The Minimum Flow Cost Hamiltonian Tour Problem

Camilo Ortiz Astorquiza

A Thesis
in
the Department
of
Mathematics and Statistics

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Science (Mathematics) at
Concordia University
Montreal, Quebec, Canada

July 2013

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Camilo Ortiz Astorquiza

Entitled: The Minimum Flow Cost Hamiltonian Tour Problem

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Mathematics)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Ronald Stern Chair

Dr. Ronald Stern Examiner

Dr. Gilbert Laporte Examiner

Dr. Ivan Contreras Supervisor

Dr. Pawel Gora Supervisor

Approved by

Chair of Department or Graduate Program Director

2013

Dean of Faculty of Arts and Science

ABSTRACT

The Minimum Flow Cost Hamiltonian Tour Problem

Camilo Ortiz

In this thesis we introduce the *minimum flow cost Hamiltonian tour problem* (FCHT). Given a graph and positive flow between pairs of vertices, the FCHT consists of finding a Hamiltonian cycle that minimizes the total cost for sending flows between pairs of vertices thorough the shortest path on the cycle. We prove that the FCHT belongs to the class of *NP*-hard problems and study the polyhedral structure of its set of feasible solutions. In particular, we present five different MIP formulations which are theoretically and computationally compared. We also develop some approximate and exact solution procedures to solve the FCHT. We present a combinatorial bound and two heuristic procedures: a greedy deterministic method and a greedy randomized adaptive search procedure. Finally, a branch-and-cut algorithm is also proposed to solve the problem exactly.

Acknowledgements

I would like to thank my supervisors Dr. Ivan Contreras and Dr. Pawel Gora. Your support, patience and advice have been tremendously helpful for my professional and personal growth.

I thank my family for their constant encouragement and effort to help me achieve this important goal in my life. This work would had never been possible without any of you. Special gratitude to my wife, for sharing this journey with me. Your love and company are invaluable to me.

Estaré eternamente agradecido con mis padres, Julio Vicente y Maria Cristina, mi hermana Catalina y todos los familiares y amigos que siempre han estado pendientes y dispuestos a darme una mano cuando lo necesito. A mis padres especialmente, que han sido y serán mi ejemplo de trabajo, dedicación y principios.

Contents

Chapter 1

Introduction

Combinatorial optimization has emerged as a major research area in discrete and applied mathematics. It deals with the solution of optimization problems over discrete structures by integrating techniques from combinatorics, linear programming and the theory of algorithms. It has played an important role in the development of mathematical results and theories for the past few decades and it has shown a great impact on a wide variety of applications in science and engineering such as production planning, transportation, telecommunications, computer science, statistics, and biology, among others (see, Cook et al., 1998; Schrijver, 2003; Bazaraa et al., 2009).

A natural and systematic way to study combinatorial optimization problems is to express them as integer programming problems. Integer programming refers to the class of constrained optimization problems in which some or all of the variables are restricted to take integer values. In the most studied and used integer programs the objective function is linear and the constraints are linear inequalities. Often, the main issue for the solution of integer programs lies on the efficiency of the mathematical model and the methodology used to solve it, since we usually deal with a finite but very large set of feasible solutions. Therefore, evaluating all solutions one by one

and selecting the best one is not an option and more efficient methods should be found. Many combinatorial optimization problems are known to be in the class *NP*-hard, which means that, unless $P = NP$, no solution method, with a running time bounded by a polynomial in the size of the representation exists to optimally solve them. As a consequence, for most real size applications, combinatorial optimization problems cannot be solved by general purpose methods or solvers and thus, ad hoc mathematical models and specialized solution algorithms are needed to efficiently solve them.

One of the most intensively studied combinatorial optimization problems over the past fifty years is the so-called *traveling salesman problem* (TSP) (for a recent survey see Öncan et al., 2009). Given a weighted graph G, the TSP consists of finding a cycle visiting all the vertices of G exactly once, i.e. a Hamiltonian tour, for which the sum of the cost of the edges is minimized. The TSP focuses on the design costs of the cycle and despite its apparent simplicity, it is known to be *NP*-hard. Its wide range of applications as well as the complexity to optimally solve it have made of the TSP a symbol for this field. Many exact and approximate solution techniques have risen trying to provide (near-) optimal solutions to this problem. Nowadays, state-of-the-art algorithms can solve TSP instances with up to thousands of nodes on a desktop computer (Cook, 2011).

In this thesis we introduce the *minimum flow cost Hamiltonian tour problem* (FCHT), which can be stated as follows. Given a weighted graph G and a positive flow between pairs of vertices of G, the FCHT consists of finding a Hamiltonian cycle of G that minimizes the total cost for sending flows between pairs of vertices through their shortest path on the cycle. The FCHT is a combinatorial optimization problem closely related to the TSP. Note that the set of feasible solutions (all Hamiltonian cycles in G) of both the TSP and the FCHT is the same. However, the

FCHT focuses on the operational costs of the network whereas the TSP focuses on the design costs. Therefore, the set of optimal solutions of both problems will be different because of their different objective functions.

