page...

Table A.2 – Continued

Instances				`Default`						`Constraints`						`Orbitopal`					
m	*C*	*ρ*	#	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*
35	8	3.0	5	7200	12.8	18	28.9	707.9	2.8	99.6	18	18	0	5.1	1.4	24.3	18	18	0	2	2.7
35	8	4.5	1	7200	9.4	12	21.7	732	3.6	51.2	12	12	0	5.9	23.7	43.6	12	12	0	5.9	3
35	8	4.5	2	7200	9.3	13	28.7	800.3	3	36.5	13	13	0	2.8	21	34.6	13	13	0	3.6	2.6
35	8	4.5	3	7200	9.5	14	32.4	803.5	3.1	446.1	14	14	0	59.1	1.8	138.3	14	14	0	16.2	2.5
35	8	4.5	4	7200	9.4	12	22	711.5	3.8	10	12	12	0	0.4	6.7	5.1	12	12	0	0.2	4.6
35	8	4.5	5	7200	9.2	13	29.6	842.4	3.1	156.3	13	13	0	15.7	12.3	267.8	13	13	0	41.8	2.4
40	8	1.5	1	7200	26.6	32	17	394.5	4.4	79.2	32	32	0	2.9	2.9	30.3	32	32	0	1.4	4
40	8	1.5	2	7200	26.7	31	14	504.9	3.8	101.5	31	31	0	2.5	101.5	24.6	31	31	0	1.4	5.6
40	8	1.5	3	7200	24.4	31	21.2	457.5	6.2	265.5	31	31	0	9.9	106.5	176.8	31	31	0	11.7	25.1
40	8	1.5	4	7200	27.7	34	18.6	513.5	5.9	561.2	34	34	0	27.3	36.8	513.2	34	34	0	40.8	112.1
40	8	1.5	5	7200	27.6	32	13.7	451.2	4.1	26.7	32	32	0	0.8	2.6	10.5	32	32	0	0.2	3.7
40	8	3.0	1	7200	14.5	22	34.1	402.4	4.4	1010.3	22	22	0	42.1	21.5	968.4	22	22	0	76.1	25.5
40	8	3.0	2	7200	14.8	20	25.9	396.9	5	109.9	20	20	0	4.6	13.1	79.9	20	20	0	4.8	68
40	8	3.0	3	7200	14.5	23	37	386.8	6.7	2873.7	23	23	0	144.2	48.1	3802.6	23	23	0	291.8	218.7
40	8	3.0	4	7200	14.7	22	33.3	379.1	102.9	931.8	22	22	0	42.2	150.6	522.1	22	22	0	41.4	279.7
40	8	3.0	5	7200	14.7	22	33.2	390.3	7.2	482.5	22	22	0	28.7	188.1	584.2	22	22	0	38.3	243.6
40	8	4.5	1	7200	10.2	14	27.4	432.8	4.5	75.3	14	14	0	1.6	1.5	76.7	14	14	0	5.9	69.5
40	8	4.5	2	7200	9.7	17	42.7	476.4	5.9	4181	17	17	0	336	222.6	4477.6	17	17	0	348	86.4
40	8	4.5	3	7200	9.6	14	31.4	483.8	4.4	211	14	14	0	9.7	203.5	921.6	14	14	0	103.3	813
40	8	4.5	4	7200	9.7	14	31.1	405.7	5.2	115.5	14	14	0	6.6	9.4	34.8	14	14	0	1.7	5.9
40	8	4.5	5	7200	10.1	17	40.8	447.9	5.6	5319.4	17	17	0	410.4	11.3	7200	15.1	17	11.3	559.4	20.3
45	8	1.5	1	7200	31.3	38	17.6	259.2	325.2	384.2	38	38	0	12.2	170.2	499.8	38	38	0	31.2	307.7
45	8	1.5	2	7200	31.4	38	17.5	225.6	198.2	684.4	38	38	0	16.1	602.3	299.5	38	38	0	14.9	70.7
45	8	1.5	3	7200	28.4	36	21.2	289.7	6.3	2093.1	36	36	0	65.7	253.6	1291.4	36	36	0	66.1	6.1
45	8	1.5	4	7200	29.3	36	18.6	263.7	8.1	733.9	36	36	0	15	355.8	398.1	36	36	0	18.5	18.3
45	8	1.5	5	7200	30.3	36	15.9	366.1	5.6	119.8	36	36	0	2.7	37.4	84.5	36	36	0	3.3	8
45	8	3.0	1	7200	16.5	25	34.1	276.9	33.4	5576.2	25	25	0	157.3	162	5692.2	25	25	0	301.1	22.3
45	8	3.0	2	7200	16.4	26	37	251.4	8.8	7200	24.1	26	7.4	153	239.3	7200	23.6	26	9.3	326.4	223.7
45	8	3.0	3	7200	16.3	24	32.2	244.2	7.3	3923	24	24	0	91.7	2820.4	1068.3	24	24	0	42.6	74.9
45	8	3.0	4	7200	16.4	25	34.3	279.3	822.8	5294.8	25	25	0	123.5	3545	7200	22.7	25	9	371	6945.2
45	8	3.0	5	7200	16.5	24	31.4	280.9	13.8	1699.1	24	24	0	48.7	474.5	2108.8	24	24	0	101.6	568.8
45	8	4.5	1	7200	11.6	19	38.9	290.6	74.4	3685.3	19	19	0	123.2	809	7200	18	19	5.1	304.3	383.4
45	8	4.5	2	7200	11.6	18	35.5	262.3	7.7	1094.4	18	18	0	26.1	86.3	1153.8	18	18	0	52.5	5.6
45	8	4.5	3	7200	11.4	18	36.6	231.7	8.4	1042.8	18	18	0	21.1	191.2	409	18	18	0	13.9	129.5
45	8	4.5	4	7200	11.4	21	45.7	321.5	8.4	7200	18	21	14.1	174	623.1	7200	17.5	21	16.5	372.7	163.9
45	8	4.5	5	7200	11.5	20	42.3	299.4	26.4	7200	18	21	14.4	185.2	493.6	7200	16.8	20	16	339	317.8
50	8	1.5	1	7200	33.1	41	19.3	193.7	11.5	1311.9	41	41	0	22.4	134.7	481.5	41	41	0	16.1	88.3
50	8	1.5	2	7200	31	38	18.4	191.7	282.2	2597.4	38	38	0	33.7	2463.6	2544	38	38	0	126.8	2130.1
50	8	1.5	3	7200	31	39	20.5	176.2	90.9	3486.4	39	39	0	71.1	129.1	2969.1	39	39	0	80.6	12.8

Continued on Next Page...

Table A.2 – Continued

Instances				`Default`						`Constraints`						`Orbitopal`					
m	*C*	*ρ*	#	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*
50	8	1.5	4	7200	32	41	22	161.5	939.4	7200	40.4	42	3.7	94.5	1338.1	2796.7	41	41	0	86.6	422.4
50	8	1.5	5	7200	33	42	21.4	161.1	6673.9	7200	40.6	43	5.5	99.6	1454.1	7200	40.3	42	4.1	233.2	635
50	8	3.0	1	7200	17	28	39.3	155.6	126.1	7200	24.5	29	15.7	88.1	1218.4	7200	24	29	17.3	202.6	302.5
50	8	3.0	2	7200	17.1	28	38.9	170.7	66.2	7200	24.9	28	11.1	73.8	2215.3	7200	24.3	28	13.1	190.8	61.8
50	8	3.0	3	7200	17	28	39.3	170.6	1460.8	7200	24.1	28	13.8	100.6	4461	7200	23	29	20.5	230.9	1464.2
50	8	3.0	4	7200	17	28	39.3	163.1	905.3	7200	24.4	29	15.9	88.6	163.7	7200	24.2	28	13.4	249.5	7035.8
50	8	3.0	5	7200	17.1	29	41.2	152	14.3	7200	24.9	29	14	105.9	2321.4	7200	25	29	13.8	250.1	3058.8
50	8	4.5	1	7200	12.4	22	43.8	179.5	43.7	7200	19	22	13.8	101.8	4161.7	7200	18	22	18.2	236.8	823.3
50	8	4.5	2	7200	12.6	22	42.9	210.4	11.6	7200	19.4	23	15.7	145.9	126.1	7200	17.7	22	19.7	276.9	333.5
50	8	4.5	3	7200	12.4	22	43.7	208	62.1	7200	19	22	13.7	117.9	520.8	7200	17.7	22	19.5	247.6	75.6
50	8	4.5	4	7200	12.5	22	43.3	179.7	11.1	7200	19	22	13.7	93.1	3206.2	7200	18.2	22	17.3	246	1079.7
50	8	4.5	5	7200	12.5	23	45.5	168.3	113.8	7200	19.5	23	15.1	145.3	6527.5	7200	18.5	24	22.7	293.2	376.1
55	8	1.5	1	7200	34.7	44	21.1	99.9	1001.9	7200	42	44	4.6	56.3	5978.7	7200	41.1	44	6.5	165.2	722.4
55	8	1.5	2	7200	35.8	45	20.4	149.7	505.4	7200	43.4	45	3.6	58.6	1165.9	7200	42.8	45	5	183.1	273.2
55	8	1.5	3	7200	35.7	44	18.9	105	168.4	2613.9	44	44	0	25.1	320.5	2082.6	44	44	0	53.7	28.2
55	8	1.5	4	7200	36.7	46	20.2	94.4	75.8	7200	44.6	46	3.1	64.3	639.8	7200	43.8	46	4.8	184	349.7
55	8	1.5	5	7200	34.8	47	26	64.5	6650.2	7200	42.8	48	10.9	59.7	2028.9	7200	42.1	47	10.4	158.5	4242
55	8	3.0	1	7200	18.8	32	41.4	99.7	77.9	7200	25.4	33	23	60.6	2962.3	7200	25	31	19.4	156.8	539.3
55	8	3.0	2	7200	18.7	31	39.6	97.7	1885.9	7200	25.3	31	18.3	51.4	4859.4	7200	24.7	31	20.3	139.5	771.1
55	8	3.0	3	7200	18.7	31	39.5	86.8	6640.6	7200	26.3	32	17.8	50.2	6685.6	7200	26.1	31	15.9	139.4	946.2
55	8	3.0	4	7200	18.8	33	43.2	96	1207	7200	26.1	33	20.8	50	4626.6	7200	25.6	33	22.5	182.3	1353.7
55	8	3.0	5	7200	18.8	34	44.8	75.6	1603.5	7200	26.4	34	22.5	66.4	6523.1	7200	26.1	34	23.1	182	4797.3
55	8	4.5	1	7200	13	24	45.8	130.5	2460.5	7200	19.9	26	23.5	63.6	1220.6	7200	18.9	24	21.2	167.4	6549.8
55	8	4.5	2	7200	13	23	43.5	119.1	217.8	7200	19.1	23	17.1	62.8	681	7200	18.6	24	22.6	190.7	497.8
55	8	4.5	3	7200	13	26	50	112.1	124.3	7200	20.3	27	24.8	67.8	876.8	7200	19.9	26	23.6	178.7	1554.7
55	8	4.5	4	7200	12.9	23	44.1	123	13.2	7200	18.4	24	23.4	72.8	58.2	7200	17.8	24	25.8	182.4	6.3
55	8	4.5	5	7200	12.9	25	48.2	124	364.7	7200	19.3	26	25.7	65.8	168.5	7200	18.4	25	26.4	193	867.4
60	8	1.5	1	7200	37.8	51	25.9	58.5	2297.8	7200	44.5	53	16	34.7	2864.9	7200	44.4	51	12.9	100.2	6574.7
60	8	1.5	2	7200	38.7	50	22.5	70.4	241.5	7200	45.9	50	8.2	39.9	2061	7200	45.6	50	8.8	107.7	1272.9
60	8	1.5	3	7200	40.8	53	23.1	54.3	4192.2	7200	47.9	53	9.6	49	7171.7	7200	48	52	7.8	107	863.8
60	8	1.5	4	7200	39.8	50	20.5	59	6609.3	7200	47.3	50	5.5	41.4	2128.2	7200	48.3	50	3.4	121	4792.1
60	8	1.5	5	7200	36.7	49	25	72.7	65.9	7200	44.3	49	9.7	41.5	3355.7	7200	43.3	49	11.6	114.7	6853.9
60	8	3.0	1	7200	20.8	37	43.8	57.9	21.5	7200	27.8	38	26.9	41.7	254.8	7200	27.8	37	25	103.8	2753.6
60	8	3.0	2	7200	20.8	36	42.3	56.1	1076.4	7200	27.3	37	26.3	43.8	6991.8	7200	27.2	36	24.4	99.3	467.2
60	8	3.0	3	7200	19.8	35	43.5	51.1	2663	7200	26.4	36	26.7	38.4	4576	7200	26.1	35	25.4	83.4	6502.4
60	8	3.0	4	7200	20.7	34	39	63	3674.2	7200	28.4	34	16.5	39.1	3930.3	7200	29.2	34	14	100.3	2766.1
60	8	3.0	5	7200	20.8	36	42.3	63.1	4439.3	7200	27.7	38	27.2	46.8	5112.6	7200	28.7	37	22.5	89.6	3489.7
60	8	4.5	1	7200	13.8	27	49	69.8	356.3	7200	20	29	31.2	38.7	6418.7	7200	20.2	28	27.9	99.3	6566.7
60	8	4.5	2	7200	13.8	27	49	67.1	2836	7200	20.7	28	26.1	37.5	2696.9	7200	20.3	27	24.8	140.8	6533.4

Continued on Next Page...

Table A.2 – Continued

Instances				`Default`						`Constraints`						`Orbitopal`					
m	*C*	ρ	#	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*	*CPU*	*lb*	*ub*	*gap*	*node*	*timeBest*
60	8	4.5	3	7200	13.8	29	52.4	64.1	234.2	7200	21	31	32.4	54.4	2600.1	7200	20.5	29	29.2	124.9	7031.6
60	8	4.5	4	7200	13.8	26	47.1	81.2	870.8	7200	19.6	27	27.3	29.6	7040.6	7200	20.1	26	22.7	108.1	679.6
60	8	4.5	5	7200	13.8	28	50.7	63.2	1305.6	7200	19.3	29	33.4	70.4	6588.6	7200	19.1	28	31.6	136.7	909.9

Table A.3: Detailed results for evaluation of the impact of the new upper bound on p_{opt}

Instances				Default							Improved Bounds							Minimum						
m	C	ρ	#	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
30	2	1.5	1	30	1500	11	35	7200	35.3	808	15	750	10	35	7200	22.7	2233.3	10	500	10	35	7200.1	16.5	3129.3
30	2	1.5	2	30	1500	9	34	7200	31	1114.2	15	750	9	34	7200	22	3221.5	8	400	8	34	7200	8.2	5115.1
30	2	1.5	3	30	1500	9	36	7200	37.6	782.2	15	750	9	36	7200	23.2	2316.5	9	450	9	36	7200	17.3	2908.3
30	2	1.5	4	30	1500	8	34	7200	34.4	910.9	15	750	8	34	7200	21.9	2409.5	8	400	8	34	7200	12.3	4971.4
30	2	1.5	5	30	1500	9	36	7200	33.7	782.9	15	750	9	36	7200	19.9	2112.6	9	450	9	36	7200	13.9	4048.5
30	2	3.0	1	30	1200	11	31	7200	42.1	775	15	600	10	31	7200	35.6	2571.2	10	400	10	31	7200	28	3912.1
30	2	3.0	2	30	1200	10	29	7200	43	834.1	15	600	10	29	7200	35.4	2787.7	10	400	10	29	7200	26.7	3884.4
30	2	3.0	3	30	1200	10	31	7200	50.9	806.6	15	600	10	31	7200	38	2211.4	10	400	10	31	7200.1	29.7	3106.7
30	2	3.0	4	30	1200	8	27	7200	44	636.1	15	600	8	27	7200	33.9	2061.9	8	320	8	27	7200	19.8	3980.8
30	2	3.0	5	30	1200	10	30	7200	46.9	710.6	15	600	10	30	7200	35	2134	10	400	10	30	7200	26.1	3164.8
30	2	4.5	1	30	1080	7	24	7200	34.2	1251.6	15	540	7	24	7200	25.2	3145.8	7	252	7	24	180.9	0	221.7
30	2	4.5	2	30	1080	9	26	7200	39.1	954.9	15	540	9	26	7200	28.5	3541.4	9	324	9	26	7200	19.7	4534.6
30	2	4.5	3	30	1080	9	26	7200	40.4	1064.8	15	540	9	26	7200	28.7	4144.5	9	324	9	26	7200	12.9	6468.7
30	2	4.5	4	30	1080	9	27	7200	38.9	1055	15	540	9	27	7200	28.2	4475	9	324	9	27	7200	9.3	5585.5
30	2	4.5	5	30	1080	10	27	7200	36.7	937.6	15	540	10	27	7200	24.3	3179.4	10	360	10	27	7200	15.4	3543.3
35	2	1.5	1	35	2030	11	41	7200	38.9	591.1	18	1044	11	41	7200	27.9	1253.2	10	580	10	41	7200	19.4	2127.8
35	2	1.5	2	35	2030	13	44	7200	36.9	274.9	18	1044	13	44	7200	34.2	1619	13	754	13	44	7200	32.1	2528.4
35	2	1.5	3	35	2030	12	40	7200	38	666.7	18	1044	12	40	7200	30	1711.8	11	638	11	41	7200	21.5	2446.9
35	2	1.5	4	35	2030	11	41	7200	39.4	354.4	18	1044	11	41	7200	32.4	1410.6	11	638	11	41	7200	24.5	2511.7
35	2	1.5	5	35	2030	12	40	7200	37.3	427	18	1044	12	40	7200	35.5	1560.7	12	696	12	40	7200	22.7	2723.5
35	2	3.0	1	35	1610	12	35	7200	59.6	616.4	18	828	12	35	7200	51	1506.4	12	552	12	35	7200	41.3	2731.2
35	2	3.0	2	35	1610	10	33	7200	56.4	443.6	18	828	10	33	7200	51.4	1354.9	10	460	10	33	7200	34.3	2311.7
35	2	3.0	3	35	1610	12	35	7200	46.4	365.7	18	828	12	35	7200	41	1509	12	552	12	35	7200	25.1	2190.3
35	2	3.0	4	35	1610	11	32	7200	44.8	590.4	18	828	11	32	7200	36.1	1753.6	11	506	11	32	7200	24.7	3338.3
35	2	3.0	5	35	1610	10	33	7200	50.4	277.2	18	828	10	33	7200	43.6	1663.3	10	460	10	33	7200	32.1	2316.8
35	2	4.5	1	35	1470	12	32	7200	39.7	253.8	18	756	12	32	7200	49.1	1808.3	12	504	12	32	7200	35.7	3653.6
35	2	4.5	2	35	1470	11	32	7200	45.2	621.6	18	756	11	32	7200	39.2	2466	11	462	11	32	7200	24.7	3344.5
35	2	4.5	3	35	1470	10	30	7200	43.1	354.9	18	756	10	30	7200	34.7	1423.4	10	420	10	30	7200	20.4	2807.8
35	2	4.5	4	35	1470	12	33	7200	54.7	519	18	756	12	33	7200	49.6	2015.3	12	504	12	33	7200	38.7	2680.2
35	2	4.5	5	35	1470	10	31	7200	46.3	525.3	18	756	10	31	7200	39	2465.6	10	420	10	31	7200	26.4	4401.4
40	2	1.5	1	40	2640	14	51	7200	47.4	356.9	20	1320	13	50	7200	40.4	854.4	11	726	11	50	7200	29.3	1823.4
40	2	1.5	2	40	2640	14	48	7200	43	234.1	20	1320	12	48	7200	38.4	1000.9	11	726	11	48	7200	27.8	1873.9
40	2	1.5	3	40	2640	14	46	7200	37.7	323.6	20	1320	14	46	7200	33.4	1258.9	12	792	12	46	7200	25.7	2127.9
40	2	1.5	4	40	2640	11	46	7200	41.5	236.7	20	1320	12	46	7200	35.7	965.1	11	726	11	46	7200	25.3	1858.9
40	2	1.5	5	40	2640	13	48	7200	42	229.3	20	1320	13	48	7200	36.1	1010.8	11	726	11	49	7200	27	1920.5
40	2	3.0	1	40	2120	11	38	7200	56	244	20	1060	11	38	7200	52.8	1087.7	11	583	11	38	7200	38	1647.1
40	2	3.0	2	40	2120	10	37	7200	56.5	219.6	20	1060	10	37	7200	47.6	618.4	10	530	10	37	7200	35.7	1608.7
40	2	3.0	3	40	2120	12	40	7200	53.9	299.7	20	1060	12	40	7200	48.5	1088.2	12	636	12	40	7200	34.9	1901.9
40	2	3.0	4	40	2120	14	40	7200	57.9	242	20	1060	13	40	7200	56.8	1085	13	689	13	40	7200	43.1	1732.8

Continued on Next Page...

Table A.3 – Continued

Instances				Default							Improved Bounds							Minimum						
m	C	ρ	#	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
40	2	3.0	5	40	2120	11	39	7200	60.7	112.1	20	1060	11	39	7200	56.3	853.9	11	583	11	39	7200	46.7	1875.2
40	2	4.5	1	40	1920	11	35	7200	60.4	284.2	20	960	11	35	7200	49.7	1354.1	11	528	11	35	7200	33.5	2552.8
40	2	4.5	2	40	1920	12	36	7200	54.9	287.3	20	960	12	36	7200	49.6	1196.7	11	528	11	36	7200	31	2399
40	2	4.5	3	40	1920	11	34	7200	53	243.8	20	960	11	34	7200	45.5	1398.4	11	528	11	34	7200	31.6	2683.3
40	2	4.5	4	40	1920	11	35	7200	51	419.6	20	960	11	35	7200	42.7	1653.5	11	528	11	35	7200	27.6	3305.1
40	2	4.5	5	40	1920	11	35	7200	57.1	393.3	20	960	11	35	7200	48.2	1270.9	11	528	11	35	7200	32.5	2179.4
45	2	1.5	1	45	3375	12	50	7200	40.4	141.1	23	1725	11	49	7200	35.1	621.4	11	825	11	49	7200	25.2	1574.7
45	2	1.5	2	45	3375	14	53	7200	43.7	112.6	23	1725	14	53	7200	39.9	565.8	13	975	13	53	7200	32.7	1247
45	2	1.5	3	45	3375	16	55	7200	41.1	119.2	23	1725	16	55	7200	36.6	668.8	13	975	13	56	7200	32.9	1295.6
45	2	1.5	4	45	3375	15	55	7200	44.9	92.3	23	1725	15	54	7200	40.4	564.1	13	975	13	54	7200	33	1109
45	2	1.5	5	45	3375	13	53	7200	41.4	122	23	1725	14	54	7200	37.8	651.6	12	900	12	53	7200	28.4	1397.9
45	2	3.0	1	45	2700	13	44	7200	60.5	69.8	23	1380	13	45	7200	56.9	617.4	13	780	13	45	7200	46.5	1330.9
45	2	3.0	2	45	2700	15	47	7200	59.8	129.4	23	1380	15	47	7200	54.7	570.4	15	900	15	47	7200	45.6	957.8
45	2	3.0	3	45	2700	15	47	7200	62.4	135.8	23	1380	15	46	7200	58.4	564.4	15	900	15	46	7200	48.8	1060.2
45	2	3.0	4	45	2700	12	42	7200	59.7	160.4	23	1380	12	42	7200	55.8	574.5	11	660	11	43	7200	42.1	1236.6
45	2	3.0	5	45	2700	13	45	7200	61.6	123.1	23	1380	13	45	7200	59.4	530.8	13	780	13	44	7200	51.8	994.2
45	2	4.5	1	45	2475	14	42	7200	59.1	162.2	23	1265	14	42	7200	55.7	771	14	770	14	42	7200	44.9	1423.3
45	2	4.5	2	45	2475	15	42	7200	56.5	194.8	23	1265	15	42	7200	53.7	977.4	15	825	15	42	7200	40.4	1584.6
45	2	4.5	3	45	2475	14	42	7200	59.5	171	23	1265	14	42	7200	54.7	738.6	13	715	13	42	7200	39.7	1428
45	2	4.5	4	45	2475	17	45	7200	54.5	213.4	23	1265	17	45	7200	49.9	781.8	17	935	17	45	7200	41.6	1176.6
45	2	4.5	5	45	2475	12	39	7200	60.8	136.2	23	1265	12	39	7200	52.5	749.3	12	660	12	39	7200	43.3	1658.4
50	2	1.5	1	50	4150	17	63	7200	43.6	59.5	25	2075	17	63	7200	39	430.4	16	1328	16	62	7200	29.3	683.8
50	2	1.5	2	50	4150	15	60	7200	43.7	62.8	25	2075	15	59	7200.2	40.3	358.9	14	1162	14	59	7200	34	1009.6
50	2	1.5	3	50	4150	14	58	7200	43.7	97.2	25	2075	13	58	7200	40.4	407.6	12	996	12	58	7200	31.6	1082
50	2	1.5	4	50	4150	16	62	7200	47.2	36.8	25	2075	15	60	7200	41.3	341	15	1245	15	60	7200	36.8	988.2
50	2	1.5	5	50	4150	15	61	7200	45.5	83.7	25	2075	15	60	7200	42.6	334.2	14	1162	14	60	7200	37	1014
50	2	3.0	1	50	3300	15	50	7200	60.2	75.1	25	1650	15	50	7200	55.2	402.1	15	990	15	50	7200	46.8	899
50	2	3.0	2	50	3300	19	55	7200	62.7	78	25	1650	19	54	7200	63.7	357.5	19	1254	19	55	7200	55.5	588.2
50	2	3.0	3	50	3300	14	50	7200	64.3	66.4	25	1650	14	50	7200.1	61	404.9	14	924	14	49	7200	48.6	843.6
50	2	3.0	4	50	3300	16	49	7200	62.1	52.5	25	1650	16	50	7200	57.8	417.8	15	990	15	49	7200	48.8	891.9
50	2	3.0	5	50	3300	15	53	7200	65.6	37.1	25	1650	15	52	7200	61.8	296.8	14	924	14	52	7200	50.6	825.5
50	2	4.5	1	50	3050	14	47	7200	72.6	54.5	25	1525	15	47	7200.1	70.2	418.4	14	854	14	47	7200	57.1	963.4
50	2	4.5	2	50	3050	13	44	7200	64.9	86.9	25	1525	13	44	7200	55.8	587.8	13	793	13	44	7200	41.2	1162.5
50	2	4.5	3	50	3050	15	47	7200	65	114.7	25	1525	15	49	7200	61.1	447.3	15	915	15	47	7200	46.2	828.7
50	2	4.5	4	50	3050	18	49	7200	50.1	46.8	25	1525	17	49	7200	62.3	424.4	17	1037	17	49	7200	53.4	851.2
50	2	4.5	5	50	3050	16	48	7200	67.5	91.7	25	1525	15	48	7200	63.7	464.6	15	915	15	48	7200	56	887.3
55	2	1.5	1	55	5005	17	65	7200	42.8	47.4	28	2548	16	64	7200.1	39.3	228.7	14	1274	14	64	7200	34.2	805.8
55	2	1.5	2	55	5005	19	67	7200	42.9	59.8	28	2548	18	67	7200	44.7	232.3	17	1547	17	66	7200	37.3	572.8
55	2	1.5	3	55	5005	17	63	7200	41.6	40.9	28	2548	16	62	7200	38.8	251.8	16	1456	16	62	7200	33.3	774.4

Continued on Next Page...

Table A.3 – Continued

Instances				Default							Improved Bounds							Minimum						
m	C	ρ	#	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
55	2	1.5	4	55	5005	15	65	7200	44.4	43.8	28	2548	15	64	7200	40.3	241.5	14	1274	14	64	7200	35.4	824.3
55	2	1.5	5	55	5005	18	66	7200	47.2	52.3	28	2548	19	66	7200	45.9	239.2	16	1456	16	66	7200	41.6	587.9
55	2	3.0	1	55	4015	16	56	7200	63.7	29.2	28	2044	16	57	7200	57.7	214	16	1168	16	57	7200	54.8	517.8
55	2	3.0	2	55	4015	16	55	7200	63.9	33.9	28	2044	16	55	7200	61.3	270.5	16	1168	16	55	7200	57.6	528.9
55	2	3.0	3	55	4015	19	58	7200	65.2	22.9	28	2044	18	57	7200	62	219.5	18	1314	18	57	7200	53.6	506.9
55	2	3.0	4	55	4015	16	55	7200	65	37.3	28	2044	16	56	7200	62.1	209.4	15	1095	15	55	7200	57.6	485.4
55	2	3.0	5	55	4015	16	57	7200	66.4	38.9	28	2044	16	56	7200	61.8	231.1	15	1095	15	55	7200	56.1	569.3
55	2	4.5	1	55	3685	15	49	7200	73.3	27.7	28	1876	15	49	7200	68.5	276.1	15	1005	15	49	7200	63	886.3
55	2	4.5	2	55	3685	15	51	7200	72.4	29.7	28	1876	16	50	7200	54.8	348.2	15	1005	15	50	7200	43.9	752.4
55	2	4.5	3	55	3685	16	51	7200	70.3	57.3	28	1876	16	51	7200	57.7	345.1	16	1072	16	51	7200	51.6	834.6
55	2	4.5	4	55	3685	18	52	7200	73.2	36.9	28	1876	18	53	7200	70.4	246	17	1139	17	53	7200	52.2	363.7
55	2	4.5	5	55	3685	21	56	7200	69.4	52.6	28	1876	20	56	7200	71.7	269.7	20	1340	20	55	7200	68.9	431.5
60	2	1.5	1	60	6000	19	74	7200	45.2	17.2	30	3000	19	73	7200	42.7	162.7	19	1900	19	74	7200	38.2	436.3
60	2	1.5	2	60	6000	22	74	7200	45.1	37.8	30	3000	21	73	7200	40.8	199.8	19	1900	19	73	7200	35.9	492.8
60	2	1.5	3	60	6000	18	72	7200	45.8	24.4	30	3000	18	71	7200	43.7	165.4	17	1700	17	71	7200	36	495.4
60	2	1.5	4	60	6000	19	72	7200	42.6	34.5	30	3000	17	72	7200	44.5	141.9	16	1600	16	71	7200	34.9	506.9
60	2	1.5	5	60	6000	17	71	7200	43.6	27.3	30	3000	16	70	7200	39.6	148	16	1600	16	70	7200	32.2	470.3
60	2	3.0	1	60	4800	19	65	7200	66.1	22.5	30	2400	20	64	7200.1	62.1	168.9	18	1440	18	64	7200	61.7	282.8
60	2	3.0	2	60	4800	19	63	7200	65.9	16	30	2400	17	62	7200	56.6	183.1	17	1360	17	62	7200	51.8	415.7
60	2	3.0	3	60	4800	21	65	7200	64.9	14.8	30	2400	20	63	7200	60	167.4	20	1600	20	65	7200	51.6	274.5
60	2	3.0	4	60	4800	19	64	7200	65.8	32.8	30	2400	18	62	7200	57.3	162.1	18	1440	18	62	7200	55	346.6
60	2	3.0	5	60	4800	17	64	7200	66.8	21.3	30	2400	18	63	7200	53.7	175	16	1280	16	62	7200	49.6	392.3
60	2	4.5	1	60	4380	19	59	7200	76.4	23	30	2190	18	58	7200	71.6	177.5	16	1168	16	57	7200	68.1	434.4
60	2	4.5	2	60	4380	17	57	7200	72.9	17.4	30	2190	17	58	7200	63	234.2	17	1241	17	56	7200	56.3	430.2
60	2	4.5	3	60	4380	19	58	7200	72.1	25.4	30	2190	19	58	7200	64.7	193.5	19	1387	19	57	7200	57.5	491.9
60	2	4.5	4	60	4380	20	59	7200	73.3	18.1	30	2190	20	59	7200	67.4	218	19	1387	19	58	7200	58.5	444.2
60	2	4.5	5	60	4380	18	57	7200	72.9	28	30	2190	18	57	7200	66.9	188	18	1314	18	57	7200	59.5	424.2
30	5	1.5	1	30	1500	4	25	7200	22.1	1318.6	10	500	4	25	1592.9	0	1253.6	4	200	4	25	4.8	0	9.8
30	5	1.5	2	30	1500	4	27	7200	17	1255.4	10	500	4	27	215.5	0	156	4	200	4	27	3.2	0	6.2
30	5	1.5	3	30	1500	5	27	7200	16.3	1222.2	10	500	5	27	297.8	0	176.4	5	250	5	27	3.7	0	4.3
30	5	1.5	4	30	1500	4	27	7200	20.6	1108.9	10	500	4	27	1005.3	0	737	4	200	4	27	5.7	0	10
30	5	1.5	5	30	1500	4	28	7200	20.7	1138.5	10	500	4	28	840.5	0	638.4	4	200	4	28	5.2	0	9.9
30	5	3.0	1	30	1200	4	19	7200	32.4	1040.6	10	400	5	19	4829.2	0	3825.6	4	160	4	19	8.2	0	17.3
30	5	3.0	2	30	1200	4	19	7200	34.2	865.3	10	400	4	19	5421.4	0	4237.2	4	160	4	19	18.4	0	37.7
30	5	3.0	3	30	1200	4	18	7200	28.8	1064.4	10	400	4	18	729.8	0	507.3	4	160	4	18	6.4	0	13.1
30	5	3.0	4	30	1200	4	18	7200	29	1003.8	10	400	4	18	1008.3	0	771	4	160	4	18	3.1	0	6.9
30	5	3.0	5	30	1200	4	19	7200	33	1080.7	10	400	4	19	5737	0	4416.9	4	160	4	19	6.5	0	16.7
30	5	4.5	1	30	1080	4	14	7200	27.3	1282.2	10	360	4	14	130.7	0	140.3	4	144	4	14	1	0	1.2
30	5	4.5	2	30	1080	4	13	7200	25.4	1151.3	10	360	4	13	88.3	0	78.6	4	144	4	13	0.9	0	1.2

Continued on Next Page...

Table A.3 – Continued

Instances				Default							Improved Bounds							Minimum						
m	C	ρ	#	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
30	5	4.5	3	30	1080	4	13	7200	21.2	1078.9	10	360	4	13	47.2	0	46.9	4	144	4	13	1.2	0	1.5
30	5	4.5	4	30	1080	4	14	7200	30.6	1500.5	10	360	4	14	2424.1	0	2324.6	4	144	4	14	9	0	20.5
30	5	4.5	5	30	1080	3	13	7200	23.7	1222.3	10	360	3	13	130.9	0	93.5	3	108	3	13	0.4	0	0.2
35	5	1.5	1	35	2030	5	31	7200	24.6	627.6	12	696	5	31	7200	11.4	2601.6	5	290	5	31	111.5	0	135.7
35	5	1.5	2	35	2030	4	31	7200	19	711.2	12	696	4	31	4537.8	0	2411.5	4	232	4	31	9.9	0	15.8
35	5	1.5	3	35	2030	4	29	7200	20	712.2	12	696	4	29	3824.3	0	1644.2	4	232	4	29	6	0	6.5
35	5	1.5	4	35	2030	5	33	7200	23.7	803.9	12	696	5	33	7200	10.7	2916.5	5	290	5	33	136.6	0	173.6
35	5	1.5	5	35	2030	5	31	7200	22.2	828.1	12	696	5	31	7200	8.1	2998.8	5	290	5	31	84.6	0	111.2
35	5	3.0	1	35	1610	4	22	7200	39.8	588.6	12	552	4	22	7200	21.6	2429.9	4	184	4	22	79.9	0	131.5
35	5	3.0	2	35	1610	5	21	7200	35.8	760.5	12	552	5	21	7200	16.8	2502.6	5	230	5	21	91.2	0	134.2
35	5	3.0	3	35	1610	5	23	7200	40.6	537.6	12	552	5	23	7200	25.6	2703.3	5	230	5	23	489.3	0	672.8
35	5	3.0	4	35	1610	5	22	7200	39.1	627.9	12	552	5	22	7200	20.6	2108.6	5	230	5	22	313.2	0	459.7
35	5	3.0	5	35	1610	4	21	7200	35.7	574.4	12	552	4	21	7200	18.8	2521.4	4	184	4	21	71	0	127.5
35	5	4.5	1	35	1470	6	18	7200	39.6	717	12	504	6	18	7200	19.2	2411.9	6	252	6	18	837.3	0	1096
35	5	4.5	2	35	1470	4	16	7200	36.4	614.5	12	504	4	16	7200	17.9	2464.7	4	168	4	16	11.4	0	23.3
35	5	4.5	3	35	1470	4	17	7200	40.8	547.3	12	504	4	17	7200	15	2527	4	168	4	17	27.5	0	68.8
35	5	4.5	4	35	1470	5	16	7200	33.2	689.2	12	504	5	16	7200	13.6	3182.3	5	210	5	16	29.8	0	38
35	5	4.5	5	35	1470	5	17	7200	36.2	687.2	12	504	5	17	7200	16	2618.9	5	210	5	17	33.1	0	39
40	5	1.5	1	40	2640	5	37	7200	25	572.7	14	924	5	37	7200	14.6	2056.6	4	264	4	37	105.2	0	126.1
40	5	1.5	2	40	2640	4	36	7200	23.6	461.3	14	924	4	36	7200	12.3	1877.7	4	264	4	36	65.4	0	74.5
40	5	1.5	3	40	2640	5	37	7200	25.6	441.1	14	924	5	37	7200	14.4	1650.1	5	330	5	37	279	0	244.3
40	5	1.5	4	40	2640	5	37	7200	25.6	503.8	14	924	5	37	7200	15.1	1985	5	330	5	37	725.5	0	804.6
40	5	1.5	5	40	2640	4	33	7200	19.1	552.2	14	924	4	33	7200	6.6	1627.1	4	264	4	33	11.4	0	14.7
40	5	3.0	1	40	2120	5	26	7200	38.4	361.8	14	742	5	26	7200	25.9	1389.8	5	265	5	26	5091.3	0	5092.7
40	5	3.0	2	40	2120	5	25	7200	40.9	366	14	742	5	25	7200	26.4	1493.2	5	265	5	25	879	0	1018.1
40	5	3.0	3	40	2120	6	25	7200	39.9	331.4	14	742	6	25	7200	25.3	1502.8	6	318	6	25	3607	0	3013.7
40	5	3.0	4	40	2120	5	25	7200	40	363.8	14	742	5	25	7200	25.3	1322.1	5	265	5	25	1132.5	0	1140.3
40	5	3.0	5	40	2120	5	26	7200	41.5	325.5	14	742	5	26	7200	28.8	1393.3	5	265	5	26	3247.8	0	3434.5
40	5	4.5	1	40	1920	5	19	7200	43.6	348.9	14	672	5	19	7200	26.4	1409.6	5	240	5	19	342.2	0	369.9
40	5	4.5	2	40	1920	5	20	7200	50.5	324.1	14	672	5	20	7200	31.5	1727.5	5	240	5	20	969.3	0	1574.3
40	5	4.5	3	40	1920	4	18	7200	36.4	512.2	14	672	4	18	7200	17.6	2002.7	4	192	4	18	5.7	0	7.2
40	5	4.5	4	40	1920	6	20	7200	47.9	475.7	14	672	6	20	7200	32.6	1838.1	6	288	6	20	5850.7	0	6677.8
40	5	4.5	5	40	1920	5	20	7200	46.3	371.5	14	672	5	20	7200	29.7	1571.1	5	240	5	20	856.9	0	1028.1
45	5	1.5	1	45	3375	5	40	7200	23.8	337.9	15	1125	5	40	7200	15	1405.2	4	300	4	41	162.1	0	159.2
45	5	1.5	2	45	3375	7	41	7200	22.7	377.9	15	1125	7	41	7200	11.5	1046.4	7	525	7	41	7200	4.6	3377
45	5	1.5	3	45	3375	7	41	7200	28	204.4	15	1125	6	41	7200	16.8	913.2	6	450	6	41	7200	3.6	4233.2
45	5	1.5	4	45	3375	7	42	7200	24.7	284.5	15	1125	7	42	7200	16	1355.3	6	450	6	42	7200	5.1	3913.3
45	5	1.5	5	45	3375	7	43	7200	31.3	218.4	15	1125	7	43	7200	22.7	1228	7	525	7	43	7200	11.8	2784.8
45	5	3.0	1	45	2700	6	31	7200	46.8	198.1	15	900	6	31	7200	34.1	819.2	6	360	6	30	7200	13.4	3070.4

Continued on Next Page...

Table A.3 – Continued

Instances				Default							Improved Bounds							Minimum						
m	C	ρ	#	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
45	5	3.0	2	45	2700	7	31	7200	46.3	206.5	15	900	6	31	7200	35	1037.5	6	360	6	31	7200	16	3272.5
45	5	3.0	3	45	2700	6	30	7200	44	183.9	15	900	6	30	7200	32.6	970.5	6	360	6	30	7200	13.9	2904.2
45	5	3.0	4	45	2700	6	32	7200	48.4	186.4	15	900	6	31	7200	31.8	843.7	6	360	6	31	7200	15.6	3518.2
45	5	3.0	5	45	2700	6	31	7200	46.2	240.5	15	900	6	31	7200	35.9	1117.7	6	360	6	31	7200	15.4	3085.5
45	5	4.5	1	45	2475	6	25	7200	51.3	201.9	15	825	6	25	7200	40.4	1253.8	6	330	6	25	7200	16.4	3669.9
45	5	4.5	2	45	2475	5	23	7200	49.6	275.7	15	825	6	23	7200	35.6	1289.2	5	275	5	23	1600.9	0	1616.3
45	5	4.5	3	45	2475	6	24	7200	48.9	149	15	825	6	24	7200	33.2	992.6	6	330	6	24	7200	11.4	3994.6
45	5	4.5	4	45	2475	6	25	7200	53.4	286.2	15	825	6	25	7200	41.3	1310.7	6	330	6	25	7200	18.6	3734.8
45	5	4.5	5	45	2475	6	24	7200	49	205.7	15	825	6	24	7200	35.9	1090.6	6	330	6	24	7200	12	4235.2
50	5	1.5	1	50	4150	7	46	7200	28.3	192.6	17	1411	7	46	7200	20.8	994.4	6	498	6	46	7200	7.4	3294.2
50	5	1.5	2	50	4150	6	47	7200	27.8	146.1	17	1411	6	46	7200	18.4	956.1	5	415	5	46	6242.1	0	4411.1
50	5	1.5	3	50	4150	6	46	7200	26.1	169	17	1411	6	46	7200	18.2	902.7	5	415	5	46	3914.3	0	1975.6
50	5	1.5	4	50	4150	7	48	7200	33.4	120.8	17	1411	7	48	7200	26	711.7	7	581	7	48	7200	15.5	2025
50	5	1.5	5	50	4150	6	48	7200	31.5	133.5	17	1411	6	48	7200	21.9	585.8	6	498	6	48	7200	12	2739.1
50	5	3.0	1	50	3300	6	31	7200	44.2	146.6	17	1122	6	31	7200	34.6	706.9	6	396	6	31	7200	13.9	2670.7
50	5	3.0	2	50	3300	7	35	7200	51.4	114	17	1122	7	37	7200	44.7	653	7	462	7	37	7200	28.9	2053
50	5	3.0	3	50	3300	6	34	7200	50	116	17	1122	6	34	7200	40.4	719.7	6	396	6	34	7200	23	2492.1
50	5	3.0	4	50	3300	7	34	7200	49.2	116.2	17	1122	7	34	7200	37.7	710	7	462	7	35	7200.1	24.6	2400.1
50	5	3.0	5	50	3300	6	30	7200	42.4	156.7	17	1122	6	30	7200	31.9	785.2	6	396	6	30	7200	11.2	2916.3
50	5	4.5	1	50	3050	7	29	7200	56.7	117	17	1037	7	29	7200	45	774.3	7	427	7	29	7200	27.6	2327.4
50	5	4.5	2	50	3050	8	30	7200	56.5	157	17	1037	8	30	7200	44.6	732.3	8	488	8	30	7200	32.4	1801.5
50	5	4.5	3	50	3050	6	27	7200	54.1	147	17	1037	6	27	7200	43.2	834.6	6	366	6	27	7200	21.6	2970.7
50	5	4.5	4	50	3050	6	26	7200	52.4	144	17	1037	6	26	7200	41.3	730.1	6	366	6	26	7200	17.5	3107.7
50	5	4.5	5	50	3050	7	29	7200	57.5	148.4	17	1037	8	30	7200	49.2	859.6	7	427	7	29	7200	28	2499.7
55	5	1.5	1	55	5005	8	52	7200	31.1	69.2	19	1729	7	51	7200	23.3	614.2	7	637	7	51	7200	13.8	1933.8
55	5	1.5	2	55	5005	7	51	7200	29.7	58.9	19	1729	8	51	7200	23.5	490.6	7	637	7	51	7200	11.8	1508.7
55	5	1.5	3	55	5005	9	50	7200	26.3	116.8	19	1729	9	50	7200	18	552	8	728	8	50	7200	10.2	1418.1
55	5	1.5	4	55	5005	7	50	7200	32.3	67.7	19	1729	7	50	7200	23.1	380.1	7	637	7	50	7200	15.7	1882.3
55	5	1.5	5	55	5005	8	52	7200	29.1	103.2	19	1729	8	52	7200	21.4	468.3	7	637	7	52	7200	13.5	1986.2
55	5	3.0	1	55	4015	7	39	7200	51.7	61.2	19	1387	7	39	7200	43.5	449.4	7	511	7	39	7200	28.5	1650.9
55	5	3.0	2	55	4015	6	36	7200	47.7	70.3	19	1387	6	37	7200	41	534.3	6	438	6	36	7200	17.9	2564.4
55	5	3.0	3	55	4015	6	38	7200	50.4	73.7	19	1387	7	39	7200	43	409.7	6	438	6	39	7200	27.5	2171.2
55	5	3.0	4	55	4015	6	34	7200	44.6	83.1	19	1387	7	36	7200	39.5	536	6	438	6	34	7200	15.3	2506.9
55	5	3.0	5	55	4015	7	36	7200	47.6	84.7	19	1387	7	36	7200	39.8	584.9	7	511	7	36	7200	22.8	1850.4
55	5	4.5	1	55	3685	7	31	7200	58.6	79.8	19	1273	6	30	7200	47.5	617.5	6	402	6	30	7200	26.5	2353.6
55	5	4.5	2	55	3685	7	30	7200	56.7	92	19	1273	7	30	7200	44.1	446.3	7	469	7	30	7200	26.2	1888.7
55	5	4.5	3	55	3685	7	32	7200	59.4	61	19	1273	7	32	7200	47	510.4	7	469	7	31	7200	26.6	2032.3
55	5	4.5	4	55	3685	7	31	7200	58.6	102.1	19	1273	7	31	7200	49.6	587.8	7	469	7	30	7200	28.2	2008.5
55	5	4.5	5	55	3685	8	33	7200	61	80	19	1273	8	33	7200	52.5	587.1	7	469	7	32	7200	30.1	2133.5

Continued on Next Page...

Table A.3 – Continued

Instances				Default							Improved Bounds							Minimum						
m	C	ρ	#	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
60	5	1.5	1	60	6000	6	50	7200	26.4	66.3	20	2000	6	50	7200	21.1	370.2	5	500	5	50	7200	7.1	2715.8
60	5	1.5	2	60	6000	8	54	7200	28.1	53.7	20	2000	8	54	7200	23.5	484	6	600	6	53	7200	9.9	2028.6
60	5	1.5	3	60	6000	9	57	7200	31.9	54.8	20	2000	7	56	7200	25.2	489.6	7	700	7	56	7200	15.5	1542
60	5	1.5	4	60	6000	7	56	7200	32.4	59.4	20	2000	7	55	7200	25.3	370.2	7	700	7	55	7200	16	1654.6
60	5	1.5	5	60	6000	9	58	7200	33	67.9	20	2000	9	58	7200	26.6	310.1	8	800	8	58	7200	18.3	1320.3
60	5	3.0	1	60	4800	7	43	7200	51.5	43.2	20	1600	7	42	7200	44	326.9	7	560	7	42	7200	28.6	1261.6
60	5	3.0	2	60	4800	7	42	7200	50.4	45.4	20	1600	7	41	7200	40.4	232.5	7	560	7	41	7200	26.6	1559.5
60	5	3.0	3	60	4800	8	41	7200	49.1	33.9	20	1600	7	41	7200	43	361.4	7	560	7	41	7200	27.4	1206.4
60	5	3.0	4	60	4800	7	39	7200	46.5	52.4	20	1600	7	39	7200	37.7	343.2	7	560	7	38	7200	19.4	1459.4
60	5	3.0	5	60	4800	8	44	7200	52.6	46.8	20	1600	8	42	7200	43.8	339.9	7	560	7	43	7200	31	1483.8
60	5	4.5	1	60	4380	6	32	7200	56.8	57.3	20	1460	6	31	7200	45.6	437.7	6	438	6	31	7200	24.5	1880.9
60	5	4.5	2	60	4380	8	35	7200	60.4	43.9	20	1460	7	34	7200	46	323.4	7	511	7	35	7200	32.4	1402
60	5	4.5	3	60	4380	8	33	7200	58	48.5	20	1460	8	34	7200	47.1	316.5	8	584	8	33	7200	32.7	1196.6
60	5	4.5	4	60	4380	7	34	7200	59.3	44.9	20	1460	7	34	7200	49.5	440.4	7	511	7	35	7200	34.9	1655.3
60	5	4.5	5	60	4380	6	33	7200	58.1	48.3	20	1460	6	34	7200	47.3	285.2	6	438	6	33	7200	26.4	1555.3
30	8	1.5	1	30	1500	3	24	2351.4	0	538	6	300	3	24	1	0	0.8	3	150	3	24	0.3	0	0.1
30	8	1.5	2	30	1500	3	23	3311.8	0	569.7	6	300	3	23	1.3	0	0.6	3	150	3	23	0.3	0	0.1
30	8	1.5	3	30	1500	3	24	7200	11.7	1037.3	6	300	3	24	2.2	0	2	3	150	3	24	0.5	0	0.3
30	8	1.5	4	30	1500	3	23	7200	10.8	1200.1	6	300	3	23	3	0	3.4	3	150	3	23	0.3	0	0.3
30	8	1.5	5	30	1500	3	24	3134.4	0	494.7	6	300	3	24	1.1	0	0.9	3	150	3	24	0.4	0	0.1
30	8	3.0	1	30	1200	3	16	7200	22.5	982.9	6	240	3	16	7.9	0	8.6	3	120	3	16	0.6	0	0.9
30	8	3.0	2	30	1200	2	13	233.4	0	35.3	6	240	2	13	0.8	0	0.2	2	80	2	13	0	0	0
30	8	3.0	3	30	1200	2	14	1955.5	0	357	6	240	2	14	1.1	0	0.7	2	80	2	14	0.1	0	0
30	8	3.0	4	30	1200	2	15	7200	16.4	1131	6	240	2	15	1.9	0	1.8	2	80	2	15	0	0	0
30	8	3.0	5	30	1200	3	15	7200	16.5	1097.6	6	240	3	15	2.5	0	3.2	3	120	3	15	0.3	0	0.5
30	8	4.5	1	30	1080	3	11	7200	21.6	970.5	6	216	3	11	3.3	0	3.7	3	108	3	11	0.3	0	0.2
30	8	4.5	2	30	1080	3	11	7200	10.3	1291.2	6	216	3	11	4.1	0	8.1	3	108	3	11	0.1	0	0.1
30	8	4.5	3	30	1080	3	12	7200	24.3	1346.4	6	216	3	12	22.4	0	39.3	3	108	3	12	0.3	0	0.5
30	8	4.5	4	30	1080	2	9	43.2	0	6	6	216	2	9	0.4	0	0.1	2	72	2	9	0	0	0
30	8	4.5	5	30	1080	3	12	7200	23.8	1356.8	6	216	3	12	14	0	21	3	108	3	12	0.3	0	0.5
35	8	1.5	1	35	2030	3	29	7200	14	701.3	7	406	3	29	16.3	0	13.7	3	174	3	29	0.7	0	1
35	8	1.5	2	35	2030	3	25	7200	4.7	821.8	7	406	3	25	1.6	0	0.7	3	174	3	25	0.3	0	0.1
35	8	1.5	3	35	2030	3	29	7200	14.7	638.2	7	406	3	29	21	0	15.2	3	174	3	29	0.6	0	0.6
35	8	1.5	4	35	2030	3	27	7200	9.9	744.5	7	406	3	27	4.4	0	2.5	3	174	3	27	0.3	0	0.1
35	8	1.5	5	35	2030	4	28	7200	14.1	920.7	7	406	4	28	25.1	0	18.9	4	232	4	28	1.7	0	2.5
35	8	3.0	1	35	1610	3	18	7200	27.8	665.1	7	322	3	18	70.9	0	65.6	3	138	3	18	0.7	0	1.3
35	8	3.0	2	35	1610	3	18	7200	26.5	949.5	7	322	3	18	82.7	0	80.6	3	138	3	18	0.7	0	1.5
35	8	3.0	3	35	1610	2	14	201.5	0	34.4	7	322	2	14	0.9	0	0.1	2	92	2	14	0	0	0
35	8	3.0	4	35	1610	3	17	7200	24.1	686.2	7	322	3	17	22.4	0	18.5	3	138	3	17	0.8	0	1

Continued on Next Page...

Table A.3 – Continued

Instances				Default							Improved Bounds							Minimum						
m	C	ρ	#	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
35	8	3.0	5	35	1610	3	18	7200	28.9	707.9	7	322	3	18	70	0	69.2	3	138	3	18	0.8	0	1.7
35	8	4.5	1	35	1470	3	12	7200	21.7	732	7	294	3	12	7.4	0	6.8	3	126	3	12	0.5	0	0.1
35	8	4.5	2	35	1470	3	13	7200	28.7	800.3	7	294	3	13	16.6	0	15.6	3	126	3	13	0.3	0	0.5
35	8	4.5	3	35	1470	3	14	7200	32.4	803.5	7	294	3	14	43.8	0	45.1	3	126	3	14	0.9	0	1.4
35	8	4.5	4	35	1470	3	12	7200	22	711.5	7	294	3	12	5.5	0	4.5	3	126	3	12	0.4	0	0.1
35	8	4.5	5	35	1470	3	13	7200	29.6	842.4	7	294	3	13	39.3	0	46.5	3	126	3	13	0.4	0	0.8
40	8	1.5	1	40	2640	3	32	7200	17	394.5	8	528	3	32	89.3	0	48.6	3	198	3	32	1.1	0	1.2
40	8	1.5	2	40	2640	3	31	7200	14	504.9	8	528	3	31	41.4	0	17.9	3	198	3	31	0.4	0	0.2
40	8	1.5	3	40	2640	4	31	7200	21.2	457.5	8	528	4	31	495.9	0	285.3	4	264	4	31	9.5	0	13.4
40	8	1.5	4	40	2640	4	34	7200	18.6	513.5	8	528	4	34	504.1	0	322.8	4	264	4	34	9.3	0	14.1
40	8	1.5	5	40	2640	3	32	7200	13.7	451.2	8	528	3	32	33.2	0	17.1	3	198	3	32	0.7	0	0.7
40	8	3.0	1	40	2120	3	22	7200	34.1	402.4	8	424	3	22	2427.8	0	1477.3	3	159	3	22	2.9	0	7.8
40	8	3.0	2	40	2120	3	20	7200	25.9	396.9	8	424	3	20	144.4	0	78.6	3	159	3	20	0.7	0	1
40	8	3.0	3	40	2120	4	23	7200	37	386.8	8	424	4	23	6945.3	0	3761.7	4	212	4	23	61.5	0	107.8
40	8	3.0	4	40	2120	4	22	7200	33.3	379.1	8	424	4	22	2024.7	0	1208.4	4	212	4	22	25.4	0	30.6
40	8	3.0	5	40	2120	4	22	7200	33.2	390.3	8	424	4	22	1728	0	938.6	4	212	4	22	19.1	0	22.4
40	8	4.5	1	40	1920	3	14	7200	27.4	432.8	8	384	3	14	61.4	0	29.3	3	144	3	14	0.6	0	0.6
40	8	4.5	2	40	1920	4	17	7200	42.7	476.4	8	384	4	17	3985	0	3169.9	4	192	4	17	29.3	0	69.5
40	8	4.5	3	40	1920	3	14	7200	31.4	483.8	8	384	3	14	48.1	0	30.3	3	144	3	14	0.6	0	0.9
40	8	4.5	4	40	1920	3	14	7200	31.1	405.7	8	384	3	14	38.1	0	17.8	3	144	3	14	0.5	0	0.7
40	8	4.5	5	40	1920	4	17	7200	40.8	447.9	8	384	4	17	4824.6	0	3363.7	4	192	4	17	18.8	0	42.3
45	8	1.5	1	45	3375	3	38	7200	17.6	259.2	9	675	3	38	1327.2	0	511.3	3	225	3	38	2.4	0	3.8
45	8	1.5	2	45	3375	4	38	7200	17.5	225.6	9	675	4	38	811.3	0	289.8	4	300	4	38	12.1	0	15.5
45	8	1.5	3	45	3375	4	36	7200	21.2	289.7	9	675	4	36	6061.1	0	2214.1	4	300	4	36	31.2	0	39.7
45	8	1.5	4	45	3375	3	36	7200	18.6	263.7	9	675	3	36	1162.9	0	441.6	3	225	3	36	2.9	0	4
45	8	1.5	5	45	3375	3	36	7200	15.9	366.1	9	675	3	36	506.7	0	209.4	3	225	3	36	1.5	0	2.4
45	8	3.0	1	45	2700	4	25	7200	34.1	276.9	9	540	4	25	7200	11	2590.8	4	240	4	25	54	0	66.7
45	8	3.0	2	45	2700	4	26	7200	37	251.4	9	540	4	26	7200	13.9	2578.7	4	240	4	26	377.2	0	515.4
45	8	3.0	3	45	2700	3	24	7200	32.2	244.2	9	540	3	24	4220	0	1562.2	3	180	3	24	5.5	0	11.9
45	8	3.0	4	45	2700	4	25	7200	34.3	279.3	9	540	4	25	7200	9.1	2356.5	4	240	4	25	65.4	0	78.3
45	8	3.0	5	45	2700	4	24	7200	31.4	280.9	9	540	4	24	4432	0	1462.2	4	240	4	24	16.9	0	17.9
45	8	4.5	1	45	2475	4	19	7200	38.9	290.6	9	495	4	19	4574.2	0	1789.4	4	220	4	19	12.9	0	18.1
45	8	4.5	2	45	2475	3	18	7200	35.5	262.3	9	495	3	18	1626.9	0	591.5	3	165	3	18	2.3	0	3.5
45	8	4.5	3	45	2475	3	18	7200	36.6	231.7	9	495	3	18	1747.3	0	610.9	3	165	3	18	2.3	0	4.1
45	8	4.5	4	45	2475	4	21	7200	45.7	321.5	9	495	4	21	7200	19.7	2815.5	4	220	4	21	335.1	0	579.7
45	8	4.5	5	45	2475	4	20	7200	42.3	299.4	9	495	4	20	7200	14.7	2461.8	4	220	4	20	49.8	0	81
50	8	1.5	1	50	4150	4	41	7200	19.3	193.7	10	830	4	41	7200	5.1	1691.2	4	332	4	41	19.3	0	16.7
50	8	1.5	2	50	4150	4	38	7200	18.4	191.7	10	830	4	38	2684.2	0	738.2	4	332	4	38	21.5	0	18.1
50	8	1.5	3	50	4150	4	39	7200	20.5	176.2	10	830	4	39	7200	5.9	1653.1	4	332	4	39	33.5	0	34.7

Continued on Next Page...

Table A.3 – Continued

Instances				Default							Improved Bounds							Minimum						
m	C	ρ	#	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
50	8	1.5	4	50	4150	4	41	7200	22	161.5	10	830	4	41	7200	8	1668	4	332	4	41	40.4	0	31.7
50	8	1.5	5	50	4150	5	42	7200	21.4	161.1	10	830	5	42	7200	7.6	1575.8	5	415	5	42	213.5	0	158.5
50	8	3.0	1	50	3300	4	28	7200	39.3	155.6	10	660	4	28	7200	19.9	1553.1	4	264	4	28	501	0	667.1
50	8	3.0	2	50	3300	4	28	7200	38.9	170.7	10	660	4	28	7200	18.7	1687.2	4	264	4	28	456.3	0	579.8
50	8	3.0	3	50	3300	4	28	7200	39.3	170.6	10	660	4	28	7200	19.8	1720.9	4	264	4	28	197.5	0	258.8
50	8	3.0	4	50	3300	4	28	7200	39.3	163.1	10	660	4	28	7200	20	1624.3	4	264	4	28	715.5	0	718.8
50	8	3.0	5	50	3300	4	29	7200	41.2	152	10	660	4	29	7200	22.8	1614.9	4	264	4	29	691.3	0	1014.7
50	8	4.5	1	50	3050	4	22	7200	43.8	179.5	10	610	4	22	7200	20.6	1554.3	4	244	4	22	212.3	0	215.9
50	8	4.5	2	50	3050	5	22	7200	42.9	210.4	10	610	5	22	7200	19.1	1905.6	5	305	5	22	362.2	0	306.1
50	8	4.5	3	50	3050	4	22	7200	43.7	208	10	610	4	22	7200	22.2	2112	4	244	4	22	325.6	0	448.4
50	8	4.5	4	50	3050	4	22	7200	43.3	179.7	10	610	4	22	7200	20.7	1456.6	4	244	4	22	142	0	169.4
50	8	4.5	5	50	3050	5	23	7200	45.5	168.3	10	610	5	23	7200	22.6	1556.4	5	305	5	23	2093.6	0	1853.7
55	8	1.5	1	55	5005	5	44	7200	21.1	99.9	11	1001	4	44	7200	9.1	1261.1	4	364	4	44	131.8	0	128.3
55	8	1.5	2	55	5005	5	45	7200	20.4	149.7	11	1001	5	45	7200	9	1369.6	4	364	4	46	266.8	0	253.2
55	8	1.5	3	55	5005	4	44	7200	18.9	105	11	1001	4	44	7200	6.6	1487.6	4	364	4	44	57.9	0	50.9
55	8	1.5	4	55	5005	4	46	7200	20.2	94.4	11	1001	5	46	7200	8.6	1443.1	4	364	4	46	107.5	0	102
55	8	1.5	5	55	5005	5	47	7200	26	64.5	11	1001	5	47	7200	13.6	946.4	5	455	5	47	7200	5.2	3525.7
55	8	3.0	1	55	4015	5	32	7200	41.4	99.7	11	803	5	32	7200	25.8	1092.3	5	365	5	31	7200	5.2	4197.1
55	8	3.0	2	55	4015	4	31	7200	39.6	97.7	11	803	4	30	7200	20.3	1045	4	292	4	30	385.7	0	331.8
55	8	3.0	3	55	4015	4	31	7200	39.5	86.8	11	803	4	31	7200	22.9	1016.9	4	292	4	31	768.1	0	748.8
55	8	3.0	4	55	4015	5	33	7200	43.2	96	11	803	5	33	7200	27.3	1031.9	5	365	5	33	7200	11.5	4042.6
55	8	3.0	5	55	4015	5	34	7200	44.8	75.6	11	803	5	34	7200	29.9	949.4	5	365	5	34	7200	15.7	3681.4
55	8	4.5	1	55	3685	4	24	7200	45.8	130.5	11	737	4	24	7200	26.6	1171.6	4	268	4	24	250.7	0	260.4
55	8	4.5	2	55	3685	4	23	7200	43.5	119.1	11	737	4	23	7200	22.7	1074.8	4	268	4	23	340.5	0	325.8
55	8	4.5	3	55	3685	5	26	7200	50	112.1	11	737	5	27	7200	33.8	1151.7	5	335	5	26	7200	12.8	4154
55	8	4.5	4	55	3685	4	23	7200	44.1	123	11	737	4	23	7200	22.4	1161.7	4	268	4	23	245.8	0	261.6
55	8	4.5	5	55	3685	5	25	7200	48.2	124	11	737	5	25	7200	28.4	1185.5	5	335	5	25	7200	6.9	4603.4
60	8	1.5	1	60	6000	5	51	7200	25.9	58.5	12	1200	5	51	7200	16.3	829	5	500	5	51	7200	5.2	3430
60	8	1.5	2	60	6000	5	50	7200	22.5	70.4	12	1200	5	50	7200	13.2	831	4	400	4	50	805.7	0	562
60	8	1.5	3	60	6000	5	53	7200	23.1	54.3	12	1200	5	52	7200	11.9	853.6	5	500	5	52	1598.8	0	811.6
60	8	1.5	4	60	6000	5	50	7200	20.5	59	12	1200	5	50	7200	10.8	899.7	5	500	5	50	788.5	0	415.6
60	8	1.5	5	60	6000	4	49	7200	25	72.7	12	1200	5	49	7200	16	829.3	4	400	4	49	2266.4	0	1849.9
60	8	3.0	1	60	4800	5	37	7200	43.8	57.9	12	960	5	37	7200	30.3	660.5	5	400	5	36	7200	13.1	3275.2
60	8	3.0	2	60	4800	5	36	7200	42.3	56.1	12	960	5	35	7200	28	806	5	400	5	35	7200	12.4	2733.6
60	8	3.0	3	60	4800	5	35	7200	43.5	51.1	12	960	5	35	7200	30	814.4	5	400	5	35	7200	13.8	3091.5
60	8	3.0	4	60	4800	4	34	7200	39	63	12	960	4	34	7200	25.2	785.1	4	320	4	34	3725.3	0	3034.4
60	8	3.0	5	60	4800	5	36	7200	42.3	63.1	12	960	5	35	7200	26	673.1	5	400	5	35	7200	8.8	3113.2
60	8	4.5	1	60	4380	5	27	7200	49	69.8	12	876	5	28	7200	34.9	831.4	5	365	5	27	7200	11.8	3200.9
60	8	4.5	2	60	4380	5	27	7200	49	67.1	12	876	5	27	7200	30.2	778	5	365	5	27	7200	9.9	3639.6

Continued on Next Page...

Table A.3 – Continued

Instances				Default						Improved Bounds						Minimum					
m	C	ρ	#	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node	p	cols	p'	best	CPU	gap	node
60	8	4.5	3	60	4380	5	29	7200	52.4	64.1	12	876	6	29	7200	34.7	661.4	5	365	5	29	7200	17.5	2814.6
60	8	4.5	4	60	4380	4	26	7200	47.1	81.2	12	876	4	27	7200	31.5	756.2	4	292	4	26	567.9	0	412.7
60	8	4.5	5	60	4380	5	28	7200	50.7	63.2	12	876	5	28	7200	33.9	806.4	5	365	5	28	7200	15	3784.1

Table A.4: Detailed results for comparison between different relaxations

Instances				USNP$_{(V,E,C)}$							USNP-X$_{(V,E,C)}$							USNP-Y$_{(V,E,C)}$						
m	C	ρ	#	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts
30	2	1.5	1	750	6021.7	35	35	0	2930.2	40	450	4696.2	35	35	0	2367.2	14	750	7200	32.4	35	7.34	2127.6	40
30	2	1.5	2	750	5199.7	34	34	0	2672	23	450	729	34	34	0	350.4	11	750	7200	31.1	34	8.63	2298.5	23
30	2	1.5	3	750	7200	32.3	36	10.4	2268.7	19	450	6435.4	36	36	0	2903.7	12	750	7200	32.2	36	10.59	2116.8	19
30	2	1.5	4	750	7200	29.2	34	14	2217.4	33	450	2212.7	34	34	0	1244.8	9	750	7200	29.8	34	12.48	1982.8	34
30	2	1.5	5	750	7200	32.4	36	10	2373.8	33	450	5012.6	36	36	0	2792.8	13	750	7200	32.5	36	9.72	2050.1	33
30	2	3.0	1	600	7200	26.1	31	15.7	2560.6	28	450	6259.2	31	31	0	3726.9	3	600	7200	25.6	31	17.38	2140.2	23
30	2	3.0	2	600	7200	24.9	29	14.2	2653.4	23	450	7200	22.3	29	23.2	2449.5	5	600	7200	23.5	29	18.92	1884.8	28
30	2	3.0	3	600	7200	24.9	31	19.8	2296.8	31	450	7200	24.6	31	20.6	2340.2	3	600	7200	24.1	31	22.36	1654.7	34
30	2	3.0	4	600	7200	22.5	27	16.8	1996.7	25	450	7200	26.2	27	2.8	2371	5	600	7200	22.2	27	17.68	1779.4	25
30	2	3.0	5	600	7200	22.6	30	24.6	2073.6	28	450	7200	23.1	30	23.1	2130.1	12	600	7200	23.8	30	20.6	1940	27
30	2	4.5	1	540	4316.2	24	24	0	2747.3	38	450	7200	20.8	24	13.5	3805	13	540	3479.2	24	24	0	2157.9	38
30	2	4.5	2	540	7200	22.1	26	15	2774.7	49	450	7200	19.2	26	26.3	3233.2	3	540	7200	22.9	26	12.1	3308.2	42
30	2	4.5	3	540	7200	23.2	26	10.6	3449.9	46	450	7200	22.8	26	12.3	3872.9	12	540	7200	23.8	26	8.43	3489.4	41
30	2	4.5	4	540	7200	23.6	27	12.8	2486.3	120	450	7200	20.1	27	25.4	3118.9	3	540	1917.2	27	27	0	877.9	123
30	2	4.5	5	540	7200	23.2	27	14.2	2746	46	450	7200	21	27	22.2	3134.9	4	540	7200	24.4	27	9.6	2848.1	49
35	2	1.5	1	1044	7200	33.8	41	17.5	1212.6	31	630	7200	35.6	41	13.3	1526.5	14	1044	7200	33.5	41	18.34	1097.3	28
35	2	1.5	2	1044	7200	36.2	44	17.8	1300	14	630	7200	35.9	44	18.5	1502.3	14	1044	7200	36.1	44	17.92	1270.1	14
35	2	1.5	3	1044	7200	35.2	40	12	1368.8	10	630	7200	36.7	40	8.2	1613.8	13	1044	7200	34.2	40	14.47	1148.4	10
35	2	1.5	4	1044	7200	34.7	41	15.4	1281.6	13	630	7200	38.9	41	5.1	1814	14	1044	7200	33.5	41	18.37	1032.3	15
35	2	1.5	5	1044	7200	31.6	40	21.1	1173.6	13	630	7200	31.2	40	22	1293.6	15	1044	7200	32.1	40	19.68	1026.8	13
35	2	3.0	1	828	7200	24.8	35	29.1	1406	8	630	7200	25.4	35	27.5	1401.8	3	828	7200	24.6	35	29.65	1118.1	12
35	2	3.0	2	828	7200	25.4	33	23.1	1304.7	7	630	7200	25.7	34	24.5	1436.5	3	828	7200	25	34	26.58	1160.8	8
35	2	3.0	3	828	7200	24.4	35	30.3	1331.2	24	630	7200	27.9	35	20.4	1266.1	14	828	7200	25.3	35	27.84	1179.3	25
35	2	3.0	4	828	7200	25.5	32	20.4	1274.3	20	630	7200	28.6	32	10.6	1594.6	4	828	7200	24.4	32	23.72	1207.8	20
35	2	3.0	5	828	7200	26.8	33	18.9	1371.9	34	630	7200	24.1	33	27.1	1283.7	3	828	7200	27	33	18.11	1232.4	43
35	2	4.5	1	756	7200	20.3	32	36.7	1340	33	630	7200	19.5	32	39.2	1362.5	4	756	7200	21.7	32	32.3	1257.7	36
35	2	4.5	2	756	7200	24	32	24.9	1584.9	61	630	7200	20.8	32	35.1	1754.2	6	756	7200	23.6	32	26.18	1322.4	65
35	2	4.5	3	756	7200	25.7	30	14.5	1614.3	53	630	7200	20.3	30	32.3	1435.6	4	756	7200	25.7	30	14.37	1332.9	60
35	2	4.5	4	756	7200	22.6	33	31.4	1367.1	25	630	7200	22	33	33.3	1435.8	6	756	7200	22.9	33	30.69	1040.3	27
35	2	4.5	5	756	7200	24.8	31	20.1	1640	46	630	7200	20.8	31	32.9	1560.3	4	756	7200	24.6	31	20.64	1515.4	56
40	2	1.5	1	1320	7200	39.4	50	21.2	905.2	7	800	7200	39	50	22.1	949	0	1320	7200	37.5	50	24.97	748.9	8
40	2	1.5	2	1320	7200	36.7	48	23.5	828.4	12	800	7200	36.5	48	23.9	848.5	14	1320	7200	36.3	48	24.48	770.7	12
40	2	1.5	3	1320	7200	37.3	46	19	1160.1	11	800	7200	38.3	46	16.7	1100.8	14	1320	7200	37.4	46	18.72	867.4	11
40	2	1.5	4	1320	7200	38.1	46	17.2	906.8	14	800	7200	40.8	46	11.2	935.2	13	1320	7200	36.6	46	20.41	819.7	14
40	2	1.5	5	1320	7200	40.7	48	15.3	832.5	10	800	7200	39.4	49	19.5	923.3	6	1320	7200	39.3	48	18.13	761.5	9
40	2	3.0	1	1060	7200	25.1	38	33.9	817.8	22	800	7200	25	38	34.3	846.5	2	1060	7200	24.8	38	34.68	701.6	24
40	2	3.0	2	1060	7200	23.9	37	35.4	829.1	28	800	7200	26.1	37	29.5	794.5	4	1060	7200	25.4	37	31.27	689	28
40	2	3.0	3	1060	7200	29	40	27.5	798	50	800	7200	25.1	40	37.2	808.2	3	1060	7200	27.3	40	31.79	828.5	47
40	2	3.0	4	1060	7200	26.7	40	33.4	842.5	14	800	7200	28.1	40	29.6	960.4	2	1060	7200	26.2	40	34.55	639.6	14

Continued on Next Page...

Table A.4 – Continued

Instances				USNP$_{(V,E,C)}$							USNP-X$_{(V,E,C)}$							USNP-Y$_{(V,E,C)}$						
m	C	ρ	#	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts
40	2	3.0	5	1060	7200	25.2	39	35.4	833.2	5	800	7200	27.5	39	29.5	893.6	6	1060	7200	26.6	39	31.87	662.3	6
40	2	4.5	1	960	7200	24	35	31.3	1062.3	48	800	7200	21.1	35	39.7	1022	6	960	7200	23.2	35	33.85	1020.3	46
40	2	4.5	2	960	7200	24.6	36	31.7	1074.2	65	800	7200	20.6	36	42.7	865.1	2	960	7200	26.9	36	25.37	935.8	59
40	2	4.5	3	960	7200	25.7	34	24.4	1139.8	44	800	7200	22	34	35.4	876.3	4	960	7200	25.9	34	23.89	924.1	53
40	2	4.5	4	960	7200	24.8	36	31	941.3	54	800	7200	21.7	35	38.1	914.1	3	960	7200	23.7	35	32.24	931.9	49
40	2	4.5	5	960	7200	21.9	35	37.3	922.9	47	800	7200	22.4	35	36.1	878.4	4	960	7200	23.4	35	33.17	854.1	47
45	2	1.5	1	1725	7200	40.2	49	18	563.3	11	1035	7200	42	49	14.2	748.1	3	1725	7200	38.9	50	22.29	517.3	9
45	2	1.5	2	1725	7200	39.4	53	25.7	481.2	14	1035	7200	40.8	53	23.1	582.8	6	1725	7200	38.8	53	26.87	447.4	14
45	2	1.5	3	1725	7200	43.2	55	21.4	550.2	17	1035	7200	44	55	20.1	710.4	16	1725	7200	41.8	55	23.96	611.6	17
45	2	1.5	4	1725	7200	43.6	54	19.3	565.5	20	1035	7200	42.8	54	20.8	611.2	15	1725	7200	42	54	22.19	430.9	20
45	2	1.5	5	1725	7200	42.8	53	19.3	565.1	15	1035	7200	45.2	53	14.6	695.2	8	1725	7200	40.8	53	23.11	463.2	14
45	2	3.0	1	1380	7200	29.4	45	34.6	478.7	28	1035	7200	28.9	44	34.3	527.4	4	1380	7200	28.6	45	36.48	472.7	28
45	2	3.0	2	1380	7200	30.9	48	35.6	528.6	27	1035	7200	30.2	47	35.8	548.7	0	1380	7200	30	47	36.24	437.7	27
45	2	3.0	3	1380	7200	29.7	46	35.5	524	4	1035	7200	30	46	34.8	543.9	3	1380	7200	28.9	46	37.13	456.5	4
45	2	3.0	4	1380	7200	28.9	43	32.7	586.4	4	1035	7200	26	43	39.6	540.3	5	1380	7200	26.7	42	36.52	498.9	3
45	2	3.0	5	1380	7200	31.2	45	30.6	632.5	1	1035	7200	30.8	45	31.6	567.7	2	1380	7200	29.7	45	34.07	524.6	3
45	2	4.5	1	1265	7200	26.3	42	37.3	627.4	48	1035	7200	24.5	42	41.8	591.1	3	1265	7200	27.1	42	35.56	456.3	51
45	2	4.5	2	1265	7200	29.6	43	31.1	718.3	51	1035	7200	23.8	42	43.4	639.9	5	1265	7200	29.2	42	30.39	578.4	50
45	2	4.5	3	1265	7200	28.1	42	33.1	599.7	50	1035	7200	24	42	43	554.3	2	1265	7200	27.3	42	34.94	511.5	54
45	2	4.5	4	1265	7200	29.5	45	34.4	561	65	1035	7200	25.7	46	44	647.1	5	1265	7200	29	46	36.95	507.7	64
45	2	4.5	5	1265	7200	24.8	40	38.1	634	32	1035	7200	21.7	40	45.7	554.1	3	1265	7200	26.1	39	32.95	569.8	36
50	2	1.5	1	2075	7200	50	63	20.6	389.1	7	1250	7200	49	62	20.9	418.6	6	2075	7200	46.7	62	24.66	346.9	9
50	2	1.5	2	2075	7200	45.7	59	22.6	401.2	16	1250	7200	44.7	59	24.3	411.8	7	2075	7200	44.5	59	24.52	339.4	16
50	2	1.5	3	2075	7200	44.1	58	24	353.9	13	1250	7200	43.4	58	25.2	417.2	4	2075	7200	42.3	59	28.29	341.5	12
50	2	1.5	4	2075	7200	46.1	60	23.2	395.6	4	1250	7200	44.7	60	25.5	427.7	3	2075	7200	43.3	60	27.79	438	3
50	2	1.5	5	2075	7200	44.7	60	25.6	328.1	8	1250	7200	45	60	25.1	374.3	6	2075	7200	43.3	60	27.9	360.4	9
50	2	3.0	1	1650	7200	30.5	50	38.9	413.7	30	1250	7200	29.8	51	41.6	363.9	0	1650	7200	30.2	51	40.85	367.2	30
50	2	3.0	2	1650	7200	32.7	55	40.6	354.9	6	1250	7200	31.9	55	42	399.2	0	1650	7200	33	55	39.94	245.6	7
50	2	3.0	3	1650	7200	31.1	49	36.6	439	5	1250	7200	30.4	49	38	407.1	7	1650	7200	29.9	49	39.01	307.5	7
50	2	3.0	4	1650	7200	31.5	49	35.6	367.1	29	1250	7200	30.2	49	38.5	392.2	0	1650	7200	30.5	49	37.71	370.3	29
50	2	3.0	5	1650	7200	32.9	52	36.7	323.5	4	1250	7200	30.4	52	41.6	346.3	1	1650	7200	29.6	52	43.05	324.6	5
50	2	4.5	1	1525	7200	25.5	48	46.9	330.6	6	1250	7200	24.5	48	48.9	368.6	0	1525	7200	24.3	47	48.34	278.5	5
50	2	4.5	2	1525	7200	29.9	44	32	517.3	40	1250	7200	25.9	44	41.2	414.2	0	1525	7200	31.1	44	29.42	422.5	37
50	2	4.5	3	1525	7200	28.2	47	40.1	353.4	27	1250	7200	25.9	47	44.9	391.1	0	1525	7200	26.7	47	43.28	355.8	27
50	2	4.5	4	1525	7200	28.8	49	41.3	326.8	31	1250	7200	26.2	48	45.3	387.2	0	1525	7200	25.1	49	48.71	316.5	29
50	2	4.5	5	1525	7200	29.6	48	38.3	371.4	33	1250	7200	27.5	47	41.6	399.9	3	1525	7200	28.9	48	39.83	351.2	33
55	2	1.5	1	2548	7200	49.6	64	22.5	242	8	1540	7200	48.7	64	23.8	329.2	9	2548	7200	47.2	64	26.31	313.8	8
55	2	1.5	2	2548	7200	48.4	66	26.7	250.9	6	1540	7200	49.3	66	25.3	323.6	3	2548	7200	46.6	66	29.4	287.9	6
55	2	1.5	3	2548	7200	50.3	64	21.4	229.1	6	1540	7200	49.6	62	20	305.4	7	2548	7200	47.3	63	24.98	312.4	5

Continued on Next Page...

Table A.4 – Continued

Instances				USNP$_{(V,E,C)}$							USNP-X$_{(V,E,C)}$							USNP-Y$_{(V,E,C)}$						
m	C	ρ	#	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts
55	2	1.5	4	2548	7200	48.8	64	23.7	210.9	9	1540	7200	47.3	64	26.1	250.8	8	2548	7200	46.7	64	27.11	223.8	9
55	2	1.5	5	2548	7200	47.9	65	26.2	243.8	8	1540	7200	46.4	66	29.6	270.1	10	2548	7200	45.7	65	29.7	253.8	8
55	2	3.0	1	2044	7200	36.7	57	35.7	205.8	33	1540	7200	32.1	56	42.7	296.8	0	2044	7200	34.4	58	40.71	177.2	35
55	2	3.0	2	2044	7200	32.1	55	41.7	249.8	3	1540	7200	33.7	55	38.8	275	0	2044	7200	31.9	55	42.06	233.3	3
55	2	3.0	3	2044	7200	32.3	58	44.3	209.2	1	1540	7200	32.1	57	43.7	243.3	1	2044	7200	31.5	57	44.7	196.4	5
55	2	3.0	4	2044	7200	30.6	55	44.4	220.9	2	1540	7200	32.6	56	41.8	233.4	0	2044	7200	33.1	54	38.67	194.4	2
55	2	3.0	5	2044	7200	30.7	57	46.2	215.1	3	1540	7200	31.1	57	45.4	340.6	0	2044	7200	31.3	56	44.09	194.1	4
55	2	4.5	1	1876	7200	23.3	50	53.3	229.1	9	1540	7200	24.8	49	49.5	343.7	0	1876	7200	23.3	49	52.36	234.6	9
55	2	4.5	2	1876	7200	29.6	51	42	241.8	62	1540	7200	25.6	52	50.7	267.8	0	1876	7200	30.7	50	38.58	250	62
55	2	4.5	3	1876	7200	29.5	51	42.2	240.8	69	1540	7200	26.6	51	47.8	289.3	0	1876	7200	28.8	51	43.54	259.9	66
55	2	4.5	4	1876	7200	27.4	52	47.4	237.7	3	1540	7200	26.2	53	50.5	298.7	0	1876	7200	26	53	50.96	197.7	3
55	2	4.5	5	1876	7200	27.6	56	50.7	272.6	9	1540	7200	27	55	50.9	281.6	0	1876	7200	27.2	56	51.44	179.2	11
60	2	1.5	1	3000	7200	52.6	74	28.9	163.9	8	1800	7200	53	73	27.5	176.7	2	3000	7200	51.5	74	30.38	178.9	5
60	2	1.5	2	3000	7200	50.6	73	30.7	209.6	13	1800	7200	51.2	73	29.9	185.6	7	3000	7200	50	73	31.56	194.4	13
60	2	1.5	3	3000	7200	50.1	73	31.4	136.5	8	1800	7200	51.1	71	28	182.3	6	3000	7200	48.4	72	32.77	201.4	8
60	2	1.5	4	3000	7200	51.8	72	28	167.5	34	1800	7200	51.5	71	27.5	189.7	0	3000	7200	50.1	73	31.36	153.2	35
60	2	1.5	5	3000	7200	52.7	70	24.8	179.8	5	1800	7200	53	70	24.4	211.8	8	3000	7200	52.2	70	25.39	171	5
60	2	3.0	1	2400	7200	36.3	64	43.3	142.5	4	1800	7200	34.7	65	46.6	163	0	2400	7200	35	65	46.11	136.7	5
60	2	3.0	2	2400	7200	41.6	62	32.9	153.2	38	1800	7200	33.5	62	45.9	157.4	5	2400	7200	37.5	62	39.51	141	38
60	2	3.0	3	2400	7200	38.9	64	39.3	197.6	38	1800	7200	36.1	63	42.7	163.6	0	2400	7200	35.2	63	44.11	138.9	34
60	2	3.0	4	2400	7200	35.7	64	44.3	137	34	1800	7200	35.5	63	43.7	147.2	0	2400	7200	36.2	63	42.53	106.6	35
60	2	3.0	5	2400	7200	36.6	64	42.9	145.9	64	1800	7200	33.7	63	46.5	187.8	0	2400	7200	36.7	63	41.82	171.9	64
60	2	4.5	1	2190	7200	27	59	54.2	168.6	2	1800	7200	28.6	58	50.7	205.9	0	2190	7200	26.9	59	54.33	185.3	0
60	2	4.5	2	2190	7200	30.5	57	46.4	178.7	36	1800	7200	26.8	56	52.1	200.8	3	2190	7200	30.2	57	47.06	209	35
60	2	4.5	3	2190	7200	29.9	57	47.6	163.5	37	1800	7200	26	57	54.3	207.5	0	2190	7200	28.6	57	49.78	150.2	36
60	2	4.5	4	2190	7200	30.4	58	47.5	202.9	35	1800	7200	28.6	58	50.8	169.5	0	2190	7200	31.1	59	47.24	169.7	35
60	2	4.5	5	2190	7200	28.3	57	50.3	175.3	31	1800	7200	26.8	57	53.1	187.9	0	2190	7200	28.1	58	51.61	147.6	32
30	5	1.5	1	500	14.1	25	25	0	9.8	13	300	16.5	25	25	0	8.3	4	500	5	25	25	0	3.1	13
30	5	1.5	2	500	8.1	27	27	0	4.8	9	300	13.1	27	27	0	7.1	6	500	9.3	27	27	0	5.6	9
30	5	1.5	3	500	3.2	27	27	0	0.9	22	300	2.3	27	27	0	0.9	11	500	2.7	27	27	0	0.6	24
30	5	1.5	4	500	13.6	27	27	0	7.9	7	300	17.4	27	27	0	10	3	500	40.3	27	27	0	21.2	8
30	5	1.5	5	500	9.6	28	28	0	5.8	7	300	7.3	28	28	0	3.7	1	500	18.5	28	28	0	10.6	7
30	5	3.0	1	400	92.2	19	19	0	60.5	20	300	39.6	19	19	0	20.7	6	400	167	19	19	0	107.3	18
30	5	3.0	2	400	31.8	19	19	0	20	15	300	33.8	19	19	0	18.9	2	400	24.4	19	19	0	14.7	14
30	5	3.0	3	400	39.3	18	18	0	21.1	14	300	41.8	18	18	0	19.9	7	400	20.2	18	18	0	10.1	15
30	5	3.0	4	400	28.9	18	18	0	19.4	12	300	82.8	18	18	0	39.3	4	400	88	18	18	0	65.9	11
30	5	3.0	5	400	209.8	19	19	0	156.8	9	300	186.2	19	19	0	101.4	2	400	92.5	19	19	0	63.1	8
30	5	4.5	1	360	1.8	14	14	0	0.6	19	300	68.9	14	14	0	48.9	1	360	1.8	14	14	0	0.9	18
30	5	4.5	2	360	3.5	13	13	0	1.8	42	300	25.4	13	13	0	15.1	2	360	1.3	13	13	0	0.3	43

Continued on Next Page...

Table A.4 – Continued

Instances				USNP$_{(V,E,C)}$							USNP-X$_{(V,E,C)}$							USNP-Y$_{(V,E,C)}$						
m	C	ρ	#	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts
30	5	4.5	3	360	1.4	13	13	0	0.3	30	300	169.3	13	13	0	99.6	4	360	1	13	13	0	0.3	29
30	5	4.5	4	360	51.2	14	14	0	34.9	28	300	84.9	14	14	0	56.5	5	360	7.4	14	14	0	4	30
30	5	4.5	5	360	4	13	13	0	1.9	20	300	28.3	13	13	0	15.2	0	360	2.8	13	13	0	1.2	19
35	5	1.5	1	696	522	31	31	0	227.1	27	420	111	31	31	0	39.6	3	696	755.6	31	31	0	295.6	32
35	5	1.5	2	696	25.6	31	31	0	10.1	8	420	114.8	31	31	0	50	3	696	69.4	31	31	0	28.5	8
35	5	1.5	3	696	35.1	29	29	0	13.3	9	420	76.9	29	29	0	29.8	10	696	53.7	29	29	0	20.1	10
35	5	1.5	4	696	2583.6	33	33	0	1136	15	420	1033.1	33	33	0	409.5	1	696	1277.5	33	33	0	537.7	16
35	5	1.5	5	696	413.3	31	31	0	172.2	14	420	242	31	31	0	89	6	696	612.8	31	31	0	249.5	15
35	5	3.0	1	552	1701.5	22	22	0	882.2	12	420	906.5	22	22	0	281.9	5	552	1640.4	22	22	0	704.3	9
35	5	3.0	2	552	461.4	21	21	0	243.1	19	420	455.5	21	21	0	156.6	9	552	733	21	21	0	316.4	18
35	5	3.0	3	552	7200	19.3	23	16.1	2162.2	40	420	7200	21.4	23	6.8	2284.2	6	552	7200	19.8	23	13.91	1913.8	40
35	5	3.0	4	552	7200	19.9	22	9.6	2466.5	15	420	7200	19.5	22	11.3	2130	5	552	7051.9	22	22	0	2868.8	12
35	5	3.0	5	552	385.4	21	21	0	156.8	18	420	276.5	21	21	0	99.2	3	552	282.8	21	21	0	122.9	17
35	5	4.5	1	504	7200	16.5	18	8.2	3069.5	46	420	4329.6	18	18	0	2011.6	8	504	186	18	18	0	67.5	46
35	5	4.5	2	504	122.1	16	16	0	52.9	25	420	324.6	16	16	0	111.1	3	504	36.1	16	16	0	12.6	26
35	5	4.5	3	504	6202	17	17	0	3083.9	24	420	7200	15.2	17	10.4	2374.8	8	504	161	17	17	0	63.1	25
35	5	4.5	4	504	359.1	16	16	0	151	56	420	1182.6	16	16	0	430.7	5	504	158.6	16	16	0	54.8	59
35	5	4.5	5	504	193	17	17	0	91.9	27	420	1487.8	17	17	0	530.6	0	504	75.6	17	17	0	27.5	27
40	5	1.5	1	924	1895.6	37	37	0	546.1	7	560	2182.5	37	37	0	501.3	1	924	7200	35.2	37	4.79	1637.2	11
40	5	1.5	2	924	177.5	36	36	0	47	5	560	128.8	36	36	0	25.2	5	924	593.7	36	36	0	149.2	8
40	5	1.5	3	924	7200	33.9	37	8.3	1206.1	12	560	4112.6	37	37	0	840.6	4	924	7200	34.5	38	9.27	1249.3	11
40	5	1.5	4	924	2786.3	37	37	0	851.9	17	560	1908.9	37	37	0	439.3	12	924	4863.6	37	37	0	1304.7	17
40	5	1.5	5	924	127.6	33	33	0	30.5	9	560	223	33	33	0	60.2	5	924	179.1	33	33	0	44	11
40	5	3.0	1	742	7200	22.2	26	14.7	1362.7	36	560	7200	23.2	26	10.8	1419.2	3	742	7200	22.9	26	11.8	1243.9	41
40	5	3.0	2	742	7200	22.2	25	11.2	1473.7	8	560	7200	22.5	25	9.9	1445.8	3	742	7200	23.1	25	7.74	1468.2	16
40	5	3.0	3	742	7200	22.2	25	11.3	1574.9	26	560	7200	22.5	25	10.1	1557.6	2	742	7200	22.1	25	11.68	1284.9	36
40	5	3.0	4	742	7200	21.5	25	14	1436.7	24	560	7200	22.4	25	10.2	1519.6	3	742	7200	22.4	25	10.5	1292.1	24
40	5	3.0	5	742	7200	22	26	15.6	1325.3	40	560	7200	23.4	26	10	1324.2	1	742	7200	22.5	26	13.61	1154.1	41
40	5	4.5	1	672	7200	16	19	16	1310.3	37	560	7200	16.3	20	18.4	1388.2	4	672	2130.2	19	19	0	498.6	39
40	5	4.5	2	672	7200	16.3	20	18.3	1526.8	26	560	7200	17.2	20	14	1529.8	1	672	5773.4	20	20	0	1546.6	26
40	5	4.5	3	672	4727	18	18	0	1551.1	25	560	7200	15.1	18	16.1	1504.3	1	672	262.4	18	18	0	62.6	22
40	5	4.5	4	672	7200	16.4	20	17.9	1607.6	10	560	7200	16.3	20	18.3	1483.2	7	672	2305.2	20	20	0	773	10
40	5	4.5	5	672	2980.2	20	20	0	935	22	560	7200	18	20	9.9	1511.2	4	672	468.5	20	20	0	109.2	25
45	5	1.5	1	1125	7200	37.7	40	5.7	1143.9	6	675	7200	37.2	40	7.1	1205.4	0	1125	7200	36.6	40	8.44	1062.7	6
45	5	1.5	2	1125	7200	38.1	41	7.1	1036.2	44	675	7200	38.3	41	6.6	1312.9	18	1125	7200	37.8	41	7.75	944.4	47
45	5	1.5	3	1125	7200	37.1	41	9.5	1075	26	675	7200	38.8	41	5.5	993.3	4	1125	7200	37.2	41	9.38	850.7	29
45	5	1.5	4	1125	7200	39	42	7.2	1149.4	4	675	7200	38.6	42	8.2	1123.4	0	1125	7200	38.3	42	8.89	971.5	4
45	5	1.5	5	1125	7200	36.1	43	15.9	1040.5	13	675	7200	36.5	43	15.1	1133.5	11	1125	7200	36.3	43	15.63	1008.7	15
45	5	3.0	1	900	7200	23.3	30	22.3	992.7	30	675	7200	24.4	31	21.3	899.7	1	900	7200	23	30	23.33	853.8	30

Continued on Next Page...

Table A.4 – Continued

Instances				USNP$_{(V,E,C)}$							USNP-X$_{(V,E,C)}$							USNP-Y$_{(V,E,C)}$						
m	C	ρ	#	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts
45	5	3.0	2	900	7200	23.1	31	25.4	1035.5	13	675	7200	25.6	31	17.6	956.9	6	900	7200	24.5	31	20.91	842	13
45	5	3.0	3	900	7200	23.1	30	23.2	1047.4	30	675	7200	23.9	30	20.5	988.1	6	900	7200	23.6	30	21.24	844.6	31
45	5	3.0	4	900	7200	24.7	31	20.3	1020.5	10	675	7200	25.7	31	16.9	1014.4	8	900	7200	25.4	31	17.94	881.5	12
45	5	3.0	5	900	7200	23.6	31	23.7	994.4	9	675	7200	24.5	31	20.9	964.5	4	900	7200	24.6	31	20.53	859.9	10
45	5	4.5	1	825	7200	18.5	25	26.2	1056.3	54	675	7200	18.8	25	25	1082.2	3	825	7200	20.1	25	19.52	809	52
45	5	4.5	2	825	7200	17.7	23	22.9	1078.4	13	675	7200	18.4	23	19.8	1032.7	2	825	7200	19.3	23	16.24	879.7	15
45	5	4.5	3	825	7200	18.9	24	21.1	1130.6	53	675	7200	19.8	24	17.7	1039.2	5	825	7200	20.1	24	16.39	883.3	56
45	5	4.5	4	825	7200	19.4	25	22.2	993.1	23	675	7200	18.4	25	26.4	949.3	3	825	7200	20.6	25	17.45	913.6	27
45	5	4.5	5	825	7200	18.9	24	21.1	1137.7	50	675	7200	18.6	24	22.6	1039.3	1	825	7200	20.1	24	16.36	863.5	50
50	5	1.5	1	1411	7200	41.6	46	9.5	858	5	850	7200	41	46	11	780.9	0	1411	7200	40.6	46	11.83	826.8	4
50	5	1.5	2	1411	7200	41	46	10.8	780.4	14	850	7200	42.3	46	8.1	743.4	5	1411	7200	41.6	46	9.59	705.8	14
50	5	1.5	3	1411	7200	41.3	46	10.1	760.5	8	850	7200	41.7	46	9.4	724.7	0	1411	7200	41.1	46	10.75	670.6	9
50	5	1.5	4	1411	7200	40.7	48	15.3	708	12	850	7200	40	48	16.6	671.1	4	1411	7200	39.3	48	18.1	644.8	12
50	5	1.5	5	1411	7200	41.1	48	14.5	656.3	18	850	7200	41	48	14.5	661.6	3	1411	7200	40.7	48	15.2	569.9	18
50	5	3.0	1	1122	7200	24.4	32	23.7	705.1	17	850	7200	23.9	31	22.8	716.3	4	1122	7200	24.7	31	20.36	464.7	16
50	5	3.0	2	1122	7200	25.3	37	31.5	677.7	6	850	7200	25	36	30.6	611.4	2	1122	7200	26.1	36	27.61	512	7
50	5	3.0	3	1122	7200	25.2	35	28.1	682.3	6	850	7200	25.2	34	25.8	649.2	2	1122	7200	25.9	35	25.95	581.7	7
50	5	3.0	4	1122	7200	24.4	35	30.4	717.9	25	850	7200	25.3	35	27.8	561.6	2	1122	7200	26.2	34	22.89	624.6	25
50	5	3.0	5	1122	7200	24.2	30	19.5	729.7	13	850	7200	26.2	30	12.6	759	2	1122	7200	24.2	30	19.24	656.2	13
50	5	4.5	1	1037	7200	20.9	29	27.9	755.3	53	850	7200	21	30	30.2	778.1	3	1037	7200	21.8	29	24.92	603.6	53
50	5	4.5	2	1037	7200	21.1	31	31.8	775.8	26	850	7200	20.3	30	32.2	811.9	2	1037	7200	22.3	30	25.61	605.8	33
50	5	4.5	3	1037	7200	19.3	27	28.6	805.3	34	850	7200	19.9	28	28.9	749.7	4	1037	7200	20.7	28	26.15	630.9	34
50	5	4.5	4	1037	7200	18.9	26	27.4	824.1	14	850	7200	19.9	26	23.6	725.3	1	1037	7200	19.8	26	23.8	627.2	14
50	5	4.5	5	1037	7200	19.2	29	33.8	718.1	13	850	7200	19.2	30	36	650	2	1037	7200	20	30	33.45	504.8	12
55	5	1.5	1	1729	7200	43.2	51	15.3	532.9	16	1045	7200	44.5	51	12.7	507.1	0	1729	7200	44.4	51	12.92	416.4	16
55	5	1.5	2	1729	7200	44	51	13.7	567.2	10	1045	7200	43.7	51	14.3	507.4	1	1729	7200	43.5	52	16.39	469.4	10
55	5	1.5	3	1729	7200	43.1	50	13.8	588.4	21	1045	7200	43.7	50	12.7	652.5	15	1729	7200	43.3	50	13.45	556.4	25
55	5	1.5	4	1729	7200	40.6	50	18.8	520.8	8	1045	7200	40.5	49	17.4	478.6	6	1729	7200	40.6	49	17.24	474.6	8
55	5	1.5	5	1729	7200	44.3	52	14.7	583.9	17	1045	7200	43.9	52	15.5	547.1	9	1729	7200	44.2	52	15.06	464.3	17
55	5	3.0	1	1387	7200	27.7	40	30.7	482	10	1045	7200	26.6	40	33.4	461.5	1	1387	7200	27.3	39	29.88	421.6	7
55	5	3.0	2	1387	7200	26.3	37	29	453	20	1045	7200	25.8	36	28.3	426.7	2	1387	7200	25.9	37	29.98	482.9	21
55	5	3.0	3	1387	7200	26	39	33.4	519.9	28	1045	7200	26.4	39	32.2	458.5	2	1387	7200	26.7	39	31.44	336.9	29
55	5	3.0	4	1387	7200	27	34	20.6	488.5	6	1045	7200	26.8	34	21	521.9	4	1387	7200	28.1	35	19.63	385.2	6
55	5	3.0	5	1387	7200	27	37	27.1	475.8	10	1045	7200	26.7	37	27.7	510.9	2	1387	7200	27.6	37	25.41	414.2	11
55	5	4.5	1	1273	7200	20.4	31	34.1	493.6	4	1045	7200	20.4	31	34.1	473.4	3	1273	7200	20.9	30	30.31	431.1	7
55	5	4.5	2	1273	7200	20.4	31	34.2	513.8	60	1045	7200	19.6	31	36.7	553.1	3	1273	7200	21.5	31	30.78	378.2	60
55	5	4.5	3	1273	7200	20.8	32	35	581.5	43	1045	7200	21.3	31	31.4	435.9	2	1273	7200	22.1	32	30.8	355.4	44
55	5	4.5	4	1273	7200	20.9	32	34.6	512	11	1045	7200	21	32	34.4	541.2	2	1273	7200	21.7	30	27.78	406.8	13
55	5	4.5	5	1273	7200	22.2	33	32.9	419.1	12	1045	7200	20.4	33	38.3	495.2	4	1273	7200	22	33	33.48	347.4	11

Continued on Next Page...

Table A.4 – Continued

Instances				USNP$_{(V,E,C)}$							USNP-X$_{(V,E,C)}$							USNP-Y$_{(V,E,C)}$						
m	C	ρ	#	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts
60	5	1.5	1	2000	7200	44.7	50	10.6	435.4	8	1200	7200	45.1	50	9.8	431	8	2000	7200	45.2	50	9.69	346	7
60	5	1.5	2	2000	7200	45.8	53	13.7	462.3	0	1200	7200	46.2	53	12.9	382.4	0	2000	7200	45.5	53	14.15	371.1	1
60	5	1.5	3	2000	7200	46.8	56	16.4	475.6	12	1200	7200	48.5	56	13.5	429.7	5	2000	7200	46.6	56	16.82	399.3	12
60	5	1.5	4	2000	7200	45	54	16.7	387.8	13	1200	7200	46.1	55	16.2	380.1	7	2000	7200	45.5	55	17.2	305.1	12
60	5	1.5	5	2000	7200	47.5	58	18	356.3	20	1200	7200	47.3	58	18.4	386.9	3	2000	7200	46.6	58	19.63	396.6	21
60	5	3.0	1	1600	7200	29.6	42	29.5	336	17	1200	7200	28.4	43	34.1	381	3	1600	7200	29.2	44	33.61	340.3	17
60	5	3.0	2	1600	7200	28.3	41	30.9	430.7	13	1200	7200	28.9	41	29.5	326.8	4	1600	7200	29.6	40	26.05	296.6	14
60	5	3.0	3	1600	7200	29.6	41	27.8	385.9	42	1200	7200	29.7	41	27.5	392.5	13	1600	7200	28.9	42	31.19	318.1	42
60	5	3.0	4	1600	7200	31.2	38	18	330.1	46	1200	7200	29.1	38	23.5	379.3	12	1600	7200	30.6	38	19.43	291.7	49
60	5	3.0	5	1600	7200	28.4	43	34	395.6	8	1200	7200	30.1	43	30.1	346.5	2	1600	7200	29.5	43	31.48	297.1	10
60	5	4.5	1	1460	7200	20.5	33	37.8	478.2	20	1200	7200	20.8	31	32.9	408.1	9	1460	7200	20.8	31	32.91	391.5	23
60	5	4.5	2	1460	7200	22.5	35	35.7	422	47	1200	7200	22	36	38.9	446	2	1460	7200	24.2	34	28.69	266.2	47
60	5	4.5	3	1460	7200	21.4	33	35.2	412.9	67	1200	7200	21.4	34	37	396.4	2	1460	7200	22.6	33	31.64	280.5	69
60	5	4.5	4	1460	7200	21.1	35	39.7	330.5	33	1200	7200	21.3	36	40.7	376.6	10	1460	7200	23	35	34.28	248.2	33
60	5	4.5	5	1460	7200	22.1	34	35.1	388.3	43	1200	7200	20.7	34	39	379	4	1460	7200	23.5	33	28.77	314.8	43
30	8	1.5	1	300	0.4	24	24	0	0.1	6	180	0.4	24	24	0	0.1	3	300	0.4	24	24	0	0.1	6
30	8	1.5	2	300	0.4	23	23	0	0	3	180	0.4	23	23	0	0.1	2	300	0.3	23	23	0	0.1	3
30	8	1.5	3	300	0.9	24	24	0	0.2	11	180	0.8	24	24	0	0.5	2	300	0.8	24	24	0	0.3	12
30	8	1.5	4	300	0.7	23	23	0	0.3	4	180	0.9	23	23	0	0.4	2	300	0.8	23	23	0	0.6	4
30	8	1.5	5	300	0.5	24	24	0	0.1	7	180	0.5	24	24	0	0.1	3	300	0.4	24	24	0	0.2	8
30	8	3.0	1	240	1.4	16	16	0	1.1	15	180	2.3	16	16	0	1.7	5	240	0.7	16	16	0	0.4	12
30	8	3.0	2	240	0.4	13	13	0	0	3	180	0.4	13	13	0	0.1	2	240	0.2	13	13	0	0	3
30	8	3.0	3	240	0.4	14	14	0	0	2	180	0.3	14	14	0	0	4	240	0.2	14	14	0	0	2
30	8	3.0	4	240	0.7	15	15	0	0.2	5	180	0.8	15	15	0	0.3	3	240	0.4	15	15	0	0.2	6
30	8	3.0	5	240	0.8	15	15	0	0.1	13	180	0.6	15	15	0	0.2	2	240	0.5	15	15	0	0.2	14
30	8	4.5	1	216	0.8	11	11	0	0.2	26	180	1.4	11	11	0	1.1	2	216	0.6	11	11	0	0.3	25
30	8	4.5	2	216	2.3	11	11	0	2	20	180	35.3	11	11	0	40.1	2	216	0.5	11	11	0	0.4	22
30	8	4.5	3	216	14	12	12	0	16.7	22	180	12.4	12	12	0	14.6	4	216	1.3	12	12	0	1.1	18
30	8	4.5	4	216	0.4	9	9	0	0	6	180	0.3	9	9	0	0	1	216	0.1	9	9	0	0	6
30	8	4.5	5	216	2	12	12	0	1.7	14	180	30.5	12	12	0	36.2	4	216	1	12	12	0	0.9	15
35	8	1.5	1	406	2.1	29	29	0	0.9	5	245	3.3	29	29	0	1.7	1	406	2.5	29	29	0	1.5	5
35	8	1.5	2	406	0.8	25	25	0	0.1	5	245	0.6	25	25	0	0.1	4	406	0.6	25	25	0	0.1	5
35	8	1.5	3	406	1.4	29	29	0	0.4	7	245	1.6	29	29	0	0.4	4	406	1	29	29	0	0.4	7
35	8	1.5	4	406	0.8	27	27	0	0	8	245	0.8	27	27	0	0.1	3	406	0.5	27	27	0	0.1	8
35	8	1.5	5	406	4.4	28	28	0	3.1	10	245	5.8	28	28	0	3.5	8	406	5.3	28	28	0	3.8	11
35	8	3.0	1	322	2.2	18	18	0	0.9	11	245	7.9	18	18	0	4.1	1	322	1	18	18	0	0.4	10
35	8	3.0	2	322	13.9	18	18	0	9.6	6	245	94.5	18	18	0	65.3	5	322	5.4	18	18	0	3.7	5
35	8	3.0	3	322	0.5	14	14	0	0	3	245	0.5	14	14	0	0	4	322	0.2	14	14	0	0	3
35	8	3.0	4	322	4.5	17	17	0	3.5	8	245	8.2	17	17	0	4.7	0	322	1.1	17	17	0	0.6	9

Continued on Next Page...

Table A.4 – Continued

Instances				USNP$_{(V,E,C)}$							USNP-X$_{(V,E,C)}$							USNP-Y$_{(V,E,C)}$						
m	C	ρ	#	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts
35	8	3.0	5	322	6.5	18	18	0	4	5	245	5.3	18	18	0	2.8	3	322	2.1	18	18	0	1.4	4
35	8	4.5	1	294	1.3	12	12	0	0.5	13	245	4	12	12	0	2.5	1	294	0.5	12	12	0	0.2	12
35	8	4.5	2	294	7.7	13	13	0	5.4	11	245	18	13	13	0	10.2	1	294	0.7	13	13	0	0.3	8
35	8	4.5	3	294	71.5	14	14	0	52.7	23	245	1208.3	14	14	0	839.6	0	294	2.8	14	14	0	1.5	17
35	8	4.5	4	294	1.3	12	12	0	0.4	8	245	6.5	12	12	0	4.9	3	294	0.4	12	12	0	0.1	8
35	8	4.5	5	294	24.6	13	13	0	21.9	15	245	33.4	13	13	0	21.9	7	294	1.4	13	13	0	0.8	12
40	8	1.5	1	528	7.3	32	32	0	2.8	5	320	7.4	32	32	0	3.1	7	528	2.9	32	32	0	0.9	5
40	8	1.5	2	528	4.1	31	31	0	1.2	5	320	2.9	31	31	0	0.9	1	528	2.4	31	31	0	0.6	5
40	8	1.5	3	528	40.4	31	31	0	21.1	13	320	20.8	31	31	0	8	8	528	35.7	31	31	0	16.7	13
40	8	1.5	4	528	90.1	34	34	0	55.6	6	320	100.6	34	34	0	45.8	11	528	125.9	34	34	0	65	7
40	8	1.5	5	528	2.4	32	32	0	0.6	3	320	4.4	32	32	0	1.3	3	528	2.4	32	32	0	0.8	4
40	8	3.0	1	424	101.7	22	22	0	50.8	7	320	71.7	22	22	0	30.9	0	424	108.2	22	22	0	58.9	7
40	8	3.0	2	424	5.4	20	20	0	1.4	14	320	12.5	20	20	0	5.1	0	424	2.1	20	20	0	0.7	12
40	8	3.0	3	424	162.1	23	23	0	81.2	18	320	282.4	23	23	0	103.4	6	424	373.1	23	23	0	165.6	17
40	8	3.0	4	424	327.2	22	22	0	174.3	8	320	321.5	22	22	0	140.8	6	424	51.3	22	22	0	23.1	7
40	8	3.0	5	424	73.1	22	22	0	31.7	5	320	65.9	22	22	0	27.1	5	424	81.7	22	22	0	39.2	7
40	8	4.5	1	384	2.8	14	14	0	0.9	25	320	14.5	14	14	0	6.8	1	384	1.8	14	14	0	0.5	20
40	8	4.5	2	384	826.3	17	17	0	585.5	9	320	1552.1	17	17	0	746.9	9	384	61.9	17	17	0	32.3	8
40	8	4.5	3	384	82.8	14	14	0	42.4	12	320	22	14	14	0	9	1	384	3.9	14	14	0	1.8	13
40	8	4.5	4	384	3.7	14	14	0	1.1	23	320	4.1	14	14	0	1.3	2	384	1.2	14	14	0	0.4	20
40	8	4.5	5	384	1008.8	17	17	0	566.9	20	320	5580.5	17	17	0	3109.5	5	384	81.6	17	17	0	43.1	17
45	8	1.5	1	675	19.6	38	38	0	4.8	3	405	45	38	38	0	10.9	0	675	64.3	38	38	0	19.9	3
45	8	1.5	2	675	34.6	38	38	0	9.1	5	405	129.3	38	38	0	39.9	0	675	33.1	38	38	0	9	5
45	8	1.5	3	675	135.5	36	36	0	45.1	6	405	336.6	36	36	0	105.1	1	675	81.6	36	36	0	27.2	6
45	8	1.5	4	675	50.1	36	36	0	14.5	4	405	87.8	36	36	0	26.3	0	675	40.9	36	36	0	12.1	4
45	8	1.5	5	675	7.5	36	36	0	2.1	6	405	7.3	36	36	0	1.9	2	675	11.4	36	36	0	4	5
45	8	3.0	1	540	564.5	25	25	0	204.5	6	405	509.3	25	25	0	145.9	2	540	196.1	25	25	0	65	6
45	8	3.0	2	540	1844.9	26	26	0	779.3	7	405	4123.8	26	26	0	1141.5	5	540	1226.4	26	26	0	465.5	4
45	8	3.0	3	540	79.4	24	24	0	24.1	4	405	169.7	24	24	0	47.2	0	540	46.4	24	24	0	16	4
45	8	3.0	4	540	424.3	25	25	0	138.8	4	405	712.7	25	25	0	228.7	4	540	328.8	25	25	0	107.4	4
45	8	3.0	5	540	887.4	24	24	0	362.3	23	405	208	24	24	0	62	1	540	168.3	24	24	0	59.7	25
45	8	4.5	1	495	1025.5	19	19	0	365.5	19	405	919.6	19	19	0	300.4	0	495	220.6	19	19	0	77.8	19
45	8	4.5	2	495	60.1	18	18	0	21.5	19	405	359	18	18	0	125.9	2	495	10.2	18	18	0	2.9	18
45	8	4.5	3	495	172.8	18	18	0	65.9	12	405	122	18	18	0	39.5	1	495	20.2	18	18	0	6.4	12
45	8	4.5	4	495	7200	19.3	21	8.1	2430.5	8	405	7200	18.8	21	10.6	2157.3	3	495	1426	21	21	0	598.3	8
45	8	4.5	5	495	7200	17.9	20	10.6	2497.9	14	405	7200	18.4	20	7.9	2224.3	5	495	702	20	20	0	288.3	14
50	8	1.5	1	830	210.3	41	41	0	45.6	6	500	189.9	41	41	0	35.6	2	830	109.4	41	41	0	19.6	7
50	8	1.5	2	830	280.3	38	38	0	73.2	11	500	99.6	38	38	0	17.1	7	830	277.6	38	38	0	60.3	12
50	8	1.5	3	830	116.2	39	39	0	27.6	11	500	183.1	39	39	0	35.8	1	830	92.6	39	39	0	22.9	8

Continued on Next Page...

Table A.4 – Continued

Instances				USNP$_{(V,E,C)}$							USNP-X$_{(V,E,C)}$							USNP-Y$_{(V,E,C)}$						
m	C	ρ	#	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts
50	8	1.5	4	830	1616.6	41	41	0	441.7	7	500	927.8	41	41	0	203.5	4	830	6992.2	41	41	0	1618.1	7
50	8	1.5	5	830	1129.3	42	42	0	288.6	16	500	1634.8	42	42	0	318.3	8	830	2006.8	42	42	0	434.8	19
50	8	3.0	1	660	7200.1	24.3	29	16.2	1318.2	9	500	7200	25.6	28	8.5	1347.7	0	660	7200	25.6	28	8.48	1341.5	9
50	8	3.0	2	660	7200	26.2	28	6.4	1522.6	11	500	7200	25.6	28	8.6	1301	4	660	5516.3	28	28	0	1309	11
50	8	3.0	3	660	7200	24	28	14.1	1330.1	8	500	7200	25	28	10.9	1283.6	1	660	7200	25	28	10.83	1399.4	8
50	8	3.0	4	660	7200	25.7	28	8.1	1421.6	10	500	7200	25	28	10.8	1278.3	1	660	6543.6	28	28	0	1575.3	11
50	8	3.0	5	660	7200.2	23.6	29	18.6	1497	5	500	7200	25.8	29	11	1344.2	1	660	7200	23.9	29	17.65	1422.9	5
50	8	4.5	1	610	7200	19	22	13.7	1505.7	16	500	7200	18.8	22	14.6	1328	3	610	4471.8	22	22	0	1311.8	16
50	8	4.5	2	610	7200	18.6	22	15.6	1553.2	16	500	7200	18.5	22	16	1457.7	8	610	6068.3	22	22	0	1758	17
50	8	4.5	3	610	7200	18.4	22	16.2	1611.4	10	500	7200	18.9	22	14.3	1415	3	610	3402.9	22	22	0	1021.7	10
50	8	4.5	4	610	7200	19.1	22	13.3	1438.3	19	500	7200	19.4	22	12	1342.9	2	610	3419.1	22	22	0	827.5	19
50	8	4.5	5	610	7200	18.8	23	18.3	1511.5	14	500	7200	19.4	23	15.6	1459.1	8	610	7200	20.2	23	12.08	1499.8	17
55	8	1.5	1	1001	1407.3	44	44	0	318.2	6	605	4161.8	44	44	0	630.2	1	1001	5378.7	44	44	0	1112.7	6
55	8	1.5	2	1001	1505.5	45	45	0	328.2	7	605	2413.2	45	45	0	355.9	1	1001	4296.1	45	45	0	814.6	7
55	8	1.5	3	1001	732.2	44	44	0	143.6	7	605	371.7	44	44	0	55.3	4	1001	380.1	44	44	0	70.6	7
55	8	1.5	4	1001	704.2	46	46	0	149.7	4	605	1383.6	46	46	0	220	2	1001	1273.3	46	46	0	245.4	4
55	8	1.5	5	1001	7200	43.1	47	8.3	1055.2	5	605	7200	44.3	47	5.8	1120.4	4	1001	7200	43.4	47	7.58	913.5	5
55	8	3.0	1	803	7200	25.4	32	20.5	991.1	5	605	7200	25.6	31	17.5	957.7	15	803	7200	26.3	32	17.73	865.9	4
55	8	3.0	2	803	7200	26.1	30	13	926.7	7	605	7200	25.2	31	18.6	1003.3	2	803	7200	25.2	30	15.88	896.6	6
55	8	3.0	3	803	7200	25.9	31	16.3	1005.1	9	605	7200	27.8	31	10.3	936.1	5	803	7200	27.5	31	11.24	965.2	8
55	8	3.0	4	803	7200	26.5	33	19.7	1022.2	7	605	7200	26.8	33	18.8	1042.2	1	803	7200	27	33	18.18	961.7	7
55	8	3.0	5	803	7200	26.6	34	21.8	989.7	8	605	7200	27.1	34	20.3	1061	4	803	7200	27.2	34	19.89	918.2	8
55	8	4.5	1	737	7200	19.7	24	17.9	992.7	6	605	7200	19.2	24	19.9	1150.5	2	737	7200	20.9	24	12.73	966	6
55	8	4.5	2	737	7200	19.6	23	14.9	1033	20	605	7200	19.2	23	16.4	1026.5	3	737	4361.8	23	23	0	776.9	23
55	8	4.5	3	737	7200	21.1	26	19	1012.9	5	605	7200	21.4	27	20.6	941.1	2	737	7200	21.7	26	16.48	1042.5	5
55	8	4.5	4	737	7200	18.8	23	18.5	1145.7	12	605	7200	19	23	17.4	1083.7	7	737	7200	20.7	23	10.14	1271.3	12
55	8	4.5	5	737	7200	18.5	25	25.9	1147.2	14	605	7200	19.3	25	23	974.2	8	737	7200	19.8	25	20.64	1063.5	17
60	8	1.5	1	1200	7200	45.7	51	10.3	740.5	7	720	7200	45.3	51	11.1	684.4	6	1200	7200	46	51	9.8	665.3	6
60	8	1.5	2	1200	7200	46.5	50	7.1	758.3	4	720	7200	47	50	6	773.1	1	1200	7200	46.5	50	7.08	747.1	5
60	8	1.5	3	1200	7200	49.3	52	5.1	887.9	6	720	7200	49	52	5.7	800.4	3	1200	7200	48.1	52	7.58	791.3	6
60	8	1.5	4	1200	7200	48.5	50	3	1000.7	4	720	3097.1	50	50	0	319.8	4	1200	4501.7	50	50	0	648.5	4
60	8	1.5	5	1200	7200	45.2	49	7.7	791.9	6	720	7200	44.8	49	8.5	709.9	3	1200	7200	45.5	49	7.1	793	6
60	8	3.0	1	960	7200	28.2	36	21.6	904.5	20	720	7200	28.6	36	20.6	775.2	3	960	7200	29.4	36	18.47	703.9	20
60	8	3.0	2	960	7200	28.8	36	20.1	683.8	6	720	7200	29	36	19.4	724.1	5	960	7200	29.4	36	18.43	779	5
60	8	3.0	3	960	7200	26.3	36	26.9	824.5	4	720	7200	27.5	36	23.7	810	2	960	7200	27.3	35	21.91	770.4	4
60	8	3.0	4	960	7200	29.1	34	14.4	665.6	8	720	7200	29.1	34	14.6	658.7	3	960	7200	29.8	34	12.27	685.8	8
60	8	3.0	5	960	7200	28.7	36	20.4	720.7	10	720	7200	29.9	36	16.9	718.4	0	960	7200	29.7	35	15.24	672.5	10
60	8	4.5	1	876	7200	20.6	27	23.7	895.6	21	720	7200	20.1	29	30.7	926.1	1	876	7200	22.7	27	16.07	744.1	24
60	8	4.5	2	876	7200	20.3	28	27.6	875.7	11	720	7200	20.5	28	26.9	833.2	5	876	7200	20.8	27	22.94	815.2	11

Continued on Next Page...

Table A.4 – Continued

Instances				USNP$_{(V,E,C)}$							USNP-X$_{(V,E,C)}$							USNP-Y$_{(V,E,C)}$						
m	C	ρ	#	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts	bin	CPU	lb	ub	gap	node	cuts
60	8	4.5	3	876	7200	22	29	24.3	910.7	5	720	7200	21.7	29	25.2	892.1	3	876	7200	23	29	20.76	818.7	6
60	8	4.5	4	876	7200	20.5	26	21.1	804.3	18	720	7200	20.5	27	24.1	783.4	4	876	7200	23	26	11.69	873.8	18
60	8	4.5	5	876	7200	19.6	28	29.9	861.7	19	720	7200	20	28	28.7	889.7	2	876	7200	21.2	28	24.4	819.5	19

Appendix B

Preliminary results on the separation of valid inequalities

Some preliminary studies concerning the separation of generalized k-cardinality tree inequalities are conducted here.

More precisely, even if we could not be able to derive a formal proof of the $\mathcal{NP}$-Hardness of the separation problem associated with k-cardinality forest inequalities (4.44), we present an heuristic procedure for solving such separation problem. In practice however, the number of cuts found by such procedure was irrelevant to the optimization process and for this reason it was not included in the developed Branch-and-Cut procedure.

In contrast, a formal proof of the $\mathcal{NP}$-Hardness of the separation problem associated with k-cover tree inequalities (4.47) is provided. On the other hand, a fast heuristic capable of yielding violated cuts remains a challenge to be tackled in future works.

B.1 Separation of k-cardinality forest inequalities

A simple adaptation of Algorithm 5 allows to derive an heuristic procedure for solving the separation problem associated with k-cardinality forest inequalities (4.44). Such procedure is described in Algorithm 7.

Algorithm 7 Separation of inequalities (4.44) by greedily construction of a $(C+1)$-cardinality forest

Input: An instance (V, E, C) of U-SNP, and a fractional solution $(\bar{x}, \bar{y})$ of the linear relaxation of $\text{USNP}_{(V,E,C)}$
Output: A "good" candidate k-cardinality tree inequality for cutting $(\bar{x}, \bar{y})$

```
for i = 1, ..., p do
  for j = 1, ..., n do
    for all e ∈ Δ_E(j) do
      w_e = x̄^i_e
    end for
    for all v ∈ V[Δ_E(j)] do
      c_v = ȳ^i_v
    end for
    Choose the edge e with most fractional weight.
    T = {e}
    for k = 1, ..., C do
      Let e_k ∈ arg max_{uv∈Δ_E(j)\T} {{w_uv − c_v : v ∈ V[T] and u ∉ V[T]} ∪
                                       {w_uv − 1 : v ∉ V[T] and u ∉ V[T]}}
      T = T ∪ {e_k}
    end for
    if Σ_{e∈T} x̄^i_e − Σ_{v∈V[T]} (deg_{G[T]}(v) − 1) ȳ^i_v > 0 then
      Add inequality Σ_{e∈T} x^i_e − Σ_{v∈V[T]} (deg_{G[T]}(v) − 1) y^i_v ≤ 0 as a cut
    end if
  end for
end for
```

B.2 Separation of k-cover tree inequalities

Let us recall the family of k-cover tree inequalities (4.47):

$$\sum_{e\in S} x^i_e - \sum_{v\in V[S]} (\deg_{G[S]}(v) - 1) y^i_v \leq 0 \qquad \forall i \in K, j \in V, S \subseteq E, \text{ such that } G[S] \text{ is a tree and } |S \cap \Delta_E(j)| = C + 1. \tag{4.47}$$

The family of k-cover tree inequalities (4.47) clearly has an exponential number of inequalities as it generalizes the family of k-cardinality tree inequalities (4.40). In addition, it is easy to observe that when (V, E, C) is an instance of Intersection U-SNP, these two families of inequalities (*i.e.*, (4.47) and (4.40)) are equivalent. It follows directly that the separation problem associated with k-cover tree inequalities (4.47) is at least as hard as the one associated with k-cardinality tree inequalities (4.40).

Theorem B.1 - Separation of k-cover tree inequalities

The separation problem associated with k-cover tree inequalities (4.47) *is* $\mathcal{NP}$*-Hard.*

■

References

[1] T. Achterberg and C. Raack. The mcf-separator: detecting and exploiting multi-commodity flow structures in mips. *Mathematical Programming Computation*, 2(2):125–165, 2010.

[2] M. Agrawal, N. Kayal, and N. Saxena. Primes is in p. *Annals of mathematics*, pages 781–793, 2004.

[3] O. Amini, S. Pérennes, and I. Sau. Hardness and approximation of traffic grooming. *Theoretical Computer Science*, 410(38-40):3751–3760, 2009.

[4] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Tsp cuts which do not conform to the template paradigm. In *Computational combinatorial optimization*, pages 261–303. Springer, 2001.

[5] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Implementing the dantzig-fulkerson-johnson algorithm for large traveling salesman problems. *Mathematical programming*, 97(1-2):91–153, 2003.

[6] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Concorde tsp solver. `http://www.tsp.gatech.edu/concorde/`, 2006. Accessed on May 04, 2019.

[7] E. Balas. Facets of the knapsack polytope. *Mathematical programming*, 8(1): 146–164, 1975.

[8] E. Balas. Disjunctive programming. In *Annals of Discrete Mathematics*, volume 5, pages 3–51. Elsevier, 1979.

[9] E. Balas and N. Christofides. A restricted lagrangean approach to the traveling salesman problem. *Mathematical Programming*, 21(1):19–46, 1981.

[10] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming*, 58(1-3):295–324, 1993.

[11] M. L. Balinski. On finding integer solutions to linear programs. Technical report, Mathematica Princeton NJ, 1964.

[12] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.

[13] A. Beaudry, G. Laporte, T. Melo, and S. Nickel. Dynamic transportation of patients in hospitals. *OR spectrum*, 32(1):77–107, 2010.

[14] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

[15] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31, 2007.

[16] J.-C. Bermond, L. Braud, and D. Coudert. Traffic grooming on the path. *Theoretical Computer Science*, 384(2-3):139–151, 2007.

[17] L. Bodin. Routing and scheduling of vehicles and crews, the state of the art. *Comput. Oper. Res.*, 10(2):63–211, 1983.

[18] I. Borosh and L. B. Treybig. Bounds on positive integral solutions of linear diophantine equations. *Proceedings of the American Mathematical Society*, 55 (2):299–304, 1976.

[19] K. Braekers and A. A. Kovacs. A multi-period dial-a-ride problem with driver consistency. *Transportation Research Part B: Methodological*, 94:355–377, 2016.

[20] K. Braekers, A. Caris, and G. K. Janssens. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological*, 67:166–186, 2014.

[21] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016.

[22] J. Bramel and D. Simchi-Levi. Set-covering-based algorithms for the capacitated vrp. In *The vehicle routing problem*, pages 85–108. SIAM, 2002.

[23] N. Brauner and P. Lemaire. A set-covering approach for sonet network design. rapport technique 62. *Les Cahiers du Laboratoire Leibniz-IMAG*, 2002.

[24] N. Brauner, Y. Crama, G. Finke, P. Lemaire, and C. Wynants. Approximation algorithms for the design of SDH/SONET networks. *RAIRO-Operations Research*, 37(4):235–247, 2003.

[25] S. Bsaybes. *Modèles et algorithmes de gestion de flottes de véhicules vipa*. PhD thesis, Ph. D. thesis, Université Clermont Auvergne, 2017.

[26] S. Bsaybes, A. Quilliot, and A. K. Wagler. Fleet management for autonomous vehicles: Online pdp under special constraints. *arXiv preprint arXiv:1703.10565*, 2017.

[27] T. Bunsen, P. Cazzola, M. Gorner, L. Paoli, S. Scheffer, R. Schuitmaker, J. Tattini, and J. Teter. Global ev outlook 2018: Towards cross-modal electrification. 2018.

[28] J. Carlson, J. A. Carlson, A. Jaffe, and A. Wiles. *The millennium prize problems*. American Mathematical Soc., 2006.

[29] R. Carvajal, S. Ahmed, G. Nemhauser, K. Furman, V. Goel, and Y. Shao. Using diversification, communication and parallelism to solve mixed-integer linear programs. *Operations Research Letters*, 42(2):186–189, 2014.

[30] T. Christof, A. Löbel, and S. Schenker. Polyhedron representation transformation algorithm. `http://porta.zib.de/`, 1997. Accessed on May 06, 2019.

[31] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete mathematics*, 4(4):305–337, 1973.

[32] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[33] V. Chvatal, V. Chvatal, et al. *Linear programming*. Macmillan, 1983.

[34] Clay Mathematics Institute. Millennium problems - p vs np problem. `http://claymath.org/millennium-problems/p-vs-np-problem`, 2018. Accessed on Feb 01, 2019.

[35] M. Conforti, G. Cornuéjols, and G. Zambelli. Polyhedral approaches to mixed integer linear programming. In *50 years of integer programming 1958-2008*, pages 343–385. Springer, 2010.

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

[37] J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006.

[38] J.-F. Cordeau and G. Laporte. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(2):89–101, 2003.

[39] J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003.

[40] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153(1):29–46, 2007.

[41] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. Technical report, Carnegie-mellon univ pittsburgh pa management sciences research group, 1983.

[42] C. E. Cortés, M. Matamala, and C. Contardo. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3):711–724, 2010.

[43] H. Crowder, E. L. Johnson, and M. W. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.

[44] G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. *New York*, 1951.

[45] G. B. Dantzig. *Linear Programming and Extensions*. RAND Corporation, 1963.

[46] G. B. Dantzig. Origins of the simplex method. Technical report, Stanford University CA Systems Optimization Lab, 1987.

[47] G. B. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[48] F. De Rugy and E. Borne. Projet de loi d'orientation des mobilités (tret1821032l). `https://www.senat.fr/leg/pjl18-157rec.html`, 2018. Accessed on Jun 04, 2019.

[49] A. Delbosc. The role of well-being in transport policy. *Transport Policy*, 23: 25–33, 2012.

[50] S. Deleplanque, A. Luisa Marques Fernandes, L. Bernardes Real, R. Saraiva De Camargo, and A. Quilliot. The minimization of the number of stops problem - management of an autonomous vehicle fleet. 03 2014.

[51] S. Deleplanque, A. Quilliot, and B. Bernay. Branch and price for a reliability oriented darp model. pages 081–086. IEEE, 2014.

[52] B. T. Denton, A. J. Miller, H. J. Balasubramanian, and T. R. Huschka. Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations research*, 58(4-part-1):802–816, 2010.

[53] B. Dezső, A. Jüttner, and P. Kovács. Lemon–an open source c++ graph template library. *Electronic Notes in Theoretical Computer Science*, 264(5): 23–45, 2011.

[54] R. Diestel. *Graph theory*. Springer Publishing Company, Incorporated, 2018.

[55] Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European journal of operational research*, 54(1):7–22, 1991.

[56] R. Dutta and G. N. Rouskas. Traffic grooming in wdm networks: Past and future. *IEEE network*, 16(6):46–56, 2002.

[57] M. E. Dyer and A. M. Frieze. On the complexity of partitioning graphs into connected subgraphs. *Discrete Applied Mathematics*, 10(2):139–153, 1985.

[58] M. E. Dyer and A. M. Frieze. Planar 3dm is np-complete. *Journal of Algorithms*, 7(2):174–184, 1986.

[59] J. Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.

[60] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17 (3):449–467, 1965.

[61] E. Egerváry Combinatorial Optimization Research Group. Lemon graph library. `https://lemon.cs.elte.hu/trac/lemon`, 2003. Accessed on May 04, 2019.

[62] E. Egerváry Combinatorial Optimization Research Group. Lemon graph library documentation: lemon::dijkstra. `http://lemon.cs.elte.hu/pub/doc/latest-svn/a00193.html`, 2009. Accessed on May 06, 2019.

[63] E. Egerváry Combinatorial Optimization Research Group. Lemon graph library documentation: lemon::kruskal. `http://lemon.cs.elte.hu/pub/doc/latest-svn/a00346.html`, 2009. Accessed on May 04, 2019.

[64] D. C. Ehlers. Mobility of the future – connected, autonomous, shared, electric. In *30th International AVL Conference "Engine & Environment"*, 2018.

[65] European Automobile Manufacturers Association. The 2030 urban mobility challenge: Acea's contribution. `https://www.acea.be/uploads/publications/The_2030_Urban_Mobility_Challenge_-_ACEAs_Contribution_May_2016.pdf`, 2016. Accessed on Jul 06, 2019.

[66] D. J. Fagnant and K. Kockelman. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015.

[67] L. Feng, E. Miller-Hooks, P. Schonfeld, and M. Mohebbi. Optimizing ridesharing services for airport access. *Transportation Research Record*, 2467(1):157–167, 2014.

[68] M. Fischetti and M. Monaci. Exploiting erraticism in search. *Operations Research*, 62(1):114–122, 2014.

[69] M. Fischetti and P. Toth. A polyhedral approach to the asymmetric traveling salesman problem. *Management Science*, 43(11):1520–1536, 1997.

[70] M. Fischetti, H. W. Hamacher, K. Jørnsten, and F. Maffioli. Weighted k-cardinality trees: Complexity and polyhedral structure. *Networks*, 24(1):11–21, 1994.

[71] D. Gale, H. Kuhn, and A. Tucker. Linear programming and the theory of games, activity analysis of production and allocation, ed. tc koopmans, 1951.

[72] P. Gao, H.-W. Kaas, D. Mohr, and D. Wee. Automotive revolution–perspective towards 2030: How the convergence of disruptive technology-driven trends could transform the auto industry. *Advanced Industries, McKinsey & Company*, 2016.

[73] T. Garaix, C. Artigues, D. Feillet, and D. Josselin. Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. *Computers & Operations Research*, 38(10):1435–1442, 2011.

[74] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[75] M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980.

[76] O. Gerstel, R. Ramaswami, and G. H. Sasaki. Cost-effective traffic grooming in WDM rings. *IEEE/ACM Transactions On Networking*, 8(5):618–630, 2000.

[77] A. Ghouila-Houri. Caractérisation des matrices totalement unimodulaires. *Comptes Redus Hebdomadaires des Séances de l'Académie des Sciences (Paris)*, 254:1192–1194, 1962.

[78] O. Goldschmidt, D. S. Hochbaum, A. Levin, and E. V. Olinick. The sonet edge-partition problem. *Networks: An International Journal*, 41(1):13–23, 2003.

[79] R. E. Gomory. An algorithm for the mixed integer problem. Technical report, RAND CORP SANTA MONICA CA, 1960.

[80] R. E. Gomory et al. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical society*, 64(5):275–278, 1958.

[81] M. Grötschel and M. W. Padberg. On the symmetric travelling salesman problem i: inequalities. *Mathematical Programming*, 16(1):265–280, 1979.

[82] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[83] M. Grötschel, M. Jünger, and G. Reinelt. On the acyclic subgraph polytope. *Mathematical Programming*, 33(1):28–42, 1985.

[84] M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Mathematical programming*, 33(1):43–60, 1985.

[85] T. Gschwind and S. Irnich. Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. *Transportation Science*, 49 (2):335–354, 2014.

[86] Z. Gu, G. L. Nemhauser, and M. W. Savelsbergh. Lifted cover inequalities for 0-1 integer programs: Computation. *INFORMS Journal on Computing*, 10 (4):427–437, 1998.

[87] Z. Gu, G. L. Nemhauser, and M. W. Savelsbergh. Lifted flow cover inequalities for mixed 0-1 integer programs. *Mathematical Programming*, 85(3):439–467, 1999.

[88] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[89] I. Heller and C. Tompkins. An extension of a theorem of dantzig's. *Linear inequalities and related systems*, 38:247–254, 1956.

[90] S. Ho, S. C. Nagavarapu, R. R. Pandi, and J. Dauwels. An improved tabu search heuristic for static dial-a-ride problem. *arXiv preprint arXiv:1801.09547*, 2018.

[91] S. C. Ho, W. Szeto, Y.-H. Kuo, J. M. Leung, M. Petering, and T. W. Tou. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 111:395–421, 2018.

[92] A. J. Hoffman and J. B. Kruskal. Integral boundary points of convex polyhedra. *Linear Inequalities and Related Systems (H.W. Kuhn and A.J. Tucker, eds)*, pages 223–246, 1956.

[93] A. J. Hoffman and H. Kuhn. Systems of distinct representatives and linear programming. *The American Mathematical Monthly*, 63(7):455–460, 1956.

[94] K. L. Hoffman and M. W. Padberg. Improving lp-representations of zero-one linear programs for branch-and-cut. *ORSA Journal on Computing*, 3(2): 121–134, 1991.

[95] S. Huang, R. Dutta, and G. N. Rouskas. Traffic grooming in path, star, and tree networks: Complexity, bounds, and algorithms. *IEEE Journal on Selected Areas in Communications*, 24(4), 2006.

[96] K. Jakob and P. M. Pruzan. The simple plant location problem: Survey and synthesis. *European journal of operational research*, 12:36–81, 1983.

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

[98] R. M. Jorgensen, J. Larsen, and K. B. Bergvinsdottir. Solving the dial-a-ride problem using genetic algorithms. *Journal of the operational research society*, 58(10):1321–1331, 2007.

[99] D. Junglas. What is really disabled by control callbacks. `https://www.ibm.com/developerworks/community/forums/html/topic?id=5d8dc682-93c6-4bf5-b28d-aa0427b0e47d`, 2018. Accessed on Feb 25,2019.

[100] V. Kaibel and M. E. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 114(1):1–36, 2008.

[101] V. Kaibel, M. Peinhardt, and M. E. Pfetsch. Orbitopal fixing. *Discrete Optimization*, 8(4):595–610, 2011.

[102] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[103] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[104] L. G. Khachiyan. A polynomial algorithm in linear programming. In *Doklady Academii Nauk SSSR*, volume 244, pages 1093–1096, 1979.

[105] S.-W. Kim, G.-P. Gwon, S.-T. Choi, S.-N. Kang, M.-O. Shin, I.-S. Yoo, E.-D. Lee, E. Frazzoli, and S.-W. Seo. Multiple vehicle driving control for traffic flow efficiency. In *2012 IEEE Intelligent Vehicles Symposium*, pages 462–468. IEEE, 2012.

[106] D. Kirchler and R. W. Calvo. A granular tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B: Methodological*, 56:120–135, 2013.

[107] V. Klee and G. J. Minty. How good is the simplex algorithm. Technical report, Washington University Seattle Dept. of Mathematics, 1970.

[108] KPMG International. Global automotive executive survey 2019,. `https://home.kpmg/xx/en/home/insights/2019/01/global\unhbox\voidb@x\hbox{-}automotive\unhbox\voidb@x\hbox{-}executive\unhbox\voidb@x\hbox{-}survey\unhbox\voidb@x\hbox{-}2019.html`, 2019. Accessed on Jun 04, 2019.

[109] KPMG International. Mobility 2030: Transforming the mobility landscape. `https://assets.kpmg/content/dam/kpmg/xx/pdf/2019/02/mobility-2030-transforming-the-mobility-landscape.pdf`, 2019. Accessed on Jun 04, 2019.

[110] M. S. Krishnamoorthy. An np-hard problem in bipartite graphs. *ACM SIGACT News*, 7(1):26–26, 1975.

[111] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1): 48–50, 1956.

[112] A. A. Kuehn and M. J. Hamburger. A heuristic program for locating warehouses. *Management science*, 9(4):643–666, 1963.

[113] R. E. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM (JACM)*, 22(1):155–171, 1975.

[114] R. Lahyani, M. Khemakhem, and F. Semet. Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1–14, 2015.

[115] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.

[116] T. Laseter, A. Tipping, and F. Duiven. The rise of the last-mile exchange. *PWC Strategy+ Business*, 92:30, 2018.

[117] Y. Lee, H. D. Sherali, J. Han, and S.-i. Kim. A branch-and-cut algorithm for solving an intraring synchronous optical network design problem. *Networks: An International Journal*, 35(3):223–232, 2000.

[118] P. Lemaire. Optimisation de réseaux sonet/sdh: éléments théoriques et résolution pratique. *Mémoire de DEA*, 2001.

[119] L. A. Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

[120] A. D. Little. *The Future of Mobility 3.0*. Little, Arthur D, 2018. URL `https://www.adlittle.com/futuremobilitylab/assets/file/180330_Arthur_D.Little_&_UITP_Future_of_Mobility_3_study.pdf`.

[121] M. Liu, Z. Luo, and A. Lim. A branch-and-cut algorithm for a realistic dial-a-ride problem. *Transportation Research Part B: Methodological*, 81:267–288, 2015.

[122] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM journal on optimization*, 1(2):166–190, 1991.

[123] Z. Luo, M. Liu, and A. Lim. A two-phase branch-and-price-and-cut for a dial-a-ride problem in patient transportation. *Transportation Science*, 53(1): 113–130, 2018.

[124] A. S. Manne. Plant location under economies-of-scale—decentralization and computation. *Management Science*, 11(2):213–235, 1964.

[125] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94(1):71–90, 2002.

[126] F. Margot. Exploiting orbits in symmetric ilp. *Mathematical Programming*, 98(1-3):3–21, 2003.

[127] F. Margot. Symmetry in integer linear programming. In *50 Years of Integer Programming 1958-2008*, pages 647–686. Springer, 2010.

[128] N. Marković, R. Nair, P. Schonfeld, E. Miller-Hooks, and M. Mohebbi. Optimizing dial-a-ride services in maryland: benefits of computerized routing and scheduling. *Transportation Research Part C: Emerging Technologies*, 55: 156–165, 2015.

[129] M. A. Masmoudi, K. Braekers, M. Masmoudi, and A. Dammak. A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. *Computers & operations research*, 81:1–13, 2017.

[130] S. Masuyama and T. Ibaraki. Chain packing in graphs. *Algorithmica*, 6(1-6): 826–839, 1991.

[131] P. Merlin. Géographie des transports. *Que sais-je?*, 1992.

[132] R. R. Meyer. On the existence of optimal solutions to integer and mixed-integer programming problems. *Mathematical Programming*, 7(1):223–235, 1974.

[133] E. Modiano. Traffic grooming in wdm networks. *IEEE communications Magazine*, 39(7):124–129, 2001.

[134] Y. Molenbruch, K. Braekers, and A. Caris. Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, 259(1-2):295–325, 2017.

[135] T. Motzkin. The assignment problem. In *Proc. Symposia in Applied Mathematics*, volume 6, pages 109–125, 1956.

[136] G. L. Nemhauser and L. E. Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6(1):48–61, 1974.

[137] G. L. Nemhauser and L. E. Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

[138] J. Ostrowski, M. F. Anjos, and A. Vannelli. *Symmetry in scheduling problems*. Citeseer, 2010.

[139] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. *Mathematical Programming*, 126(1):147–178, 2011.

[140] U. Ozguner, C. Stiller, and K. Redmill. Systems for safety and autonomous behavior in cars: The darpa grand challenge experience. *Proceedings of the IEEE*, 95(2):397–412, 2007.

[141] M. W. Padberg. On the facial structure of set packing polyhedra. *Mathematical programming*, 5(1):199–215, 1973.

[142] M. W. Padberg and S. Hong. On the symmetric travelling salesman problem: a computational study. In *Combinatorial Optimization*, pages 78–107. Springer, 1980.

[143] M. W. Padberg, T. J. Van Roy, and L. A. Wolsey. Valid linear inequalities for fixed charge problems. *Operations Research*, 33(4):842–861, 1985.

[144] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of computer and system sciences*, 43(3):425–440, 1991.

[145] J. Paquette, J.-F. Cordeau, G. Laporte, and M. M. Pascoal. Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological*, 52:1–16, 2013.

[146] S. N. Parragh. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies*, 19(5):912–930, 2011.

[147] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery models part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008.

[148] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[149] S. N. Parragh, K. F. Doerner, and R. F. Hartl. Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, 37(6):1129–1138, 2010.

[150] S. N. Parragh, J.-F. Cordeau, K. F. Doerner, and R. F. Hartl. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. *OR spectrum*, 34(3):593–633, 2012.

[151] V. Pimenta, A. Quilliot, H. Toussaint, and D. Vigo. Models and algorithms for reliability-oriented dial-a-ride with autonomous electric vehicles. *European Journal of Operational Research*, 257(2):601–613, 2017.

[152] R. C. Prim. Shortest connection networks and some generalizations. *Bell system technical journal*, 36(6):1389–1401, 1957.

[153] W. R. Pulleyblank. *Faces of Matching Polyhedra, Univ. of Waterloo, Dept. Combinatorics and Optimization*. PhD thesis, Ph. D. Thesis, 1973.

[154] Y. Qu and J. F. Bard. A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. *Transportation Science*, 49(2):254–270, 2014.

[155] RAC Foundation. Spaced out: Perspectives on parking policy. `https://www.racfoundation.org/assets/rac_foundation/content/downloadables/spaced_out-bates_leibling-jul12.pdf`, 2012. Accessed on Jun 04, 2019.

[156] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees—short or small. *SIAM Journal on Discrete Mathematics*, 9 (2):178–200, 1996.

[157] L. B. Reinhardt, T. Clausen, and D. Pisinger. Synchronized dial-a-ride transportation of disabled passengers at airports. *European Journal of Operational Research*, 225(1):106–117, 2013.

[158] J.-P. Rodrigue, C. Comtois, and B. Slack. *The geography of transport systems*. Routledge, 2016.

[159] S. Ropke and J.-F. Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009.

[160] S. Ropke, J.-F. Cordeau, and G. Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks: An International Journal*, 49(4):258–272, 2007.

[161] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6 (3):563–581, 1977.

[162] A. Sabharwal, H. Samulowitz, and C. Reddy. Guiding combinatorial optimization with uct. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 356–361. Springer, 2012.

[163] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[164] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[165] M. Shalom, W. Unger, and S. Zaks. On the complexity of the traffic grooming problem in optical networks. In *International Conference on Fun with Algorithms*, pages 262–271. Springer, 2007.

[166] H. D. Sherali and J. C. Smith. Improving discrete model representations via symmetry considerations. *Management Science*, 47(10):1396–1407, 2001.

[167] H. D. Sherali, J. C. Smith, and Y. Lee. Enhanced model representations for an intra-ring synchronous optical network design problem allowing demand splitting. *INFORMS Journal on Computing*, 12(4):284–298, 2000.

[168] S. Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report, NHTSA's National Center for Statistics and Analysis, 2015.

[169] H. M. Stark. *An introduction to number theory*. Markham Publishing Company Chicago, Illinois, 1970.

[170] Statista. Retail e-commerce sales worldwide from 2014 to 2021 (in billion u.s. dollars). `https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/`, 2018. Accessed on Jul 05, 2019.

[171] A. S. Sutter, F. Vanderbeck, and L. A. Wolsey. Optimal placement of add/drop multiplexers: heuristic and exact algorithms. *Operations Research*, 46(5):719–728, 1998.

[172] A. Talebpour and H. S. Mahmassani. Influence of connected and autonomous vehicles on traffic flow stability and throughput. *Transportation Research Part C: Emerging Technologies*, 71:143–163, 2016.

[173] P. Toth and D. Vigo. *The vehicle routing problem*. SIAM, 2002.

[174] P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[175] United Nations Department of Economic and Social Affairs. 2018 revision of world urbanization prospects. `https://www.un.org/development/desa/publications/2018-revision-of-world-urbanization-prospects.html`, 2018. Accessed on May 06, 2019.

[176] T. J. Van Roy and L. A. Wolsey. Valid inequalities and separation for uncapacitated fixed charge networks. *Operations Research Letters*, 4(3):105–112, 1985.

[177] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[178] J. Von Neumann. *Discussion of a maximum problem, John von Neumann: Collected Works, Vol. VI*. Pergamon, 1963.

[179] L. A. Wolsey. Further facet generating procedures for vertex packing polytopes. *Mathematical Programming*, 11(1):158–163, 1976.

[180] L. A. Wolsey. Valid inequalities for 0–1 knapsacks and mips with generalised upper bound constraints. *Discrete Applied Mathematics*, 29(2-3):251–261, 1990.

[181] L. A. Wolsey. *Integer Programming. Series in Discrete Mathematics and Optimization*. Wiley-Interscience New Jersey, 1998.

[182] World Commission on Environment and Development. Our common future. Oxford University Press. Oxford, 1987.

[183] T.-H. Wu. *Fiber network service survivability*. Artech House, Inc., 1992.

[184] Z. Xiang, C. Chu, and H. Chen. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European journal of operational research*, 174(2):1117–1139, 2006.

[185] Z. Xiang, C. Chu, and H. Chen. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. *European Journal of Operational Research*, 185(2):534–551, 2008.

[186] K. Zhu and B. Mukherjee. A review of traffic grooming in wdm optical networks: Architectures and challenges. *Optical Networks Magazine*, 4(2):55–64, 2003.

[187] I. Zidi, K. Mesghouni, K. Zidi, and K. Ghedira. A multi-objective simulated annealing for the multi-criteria dial a ride problem. *Engineering Applications of Artificial Intelligence*, 25(6):1121–1131, 2012.

Résumé en français

Introduction et contexte

L'optimisation combinatoire est un domaine de recherche important des mathématiques appliquées, de l'algorithmique et de l'informatique théorique. Un problème d'optimisation combinatoire consiste à choisir le meilleur élément dans un ensemble fini. Ce genre de problèmes se posent habituellement – mais pas exclusivement – dans des situations pratiques arrivant dans les télécommunications, le transport, la logistique, la finance entre autres.

Formellement, étant donnés un ensemble fini d'éléments $N = \{1, \ldots, n\}$, une famille $\mathcal{F}$ de sous-ensembles $F \subseteq N$ et une fonction de coût $c : N \to \mathbb{R}$, un problème d'optimisation combinatoire consiste à déterminer un élément $F \in \mathcal{F}$ tel que la somme $c(F)$ des coûts de ses éléments soit maximum (ou minimum). De tels problèmes peuvent paraître faciles à résoudre à première vue. Par exemple, une approche triviale serait d'énumérer tous les éléments dans $\mathcal{F}$ – rappelons que $\mathcal{F}$ est fini – et choisir simplement le meilleur élément selon la fonction de coût c. Cependant, cette famille de sous-ensembles est souvent composée d'un très grand (*e.g.*, exponentiel dans la taille de N) nombre d'éléments, ce qui empêche la résolution du problème avec une telle approche dans un temps raisonnable en utilisant un ordinateur.

Cette remarque a conduit des scientifiques à étudier et développer des moyens plus intelligents et élégants pour la résolution de ce genre de problèmes. La programmation linéaire et la programmation linéaire en nombres entiers ainsi que les algorithmes basés sur le principe de la programmation dynamique et/ou les relations min-max entre différentes structures combinatoires sont des exemples de telles méthodes. Cependant, malgré tout le savoir développé à travers l'histoire de l'informatique, des algorithmes performants pour certains problèmes combinatoires restent inconnus. Autrement dit, cela signifie qu'aucun algorithme avec un temps d'exécution borné par une fonction polynomiale en la taille de l'entrée n'est connu.

Parmi ces problèmes difficiles à résoudre, une certaine classe de problèmes s'est démarquée : les problèmes appelés $\mathcal{NP}$-Difficile. De façon informelle, un problème

appartient à la classe $\mathcal{NP}$-Difficile s'il est au moins aussi difficile à résoudre que n'importe quel autre problème présent dans la classe $\mathcal{NP}$, et un problème appartient à la classe $\mathcal{NP}$ s'il peut être formulé comme une question ayant pour réponse oui ou non de façon que ses solutions peuvent être vérifiées de manière efficace (*i.e.*, en temps polynomial). Par rapport aux problèmes combinatoires, cela revient à dire qu'un problème combinatoire appartient à $\mathcal{NP}$ s'il peut être formulé par une question demandant l'existence ou non d'un élément $F \in \mathcal{F}$ tel que $c(F)$ vaut au moins (ou au plus) une valeur $B \in \mathbb{R}$ donnée, et que $c(F)$ peut être calculé en temps polynomial.

Il est bien connu que s'il existe un algorithme efficace pour résoudre un seul problème $\mathcal{NP}$-Difficile, alors tous les problèmes appartenant à la classe $\mathcal{NP}$-Difficile peuvent aussi être résolus en temps polynomial. Telle propriété a guidé les chercheurs en informatique à imposer une dichotomie entre les problèmes efficacement vérifiables (classe $\mathcal{NP}$) et les problèmes efficacement solubles (classe $\mathcal{P}$). L'équivalence ou non de ces deux classes n'a pas pu être déterminée jusqu'à présent et représente ainsi une des principales questions ouvertes dans le domaine de l'informatique théorique. Il est cependant communément accepté que $\mathcal{P} \neq \mathcal{NP}$.

Le besoin d'algorithmes performants capables de résoudre des problèmes réels n'a jamais cessé d'augmenter depuis la construction des premiers ordinateurs dans les années 40. Dans ce scénario, lorsqu'un nouveau problème combinatoire est proposé, l'une des premières questions que l'on se pose tourne autour de sa complexité. En effet, si l'on est capable de prouver son appartenance à $\mathcal{NP}$-Difficile, alors un algorithme efficace pour résoudre ce problème est peu probable d'être concevable. Par ailleurs, les problèmes combinatoires $\mathcal{NP}$-Difficiles possèdent plusieurs applications en pratique, et par conséquent il est insensé et contre-productive de les ignorer. Pour ces problèmes on doit plutôt se concentrer à les résoudre non pas à travers un algorithme polynomial mais à travers un algorithme approprié (*i.e.*, aussi performant que possible).

L'approche polyédrale, développée par Jack Edmonds dans les années 60 (voir Edmonds 1965a) sur le problème du couplage maximal, combine les idées de la programmation linéaire et de la programmation linéaire en nombres entiers avec les concepts de la géométrie discrète et de l'algorithmique afin de résoudre des problèmes combinatoires. Elle consiste à formuler un problème combinatoire comme un programme linéaire dont les contraintes sont les inégalités linéaires décrivant l'enveloppe convexe des solutions réalisables.

Toutefois, une description explicite et complète d'un tel polyèdre peut être difficile à trouver et souvent il est nécessaire un nombre exponentiel d'inégalités pour ce

faire. Heureusement, une description partielle de l'enveloppe convexe peut permettre de trouver la solution optimale d'un problème combinatoire. Dans cette optique, les travaux de Grötschel, Lovász et Schrijver 1981 jouent un rôle très important en précisant qu'un problème combinatoire peut être résolu en temps polynomial si et seulement si il existe un algorithme polynomial capable de résoudre le problème de séparation associé. Ce dernier consiste, étant donné une solution x, à déterminer si x appartient au polyèdre, et sinon à fournir une contrainte qui sépare x du polyèdre. Dès lors, pour les problèmes combinatoires $\mathcal{NP}$-Difficiles, on peut s'attendre à ce que le problème de séparation associé soit aussi $\mathcal{NP}$-Difficile. Dans ces cas, l'approche polyédrale peut être encore utile en utilisant des heuristiques pour résoudre la séparation. Telle approche peut être exploité pour obtenir des inégalités valides capables de renforcer la relaxation linéaire de la formulation concernée.

Les approches polyédrales ont suscité une grande attention de la part de la communauté scientifique au cours des cinquante dernières années tant d'un point de vue théorique que pratique, et ont fait preuve d'efficacité en ce qui concerne la résolution optimale de problèmes combinatoires difficiles. En effet, plusieurs d'entre les algorithmes d'optimisation les plus performants bénéficient d'une approche polyédrale. Parmi ces algorithmes, l'un des plus célèbres est le Concorde TSP Solver, un algorithme basé sur le principe des Coupes et Branchements [1] traitant le traditionnel problème du voyageur de commerce [2] (voir Applegate et al. 2001 ; Applegate et al. 2006).

Dans cette dissertation, nous étudions un nouveau problème combinatoire appelé Problème de Minimisation du Nombre d'Arrêts – Stop Number Problem en anglais – découlant de la gestion d'un système de transport à la demande utilisant une flotte de véhicules électriques identiques et autonomes. En effet, des changements importants sont en train de se produire dans la façon dont les gens se déplacent d'un point à l'autre. Les réglementations plus restrictives en termes d'émissions de CO_2, alliées à la croissance extraordinaire de l'utilisation des smartphones et des services en ligne, induisent des nouvelles tendances dans le secteur des transports. Selon Gao et al. 2016, dans le cadre de l'innovation de la mobilité et du transport, quatre tendances majeures méritent d'être soulignées : la diversification des systèmes de mobilité, la conduite autonome, l'électrification des moteurs et la connectivité entre les ressources présents dans le système de transport.

Le projet VIPAFLEET apparaît dans ce scénario contribuant à une mobilité plus durable en se concentrant sur le développement de modèles mathématiques et algorithmes capables d'aider à la gestion des flottes de véhicules autonomes et

1. Branch-and-Cut, en anglais.
2. Travelling Salesman Problem (TSP), en anglais.

électriques appelés VIPA [3]. VIPA est un véhicule électrique développé par Ligier [4] et Easymile [5], et conçu pour fonctionner de manière complètement autonome (*i.e.*, sans aucune assistance humaine), notamment dans des environnements dits fermés ou semi-ouverts comme des sites commerciaux ou industriels, des campus universitaires et des complexes médicaux de grandes dimensions.

Les VIPA sont conçus pour opérer dans des différents modes de fonctionnement :

— **Mode tramway :** Les véhicules circulent toujours dans la même direction et s'arrêtent une fois sollicités.

— **Mode ascenseur :** Les véhicules circulent comme un ascenseur horizontal, voyageant dans un trajet linéaire prédéfini et réagissant aux demandes des clients pour changer de direction.

— **Mode taxi :** Les véhicules desservent les demandes des clients en choisissant de manière intelligente leurs itinéraires dans un réseau connexe.

Dans cette thèse nous nous intéressons à la gestion de tels véhicules fonctionnant en mode tramway. Par ailleurs, nous nous concentrons sur la version hors ligne de cette gestion, c'est-à-dire, toutes les demandes des clients sont connues à l'avance. Pour des études sur d'autres modes de fonctionnement en version en ligne, le lecteur est invité à consulter les travaux de Bsaybes 2017.

Le travail de recherche effectué au cours de cette thèse a été financé par le Laboratoire d'Excellence (Labex) IMobS3 [6].

Définition du problème

Le Problème de Minimisation du Nombre d'Arrêts est issu de la gestion des véhicules VIPA fonctionnant en mode tramway. Dans ce système, des véhicules identiques circulent à travers un circuit prédéfini avec des stations déterminés, en répondant à des sollicitations de clients pour un trajet entre une station de départ et une station d'arrivée. Nous remarquons que plusieurs clients peuvent partager les mêmes stations de départ et/ou d'arrivée.

À cause des restrictions d'infrastructure, les stations ne se situent pas à l'intérieur du circuit mais sont attachées à ce dernier (voir Figure 1) Cette particularité cause un impact sur la gestion de la flotte de véhicules. En effet, pour répondre à la demande d'un client, le véhicule doit ralentir et réaliser une déviation de son parcours

3. Véhicule Individuel Public Autonome.
4. `https://www.ligier.fr/nos-gammes/ez10.html`, consulté le 07-06-2019.
5. `https://easymile.com/`, consulté le 07-06-2019.
6. Innovative Mobility : Smart and Sustainable Solutions

d'origine. Ces déviations augmentent le temps de trajet des clients à bord du véhicule ainsi que la consommation d'énergie de sa batterie. Par conséquent, améliorer la qualité de service proposé correspond à minimiser le nombre d'arrêts effectués par la flotte de véhicules.

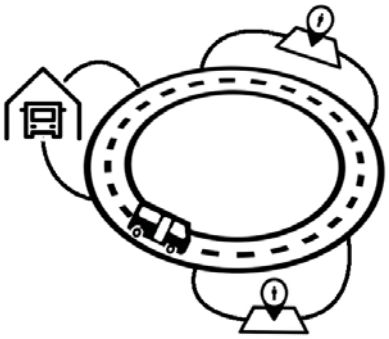


Figure 1 – Schéma du circuit

Le Problème de Minimisation du Nombre d'Arrêts consiste à affecter un véhicule à chaque client de façon à ce que la capacité de chaque véhicule soit respectée tout au long du circuit. L'objectif est de minimiser le nombre d'arrêts effectués en prenant en charge les demandes. Nous remarquons que dans la recherche d'une solution optimale, un véhicule peut se permettre de réaliser plusieurs tours du circuit avant desservir une demande. Par contre, une fois qu'un client est récupéré, il ne doit pas rester plus d'un tour complet à bord du véhicule. Les tours réalisés avant la prise en charge d'une demande sont appelés *tours d'attente*. Dans le but d'assurer une qualité de service et faire face aux fenêtres de temps accordées par les clients, le nombre maximal de tours d'attente est borné par un paramètre donné $H \geq 0 \in \mathbb{N}$. Enfin, la demande d'un client peut solliciter un ou plusieurs sièges dans un même véhicule. Dans ce sens, une demande est caractérisée par sa station de départ, sa station d'arrivée et une charge spécifiant le nombre de sièges demandés.

Cette thèse se concentre sur l'étude et la résolution du Problème de Minimisation du Nombre d'Arrêts quand aucun tour d'attente n'est accordé, c'est-à-dire $H = 0$, et chaque demande ne peut solliciter qu'un seul siège dans chaque véhicule, c'est-à-dire sa charge est unitaire. Cette version du problème est appelé Problème de Minimisation du Nombre d'Arrêts Unitaire[7]. Nous remarquons néanmoins que plusieurs demandes peuvent avoir les mêmes stations de départ et/ou d'arrivée.

Soit $V = \{1, \ldots, n\}$ l'ensemble de stations numérotés selon leur ordre d'apparition sur le circuit. Soit E l'ensemble des m demandes unitaires, où chaque demande e est spécifiée par une station de départ $o_e \in V$ et par une station d'arrivée $d_e \in V$ tel que $o_e < d_e$, c'est-à-dire, $e = (o_e, d_e)$. Pour desservir ces m demandes, une flotte

7. Unit Stop Number Problem, en anglais.

de p véhicules identiques est fournie. Tous les véhicules possèdent la même capacité $C \in \mathbb{Z}^+$. Désignons par $K = \{1, \ldots, p\}$ l'ensemble de ces véhicules disponibles. Pour chaque instance $\mathcal{I} = (V, E, C)$ du problème, un graphe $G_{\mathcal{I}} = (V, E)$ peut être associé et par conséquent, les stations et les demandes peuvent être appelés noeuds et arêtes, respectivement. Puisque $o_e < d_e$ pour tout $e \in E$, le graphe associé $G_{\mathcal{I}}$ possède une orientation naturelle. Pour plus de simplicité, nous le considérons comme un graphe non-orienté.

Pour tout sous-graphe H de $G_{\mathcal{I}}$, et toute station $v \in V(H)$, nous désignons par

$$\delta_H^-(v) = \{e \in E(H) : d_e = v\} \quad \text{et} \quad \delta_H^+(v) = \{e \in E(H) : o_e = v\}$$

les ensembles des demandes dans $E(H)$ qui possèdent v comme leur station d'arrivée et de départ, respectivement. Pour tout sous-ensemble $F \subseteq E$ et toute station $v \in V$, nous désignons par

$$\Delta_F(v) = \{e \in F : o_e \leq v \leq d_e - 1\}$$

l'ensemble des demandes dans F qui *traversent* ou *commencent* à la station v. Les demandes appartenant à $\Delta_E(v)$ *intersectent* la station v.

Une solution réalisable est une partition de E en p sous-ensembles $E_1, \ldots, E_p$ telle que $|\Delta_{E_i}(v)| \leq C$ pour tout $i = 1, \ldots, p$ et pour tout $v \in V$. Le Problème de Minimisation du Nombre d'Arrêts Unitaire consiste à trouver une partition $\{E_1, \ldots, E_p\}$ qui minimise le coût de la fonction

$$c(\{E_1, \ldots, E_p\}) = \sum_{i=1}^{p} |V[E_i]|$$

où $V[E_i]$ représente l'ensemble des stations où le véhicule $i \in \{1, \ldots, p\}$ doit s'arrêter.

Un cas particulier intéressant nommé Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection, se pose quand il existe une station $v' \in V$ dont toutes les demandes intersectent, c'est-à-dire, $\Delta_E(v') = E$. Dans ce cas, chaque véhicule $i \in K$ peut servir au plus C demandes, c'est-à-dire, chaque sous-ensemble E_i doit posséder au plus C arrêtes.

Complexité

Dans cette section nous étudions de manière approfondie la complexité du Problème de Minimisation du Nombre d'Arrêts Unitaire. Tout d'abord, nous présentons une série de propriétés et de bornes inférieures pour ce problème. Sur la base de ces

résultats, nous montrons la solvabilité de quelques cas particuliers en temps polynomial. Ensuite, nous présentons une preuve de $\mathcal{NP}$-Complétude pour ce problème, ce qui permet de répondre à une conjecture posée par Pimenta et al. 2017 sur la complexité du Problème de Minimisation du Nombre d'Arrêts Unitaire.

Proposition 1 - Borne inférieure triviale
Soit (V, E, C) une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire, le nombre de stations n est une borne inférieure triviale sur le nombre total d'arrêts. ■

Proposition 2 - Borne inférieure basée sur les degrés
Soit (V, E, C) une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire,

$$\sum_{v \in V} \left\lceil \frac{\max\{|\delta^-_{G_\mathcal{I}}(v)|, |\delta^+_{G_\mathcal{I}}(v)|\}}{C} \right\rceil$$

est une borne inférieure sur le nombre total d'arrêts. ■

Proposition 3 - Nombre minimum de véhicules
Soit (V, E, C) une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire,

$$p_{min} = \max_{v \in V} \left\{ \left\lceil \frac{|\Delta_E(v)|}{C} \right\rceil \right\}$$

est le nombre minimum de véhicules nécessaires pour satisfaire à toutes les demandes dans E. ■

Dans le domaine de la théorie des graphes, la maille d'un graphe G est la longueur du plus court de ses cycles.

Proposition 4 - Grands cycles produisent des forêts
Soit (V, E, C) une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection telle que le graphe $G_\mathcal{I} = (V, E)$ a une maille de $k > C$, alors pour toute solution réalisable $\{E_1, \ldots, E_p\}$, $G_\mathcal{I}[E_i]$ est une forêt pour tout $i \in K$. ■

Proposition 5 - Borne inférieure basée sur la maille
Soit (V, E, C) une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection telle que le graphe $G_\mathcal{I} = (V, E)$ a une maille de $k > C$, alors

$$m + \left\lceil \frac{m}{C} \right\rceil$$

est une borne inférieure sur le nombre total d'arrêts. ■

Dans Pimenta et al. 2017, les auteurs énoncent sans preuve théorique que pour toute instance (V, E, C) du Problème de Minimisation du Nombre d'Arrêts Unitaire,

il existe au moins une solution optimale utilisant p_{min} véhicules. Nous montrons que cette affirmation est valide si $C = 2$ et (V, E, C) est une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection. Néanmoins, pour $C \geq 3$, nous montrons que l'affirmation n'est plus valide. Par conséquent, fixer le nombre de véhicules à p_{min} peut nuire à la recherche d'une solution optimale.

Une autre intuition raisonnable dans la recherche d'une solution optimale est de maintenir les demandes ayant les mêmes stations de départ et d'arrivée ensemble dans un même véhicule. En effet, pour toute instance du Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection avec $C = 2$, une telle démarche s'avère optimale. Autrement dit, il existe toujours au moins une solution optimale respectant cette règle. Une telle stratégie peut être problématique pour les cas plus généraux comme indique la proposition suivante.

Proposition 6 - Demandes parallèles
Soit $\mathcal{I} = (V, E, 2)$ une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire telle que $e \in E$ et $e' \in E$ sont deux demandes différentes ayant les mêmes stations de départ et d'arrivée, il se peut que dans aucune solution optimale les demandes e et e' soient affectées au même véhicule. ■

Les résultats présentés jusqu'ici offrent un aperçu de la difficulté de caractériser les solutions optimales du problème. Parallèlement, pour quelques cas spécifiques, nous avons montré que quelques intuitions simples peuvent s'avérer pertinentes dans la construction d'une solution optimale. Nous montrons ensuite comment mettre à profit ces résultats pour concevoir des algorithmes polynomiaux capables de résoudre certains cas particuliers du Problème de Minimisation du Nombre d'Arrêts Unitaire.

Proposition 7 - Capacité Unitaire
Soit $\mathcal{I} = (V, E, 1)$ une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire, alors le problème peut être résolu en $\mathcal{O}(m)$ et la solution optimale coûte

$$\sum_{v \in V} \max\left\{|\delta^-_{G_\mathcal{I}}(v)|, |\delta^+_{G_\mathcal{I}}(v)|\right\}.$$ ■

Proposition 8 - Graphes complets et $C = 2$
Soit $(V, E, 2)$ une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire telle que $G_\mathcal{I} = (V, E)$ est un graphe complet, alors le problème peut être résolu en temps polynomial. ■

Proposition 9 - Étoiles
Soit (V, E, C) une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire telle que $G_\mathcal{I} = (V, E)$ est un graphe étoile, alors le problème peut être résolu

en temps polynomial et la solution optimale coûte

$$\sum_{v \in V} \left\lceil \frac{\max\{|\delta_G^-(v)|, |\delta_G^+(v)|\}}{C} \right\rceil . \qquad \blacksquare$$

Proposition 10 - Chemins et cycles pour le cas d'intersection
Soit (V, E, C) une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection telle que $G_{\mathcal{I}} = (V, E)$ est un chemin (ou un cycle), alors le problème peut être résolu en temps polynomial et la solution optimale coûte

$$m + \left\lceil \frac{m}{C} \right\rceil . \qquad \blacksquare$$

Proposition 11 - Arbres, grilles et graphes bipartis complets pour le cas d'Intersection avec $C = 3$
Soit $(V, E, 3)$ une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection telle que $G_{\mathcal{I}} = (V, E)$ est un arbre (ou une grille ou un graphe biparti complet), alors le problème peut être résolu en temps polynomial. $\blacksquare$

Proposition 12 - Cas d'intersection avec $C = 2$
Soit $(V, E, 2)$ une instance du Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection, alors le problème peut être résolu en $\mathcal{O}(m)$ temps. $\blacksquare$

Une fois que le Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection avec $C = 2$ peut être résolu en temps polynomial, la question de savoir si le Problème de Minimisation du Nombre d'Arrêts Unitaire en général avec $C \geq 2$ est $\mathcal{NP}$-Difficile se pose immédiatement. Dans Pimenta et al. 2017, les auteurs conjecturent ce problème comme étant $\mathcal{NP}$-Difficile. Nous répondons positivement à cette conjecture à travers la réduction polynomiale d'un problème $\mathcal{NP}$-Complet au Problème de Minimisation du Nombre d'Arrêts Unitaire.

Theorem 1 - Problème de Minimisation du Nombre d'Arrêts Unitaire, avec $C \geq 2$
Le Problème de Minimisation du Nombre d'Arrêts Unitaire est $\mathcal{NP}$-Difficile pour n'importe quelle capacité fixée à $C \geq 2$ même si $G = (V, E)$ est un graphe planaire biparti. $\blacksquare$

Theorem 2 - Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection, avec $C = 3$
Le Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection est $\mathcal{NP}$-Difficile même si $G = (V, E)$ est un graphe planaire biparti et $C = 3$. $\blacksquare$

Theorem 3 - Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection, avec $C \geq 5$

Le Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection est $\mathcal{NP}$-Difficile pour n'importe quelle capacité fixée à $C \geq 5$ même si $G = (V, E)$ est un graphe planaire biparti. ■

La complexité du Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection avec $C = 4$ reste une question ouverte à laquelle nous proposons la conjecture suivante.

Conjecture 1 - Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection, $C = 4$
Le Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection est $\mathcal{NP}$-Difficile même si $G = (V, E)$ est un graphe planaire biparti et $C = 4$. ■

Il est important de souligner que les résultats obtenus sur la complexité des problèmes étudiés peuvent être étendus à d'autres problèmes associés comme le *Traffic Grooming* et le *k-Edge Partitioning Problem*. Une telle démarche permet d'améliorer les résultats connus dans littérature sur la complexité de ces problèmes (voir Amini, Pérennes et Sau 2009 ; Goldschmidt et al. 2003 ; Lemaire 2001).

Approche Polyédrale

Si on s'intéresse à des solutions optimales, une stratégie répandue face à un problème $\mathcal{NP}$-Difficile est de formuler et résoudre ce dernier via la programmation linéaire en nombres entiers. Afin d'améliorer les performances de ce genre d'approche, beaucoup de problèmes combinatoires ont bénéficié de l'étude faciale de l'enveloppe convexe de ses solutions réalisables. En effet, si une description complète de cette enveloppe convexe est connue, le problème peut être résolu à travers un programme linéaire. Malheureusement, la description complète de ces polyèdres nécessite habituellement un nombre exponentiel d'inégalités. Ici, l'équivalence entre optimisation et séparation décrite par Grötschel, Lovász et Schrijver 1981 joue un rôle principal. En effet, si le problème de séparation associé peut être résolu en temps polynomial, alors l'optimisation d'une fonction objective linéaire sur un polyèdre convexe peut également s'effectuer en temps polynomial. Le problème du couplage parfait (voir Edmonds 1965a ; Edmonds 1965b ; Pulleyblank 1973) est un exemple classique où l'enveloppe convexe des solutions réalisables a été complètement décrit et le problème de séparation associé a été démontré résoluble en temps polynomial. En contrepartie, une description partielle de l'enveloppe convexe des solutions réalisables est également souhaitable dans la conception d'algorithmes performants. Dans cette section, nous nous intéressons à l'étude polyédrale de l'en-

veloppe convexe des solutions réalisables du Problème de Minimisation du Nombre d'Arrêts Unitaire.

Le programme linéaire en nombres entiers décrit ci-dessous a été introduit dans Pimenta et al. 2017.

$$\min \sum_{v \in V} \sum_{i \in K} y_v^i \tag{1}$$

subject to

$$\sum_{i \in K} x_e^i = 1 \qquad \forall e \in E, \tag{2}$$

$$\sum_{e \in \Delta_E(v)} x_e^i \leq C \qquad \forall v \in V, i \in K, \tag{3}$$

$$x_e^i - y_v^i \leq 0 \qquad \forall i \in K, e \in E, v \in \{o_e, d_e\}, \tag{4}$$

$$x_e^i \in \{0,1\} \qquad \forall e \in E, i \in K, \tag{5}$$

$$y_v^i \in \{0,1\} \qquad \forall v \in V, i \in K. \tag{6}$$

La formulation (1)-(6) est fondée sur deux ensembles de variables binaires $x \in \{0,1\}^{m \times p}$ et $y \in \{0,1\}^{n \times p}$. La variable x_e^i représente le fait qu'une demande e soit ou non affectée au véhicule i, c'est-à-dire,

$$x_e^i = \begin{cases} 1, \text{ si } e \in E_i \\ 0, \text{ sinon.} \end{cases}$$

La variable y_v^i indique si le véhicule i s'arrête à la station v, c'est-à-dire,

$$y_v^i = \begin{cases} 1, \text{ si } v \in V[E_i] \\ 0, \text{ sinon.} \end{cases}$$

La fonction objectif (1) compte le nombre total d'arrêts réalisé par l'ensemble des véhicules. Les contraintes d'affectation (2) assurent que chaque demande est affectée à exactement un véhicule. Les contraintes de capacité (3) garantissent que la capacité de chaque véhicule est respectée tout au long du circuit. Les contraintes d'arrêt (4) imposent les véhicules de s'arrêter sur les stations de départ et d'arrivée associées aux clients qui lui ont été affectés. Finalement, les contraintes (5) et (6) établissent les domaines des variables x et y.

Une analyse préliminaire des résultats expérimentaux obtenus avec la formulation (1)-(6) en utilisant le solveur CPLEX 12.8 montre que la solution optimale est

fréquemment obtenue bien avant la fin de la procédure d'optimisation. C'est-à-dire que la borne inférieure utilisée dans l'algorithme de Coupes et Branchements évolue de façon beaucoup plus lente que la borne supérieure, ce qui indique la faiblesse d'une telle formulation. Accélérer la convergence des bornes inférieures vers le coût de la solution optimale constitue donc un défi majeur.

La puissance d'une formulation peut être mesurée par les bornes duales (bornes inférieures pour un problème de minimisation) fournies par sa relaxation linéaire. Dans Wolsey 1998, l'écart d'intégralité est utilisé pour mesurer la puissance d'une formulation. Selon ce concept nous étudions la puissance de la formulation (1)-(6) en analysant sa relaxation linéaire. Il s'avère que les résultats obtenus confirment la faiblesse de cette formulation.

Theorem 4 - Puissance de la formulation
La relaxation linéaire de la formulation (1)-(6) *fournit toujours une borne inférieure de n arrêts.* ■

Néanmoins, cette borne inférieure ne fournit aucune nouvelle information par rapport à la borne triviale dérivée de la Proposition 1. Cela ne poserait pas de problème si une telle borne était réellement efficace. Cependant, nous proposons des exemples d'instances où l'écart d'intégralité obtenu avec une telle borne peut atteindre la valeur de

$$1 - \frac{1}{\left\lceil \frac{m}{C} \right\rceil},$$

ce qui est en effet impressionnant.

Étant donné que la borne inférieure fournie par la relaxation linéaire n'est pas satisfaisante, nous nous intéressons à l'étude faciale de l'enveloppe convexe des solutions réalisables du Problème de Minimisation du Nombre d'Arrêts Unitaire pour essayer de renforcer la formulation (1)-(6). Pour cela, nous investiguons dans un premier temps la dimension de ce polyèdre, nommé $\mathcal{P}_{(V,E,C)}$. La dimension d'un polytope $\mathcal{P} \subseteq \mathbb{R}^d$, nommé $\dim(\mathcal{P})$, est définie comme étant le nombre maximum de vecteurs affinement indépendants dans $\mathcal{P}$ moins un. Nous montrons que

$$\dim\left(\mathcal{P}_{(V,E,C)}\right) = (n+m)p - m.$$

Ensuite nous proposons les conditions nécessaires et suffisantes pour que les inégalités composant la relaxation linéaire de la formulation (1)-(6) définissent des facettes du $\mathcal{P}_{(V,E,C)}$.

Theorem 5 - Relaxation des contraintes (6) **définissant des facettes**

L'inégalité

$$y_v^i \leq 1$$

définit une facette de $\mathcal{P}_{(V,E,C)}$ *pour tout* $v \in V, i \in K$. ■

Theorem 6 - Relaxation des contraintes (5) **définissant des facettes**
L'inégalité

$$x_e^i \geq 0$$

définit une facette de $\mathcal{P}_{(V,E,C)}$ *pour tout* $e \in E, i \in K$, *étant donné que* $p \geq 3$. ■

Theorem 7 - Inégalités (4) **définissant des facettes**
L'inégalité

$$x_e^i - y_v^i \leq 0$$

définit une facette de $\mathcal{P}_{(V,E,C)}$ *pour tout* $i \in K, e \in E, v \in \{o_e, d_e\}$. ■

Theorem 8 - Inégalités (3) **définissant des facettes**
L'inégalité

$$\sum_{e \in \Delta_E(v)} x_e^i \leq C$$

définit une facette de $\mathcal{P}_{(V,E,C)}$ *pour tout* $i \in K$ *si et seulement si*

1. *il n'existe aucune station* $v' \in V$ *tel que* $\Delta_E(v) \subset \Delta_E(v')$;
2. *pour toute station* $v' \in V$, $|\Delta_E(v) \setminus \delta(v')| \geq C$. ■

Enfin nous proposons des nouvelles inégalités valides capables de renforcer la formulation (1)-(6).

Theorem 9 - Validité des inégalités de capacité fortifiée
Les inégalités de capacité fortifiée

$$\sum_{e \in \delta_G^-(v)} x_e^i - C y_v^i \leq 0 \qquad \forall v \in V, i \in K, \tag{7a}$$

$$\sum_{e \in \delta_G^+(v)} x_e^i - C y_v^i \leq 0 \qquad \forall v \in V, i \in K, \tag{7b}$$

sont valides pour $\mathcal{P}_{(V,E,C)}$. ■

Theorem 10 - Inégalités de capacité fortifiée définissant des facettes
L'inégalité de capacité fortifiée (7a) *associée avec une station* $v \in V$ *et un véhicule* $i \in K$ *définit une facette de* $\mathcal{P}_{(V,E,C)}$ *si et seulement si*

$$|\delta_G^-(v)| \geq C + 1.$$

De la même façon, l'inégalité de capacité fortifiée (7b) *associée avec une station* $v \in V$ *et un véhicule* $i \in K$ *définit une facette de* $\mathcal{P}_{(V,E,C)}$ *si et seulement si*

$$|\delta_G^+(v)| \geq C + 1.$$ ■

Theorem 11 - Validité des inégalités d'arbres de cardinalité k
Les inégalités d'arbres de cardinalité k *suivantes sont valides pour* $\mathcal{P}_{(V,E,C)}$ *:*

$$\sum_{e \in S} x_e^i - \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) y_v^i \leq 0 \qquad \forall i \in K, j \in V, S \subseteq \Delta_E(j), \tag{8}$$

tel que $G[S]$ *est un arbre de cardinalité* k *avec* $k \geq C + 1$. ■

Theorem 12 - Inégalités d'arbres de cardinalité k **définissant des facettes**
Les inégalités d'arbres de cardinalité k (8) *définissent des facettes de* $\mathcal{P}_{(V,E,C)}$ *si et seulement si les conditions suivantes sont vérifiées :*

i. $G[S]$ *n'est pas un graphe étoile ;*

ii. si $G[S]$ *contient exactement 2 noeuds internes, alors* $E \setminus S$ *ne contient pas une arrête* (u, v) *tel que* u *et* v *sont les noeuds internes de* $G[S]$. ■

L'idée derrière la construction d'une inégalité d'arbre de cardinalité k est de trouver une structure en quelque sorte connexe qui viole la capacité du véhicule. Pour ces inégalités, la structure en question est complètement connexe ($G[S]$ est un arbre) et fait complètement partie de $\Delta_E(v)$. Basé sur ces remarques nous sommes capables de généraliser les inégalités d'arbres de cardinalité k comme suit.

Theorem 13 - Validité des inégalités d'arbres de cardinalité k **généralisées**
Les inégalités

$$\sum_{e \in S} x_e^i - \sum_{v \in V[S]} (\deg_{G[S]}(v) - 1) y_v^i \leq q - 1 \qquad \forall i \in K, j \in V, S \subseteq E$$

où $G[S]$ *est une forêt composée par* $1 \leq q \leq |S|$ *composantes connexes et* $|S \cap \Delta_E(j)| = C + 1$, *sont valides pour* $\mathcal{P}_{(V,E,C)}$. ■

Jusqu'à présent seulement des structures ne comportant pas de cycles ont été considérées dans la construction des inégalités valides. Les inégalités de mailles présentées ci-dessous prennent en compte ces structures manquantes.

Theorem 14 - Validité des inégalités de mailles
Les inégalités de maille

$$\sum_{e \in S} (C + 1) x_e^i - \sum_{v \in V[S]} C y_v^i \leq 0 \qquad \forall i \in K, j \in V, S \subseteq \Delta_E(j), \tag{9}$$

où $G[S]$ a une maille d'au moins $C+1$, sont valides pour $\mathcal{P}_{(V,E,C)}$. ■

Résultats expérimentaux

Dans cette section une large étude expérimentale sur le Problème de Minimisation du Nombre d'Arrêts Unitaire est menée et un algorithme de Coupes et Branchements est développé. La formulation (1)-(6) nous sert de point de départ. Suite aux performances insatisfaisantes obtenues avec cette formulation, nous proposons plusieurs atouts capables d'améliorer ces performances. Ces atouts incluent : i. casser la symétrie présente dans le Problème de Minimisation du Nombre d'Arrêts Unitaire, ii. éliminer des variables inutiles, iii. relaxer des variables, et iv. intégrer les inégalités valides identifiées renforçant la formulation dans l'algorithme de coupes et branchements. Toutes les implémentations utilisent le solveur `ILOG CPLEX 12.8` couplé avec `CPLEX Concert Technology` pour permettre une interface avec les librairies CPLEX en utilisant le langage C++.

Un programme linéaire en nombres entiers est défini comme symétrique si l'on peut permuter ses variables sans changer la structure du problème (voir Margot 2010). Dans le cas du Problème de Minimisation du Nombre d'Arrêts Unitaire il est possible de constater que pour toute solution réalisable donnée, une autre solution réalisable de même coût peut être obtenue en permutant les véhicules. Face à des problèmes linéaires en nombres entiers, de nombreux auteurs ont souligné l'importance d'éliminer – ou au moins réduire – la symétrie (*e.g.* Sherali et Smith 2001 ; Kaibel et Pfetsch 2008 ; Denton et al. 2010 ; Ostrowski, Anjos et Vannelli 2010). En effet, pour résoudre un problème linéaire en nombres entiers symétrique via une procédure par séparation et évaluation [8] (ou par coupes et branchements), nous sommes contraints de résoudre des sous-problèmes isomorphes présents dans l'arbre d'énumération, ce qui représente un effort computationnel inutile. Dans ce scénario, prouver l'optimalité d'une solution réalisable requiert d'explorer et d'éliminer chacun des noeuds isomorphes, ce qui peut être terriblement coûteux. Dans l'intention d'améliorer les performances de la formulation (1)-(6) nous étudions plusieurs méthodes permettant de casser la symétrie. Nous adaptons au Problème de Minimisation du Nombre d'Arrêts Unitaire l'algorithme linéaire appelé Fixation Orbitale [9] (voir Kaibel, Peinhardt et Pfetsch 2011) permettant de fixer des variables au fur et à mesure que l'arbre d'énumération est exploré (*i.e.*, à chaque branchement effectué), éliminant ainsi la symétrie.

Comme l'on pouvait s'y attendre, la méthode engagée pour briser la symétrie

8. Branch-and-Bound, en anglais.
9. Orbitopal Fixing, en anglais.

du problème a eu un impacte considérable sur les performances de la formulation (1)-(6). En effet, avec l'ajout d'une telle méthode, le nombre d'instances résolues à l'optimalité dans le délai de 2 heures parmi l'ensemble de 63 instances testées a augmenté de 1 à 19.

Étant donné que le nombre de véhicules utilisés dans une solution optimale peut différer du nombre minimum de véhicules nécessaires à l'obtention d'une solution réalisable, le nombre de véhicules disponibles p était jusqu'à présent fixé à m. Parallèlement, nous constatons que le nombre de variables présentes dans la formulation (1)-(6) est $(m+n)p$. Par conséquent, trouver une meilleure borne supérieure pour la valeur de p, permettrait de réduire le nombre de variables nécessaires dans une telle formulation. De plus, la symétrie présente dans la formulation est étroitement liée au nombre de véhicules disponibles. Ainsi, réduire le nombre de véhicules disponibles permet de réduire davantage la symétrie derrière le problème.

Le théorème suivant affirme qu'on peut réduire le nombre de véhicules disponibles, sans perte d'optimalité, pour n'importe quelle instance du Problème de Minimisation du Nombre d'Arrêts Unitaire.

Theorem 15 - Borne supérieure pour p
Pour n'importe quelle instance $\mathcal{I} = (V, E, C)$ *du Problème de Minimisation du Nombre d'Arrêts Unitaire,*

$$p_{opt} \leq \left\lceil \frac{m}{\left\lfloor \frac{C}{2} \right\rfloor + 1} \right\rceil,$$

où p_{opt} *désigne le plus petit nombre de véhicules nécessaires à l'obtention d'une solution optimale.*

De plus, nous montrons que la borne supérieure fournie par le Théorème 15 est serrée. Une telle démarche a permis une réduction de 60% des variables, en moyenne. Nous évaluons l'impacte de l'approche à travers une étude expérimentale comparant les performances obtenues en fixant p à m et en fixant p à la nouvelle borne.

La méthode par coupes et branchements consiste à intégrer un algorithme de plans sécants avant chaque phase de branchement dans un algorithme par séparation et évaluation. L'algorithme par coupes et branchements que nous avons développé utilise les inégalités valides identifiées précédemment dans la construction de l'algorithme de plans sécants. Les algorithmes de séparation associés à ces inégalités nécessitent souvent la manipulation de structures de données complexes et l'implémentation d'algorithmes combinatoires de pointe. Pour ce faire, nous avons utilisé LEMON, une libraire de code source ouvert écrite en langage C++ permettant l'implémentation de plusieurs algorithmes de graphes performants (voir Dezső, Jüttner et Kovács 2011 ; Egerváry Combinatorial Optimization

Research Group 2003).

Les inégalités valides de capacité fortifiée (7) se présentent en nombre polynomial. Par conséquent, stocker ces inégalités dans une réserve et vérifier leur satisfaction à chaque fois que l'algorithme de plans sécants est appelé, reste une façon efficace de résoudre leur problème de séparation. En revanche, les inégalités d'arbres de cardinalité k (8) et les inégalités de maille (9) apparaissent en nombre exponentiel et ainsi nécessitent une approche plus ingénieuse pour résoudre leur problème de séparation.

Theorem 16 - Séparation des inégalités d'arbres de cardinalité k
Le problème de séparation associé aux inégalités d'arbres de cardinalité k (8) *est $\mathcal{NP}$-Difficile.*

Theorem 17 - Séparation des inégalités de maille
Le problème de séparation associé aux inégalités de maille (9) *est $\mathcal{NP}$-Difficile.*

De ce fait, la conception d'un algorithme polynomial capable de résoudre les problèmes de séparation associés aux inégalités d'arbres de cardinalité k (8) et aux inégalités de maille (9) est invraisemblable, à moins que $\mathcal{P}$=$\mathcal{NP}$. Il faut donc espérer qu'un algorithme heuristique soit satisfaisant dans la recherche des inégalités violées. C'est pourquoi nous proposons des heuristiques basées sur une approche gloutonne pour résoudre ces problèmes de séparation.

Les résultats obtenus avec l'algorithme par coupes et branchements développé surpassent nettement les performances obtenues avec CPLEX. En effet, nous avons pu résoudre à l'optimalité 117 parmi les 315 instances testées dans la limite de 2 heures de temps de calcul, tandis que CPLEX n'a résolu que 7 de ces instances.

Conclusion

Dans cette thèse, nous avons exploré plusieurs aspects d'un problème combinatoire issu de la gestion d'un système de transport à la demande comprenant une flotte de véhicules électriques et autonomes.

Dans un premier temps, nous avons présenté le Problème de Minimisation du Nombre d'Arrêts Unitaire, une variante du Problème de Minimisation du Nombre d'Arrêts introduit par Pimenta et al. 2017. Une revue de la littérature sur les problèmes de transport à la demande a été fournie et d'autres problèmes issus du domaine des télécommunications se sont révélés proches du Problème de Minimisation du Nombre d'Arrêts Unitaire.

Ensuite, nous avons étudié quelques propriétés des solutions optimales du problème. Une telle étude a permis la découverte de nouvelles bornes sur le coût des solutions optimales ainsi qu'une meilleure compréhension des situations où l'optimalité peut être perdue si l'on adopte certaines politiques "intuitives". Des algorithmes combinatoires performant en temps polynomial et capables de résoudre des classes d'instances spécifiques ont été proposés. En revanche, le problème a été démontré $\mathcal{NP}$-Difficile pour toute valeur de $C \geq 2$, ce qui a permis de répondre à la conjecture posée par Pimenta et al. 2017. Il est important de noter que les preuves de complexité utilisées sont valides même pour le cas où le graphe associé est planaire biparti. Une telle propriété nous a permis d'étendre nos résultats à des problèmes liés améliorant ainsi leur classification de complexité actuelle.

Une fois que le problème a été démontré $\mathcal{NP}$-Difficile, l'idée de concevoir un algorithme exacte performant en temps polynomial a dû être abandonnée. Toutefois, le besoin de méthodes exactes demeurait. Une formulation linéaire en nombres entiers fournit par Pimenta et al. 2017 a été étudiée et s'est rapidement révélée particulièrement faible. Une étude faciale de l'enveloppe convexe des solutions réalisables du Problème de Minimisation du Nombre d'Arrêts Unitaire a donc été menée afin de mieux comprendre comment cette formulation pourrait être améliorée. La dimension d'un tel polyèdre est alors fournie ainsi que les conditions nécessaires et suffisantes dans lesquelles les inégalités composant la formulation définissent les facettes. Des familles d'inégalités valides capables de renforcer la formulation ont finalement été introduites. Les conditions nécessaires et suffisantes dans lesquelles certaines de ces familles définissent des facettes ont été présentées.

Enfin, les inégalités valides introduites ont été mises à l'épreuve dans le cadre d'une approche par coupes et branchements. Les problèmes de séparation associés à chaque famille d'inégalités valides ont été étudiés et lorsqu'ils ont été détectés comme $\mathcal{NP}$-Difficile, des procédures heuristiques ont été développées pour rechercher efficacement des coupes violées. Afin d'améliorer les performances de calcul, des méthodes de rupture de symétrie connues dans la littérature ont été étudiées et adaptées au Problème de Minimisation du Nombre d'Arrêts Unitaire. Une comparaison entre ces méthodes a été effectuée et l'algorithme de Fixation Orbitale dû à Kaibel, Peinhardt et Pfetsch 2011 a été choisi pour être celui implémenté dans notre algorithme par coupes et branchements. Une nouvelle limite supérieure pour le nombre de véhicules nécessaires à l'obtention d'une solution optimale est également proposée, ce qui entraînera un grand nombre de variables éliminées.

Les résultats expérimentaux ont prouvé que notre approche surpasse largement l'algorithme traditionnel par coupes et branchements de CPLEX. L'écart d'intégralité restant de toutes les instances testées a été réduit lors de l'application de notre

méthode. De plus, un nombre considérable d'instances qui n'ont pas été résolues de façon optimale dans le délai de deux heures par CPLEX, ont finalement pu être résolues à l'aide de notre algorithme.

Il reste encore certainement beaucoup d'aspects inexplorés avec un potentiel scientifique important à étudier autour du Problème de Minimisation du Nombre d'Arrêts Unitaire. Les axes de recherche futurs pourraient impliquer une étude plus approfondie des inégalités d'arbres de cardinalité k généralisées, dont les conditions nécessaires et suffisantes dans lesquelles des facettes sont définies font toujours défaut. De plus, des procédures de séparation efficaces pour la prise en compte de ces inégalités pourraient améliorer davantage la performance de notre algorithme par coupes et branchements. Nous pensons également qu'une formulation étendue avec une relaxation linéaire plus serrée pourrait être la réponse pour éviter d'avoir à procéder à autant de coupes supplémentaires afin de renforcer la formulation. Pour l'aspect de la complexité, la réponse à notre conjecture selon laquelle le Problème de Minimisation du Nombre d'Arrêts Unitaire d'Intersection est $\mathcal{NP}$-Difficile pour la capacité $C = 4$ (même lorsque G est restreint à la classe des graphes bipartis planaires) reste une question ouverte qui mérite de l'attention.

Bibliographie

Amini, Omid, Stéphane Pérennes et Ignasi Sau (2009). "Hardness and approximation of traffic grooming". In : *Theoretical Computer Science* 410.38-40, p. 3751–3760.

Applegate, David et al. (2001). "TSP cuts which do not conform to the template paradigm". In : *Computational combinatorial optimization*. Springer, p. 261–303.

Applegate, David et al. (2006). *Concorde TSP Solver*. URL : `http://www.tsp.gatech.edu/concorde/` (visité le 04/05/2019).

Bsaybes, Sahar (2017). "Modèles et algorithmes de gestion de flottes de véhicules vipa". Thèse de doct. Ph. D. thesis, Université Clermont Auvergne.

Denton, Brian T et al. (2010). "Optimal allocation of surgery blocks to operating rooms under uncertainty". In : *Operations research* 58.4-part-1, p. 802–816.

Dezső, Balázs, Alpár Jüttner et Péter Kovács (2011). "LEMON–an open source C++ graph template library". In : *Electronic Notes in Theoretical Computer Science* 264.5, p. 23–45.

Edmonds, Jack (1965a). "Maximum matching and a polyhedron with 0, 1-vertices". In : *Journal of research of the National Bureau of Standards B* 69.125-130, p. 55–56.

Edmonds, Jack (1965b). "Paths, trees, and flowers". In : *Canadian Journal of mathematics* 17.3, p. 449–467.

Egerváry Combinatorial Optimization Research Group, EGRES (2003). *Lemon Graph Library*. URL : `https://lemon.cs.elte.hu/trac/lemon` (visité le 04/05/2019).

Gao, Paul et al. (2016). "Automotive revolution–perspective towards 2030 : How the convergence of disruptive technology-driven trends could transform the auto industry". In : *Advanced Industries, McKinsey & Company*.

Goldschmidt, Olivier et al. (2003). "The SONET edge-partition problem". In : *Networks : An International Journal* 41.1, p. 13–23.

Grötschel, Martin, László Lovász et Alexander Schrijver (1981). "The ellipsoid method and its consequences in combinatorial optimization". In : *Combinatorica* 1.2, p. 169–197.

Kaibel, Volker, Matthias Peinhardt et Marc E Pfetsch (2011). "Orbitopal fixing". In : *Discrete Optimization* 8.4, p. 595–610.

Kaibel, Volker et Marc E Pfetsch (2008). "Packing and partitioning orbitopes". In : *Mathematical Programming* 114.1, p. 1–36.

Lemaire, P (2001). "Optimisation de réseaux SONET/SDH : éléments théoriques et résolution pratique". In : *Mémoire de DEA*.

Margot, François (2010). "Symmetry in integer linear programming". In : *50 Years of Integer Programming 1958-2008*. Springer, p. 647–686.

Ostrowski, James, Miguel F Anjos et Anthony Vannelli (2010). *Symmetry in scheduling problems*. Citeseer.

Pimenta, Victor et al. (2017). "Models and algorithms for reliability-oriented Dial-a-Ride with autonomous electric vehicles". In : *European Journal of Operational Research* 257.2, p. 601–613.

Pulleyblank, WR (1973). "Faces of Matching Polyhedra, Univ. of Waterloo, Dept. Combinatorics and Optimization". Thèse de doct. Ph. D. Thesis.

Sherali, Hanif D et J Cole Smith (2001). "Improving discrete model representations via symmetry considerations". In : *Management Science* 47.10, p. 1396–1407.

Wolsey, Laurence A (1998). *Integer Programming. Series in Discrete Mathematics and Optimization*. Wiley-Interscience New Jersey.