The goal of this thesis is twofold. The first one is to introduce and study a new challenging combinatorial optimization problem, refereed to as the FCHT. We show that it belongs to the class of *NP*-hard problems and study the polyhedral structure of its set of feasible solutions. In particular, five different integer programming formulations are presented and theoretically compared with respect to the quality of their linear programming (LP) relaxation bounds. The first two formulations are based on constructing the paths between pairs of origin/destination vertices, one of them requiring a number of decision variables given by a polynomial function and the other requiring an exponential one. The third formulation is based on determining the amount of flow passing through every arc of the network, coming from a particular origin vertex. This formulation requires fewer variables and constraints to model the problem than the previous two formulations, but at the expense of producing weaker LP bounds. The last two formulations contain the least number of decision variables among all proposed formulations, however, the bounds are shown to be rather weak.

The second contribution is to develop some approximate and exact solution methods to solve the FCHT. We first present a combinatorial bound based on shortest paths and two heuristic methods that can be used to obtain lower and upper bounds, respectively, on the optimal solution value of the problem. The first heuristic uses a greedy deterministic approach to construct an initial feasible solution whereas the second one uses a greedy randomized adaptive search procedure (see, Gendreau and Potvin, 2010). Both heuristics use a simple local search method to improve the initial solutions. We then present an exact branch-and-cut method to obtain optimal solutions for the FCHT. This algorithm employs the flow-based formulation and some

families of valid inequalities as a bounding procedure at the root node of the enumeration tree. Finally, we present a series of computational experiments to evaluate the practical performance of some of the proposed mathematical models and the solution algorithms.

The remainder of this thesis is organized as follows. In chapter 2 we introduce some relevant concepts of convex analysis, combinatorial optimization, integer programming and complexity theory, needed for making this document self-contained. In chapter 3 we formally define the FCHT and in chapter 4 we introduce five different integer programming formulations and a comparison of their LP bounds. Chapter 5 presents the proposed approximate and exact solution methods. In chapter 6 we show the main computational results. Finally, in chapter 7 we summarize our conclusions and point out future research directions on the topic.

Chapter 2

Preliminaries

In this chapter we present some relevant definitions and mathematical results from convex analysis, linear programming, integer programming, combinatorial optimization, and complexity theory. The results presented in this section are derived from the books of Bazaraa et al. (2009); Nemhauser and Wolsey (1988); Wolsey (1998), and Cook et al. (1998).

2.1 Convex Analysis

In this section we review some of the basic results from convex analysis, including the notions of convex sets, hyperplanes, and polyhedral sets used to describe linear and integer programs.

Definition 2.1. *A set $X \subseteq \mathbb{R}^n$ is called a convex set if given any two points $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ in X then $\lambda \boldsymbol{x_1} + (1-\lambda)\boldsymbol{x_2} \in X$ for any $\lambda \in [0,1]$. Any point of the form $\lambda \boldsymbol{x_1} + (1-\lambda)\boldsymbol{x_2}$ is called a convex combination of $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$.*

Definition 2.2. *A point $\boldsymbol{x}$ in a convex set X is called an extreme point of X if $\boldsymbol{x}$ cannot be represented as a convex combination of two distinct points in X.*

Consider now a nonzero vector $\boldsymbol{p}$ in $\mathbb{R}^n$ and a scalar k. Then, we have the following definitions.

Definition 2.3. *A hyperplane H in $\mathbb{R}^n$ is a set of the form $\{\boldsymbol{x} : \boldsymbol{px} = k\}$.*

Definition 2.4. *A half-space is a collection of points of the form $\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{px} \geq k\}$.*

Definition 2.5. *A polyhedral set (or polyhedron) is the intersection of a finite number of half-spaces.*

Also a polyhedron can be defined as a subset $P \subseteq \mathbb{R}^n$ described by a finite number of linear constraints $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The polyhedral set corresponding to the constraints in an optimization problem is called the *feasible region* and its elements are called *feasible solutions* i.e. the points satisfying all constraints. Finally, we present the formal definition of convex hull.

Definition 2.6. *Given a set $X \subseteq \mathbb{R}^n$, the convex hull of X, denoted $conv(X)$ is defined as:*

$$conv(X) = \{x : x = \sum_{i=1}^{t} \lambda_i x^i, \sum_{i=1}^{t} \lambda_i = 1, \lambda_i \geq 0 \text{ for } i = 1, \cdots, t \text{ with } \{x^1, \cdots, x^t\} \subseteq X\}.$$

In other words, the $conv(X)$ is the (unique) minimal convex set containing X.

2.2 Linear, Integer and Combinatorial Optimization

In this section we focus on the definitions of linear and integer programs, combinatorial optimization problems, formulations, and relaxations. We also provide some definitions that will help us compare integer programming formulations.

Definition 2.7. *Consider a finite set* $N = \{1, \cdots, n\}$ *with corresponding weights* c_j *for each* $j \in N$ *and a set* F *of feasible subsets of* N*. The problem of finding a minimum weight feasible subset is the combinatorial optimization problem*

$$\min_{S \subseteq N} \{\sum_{j \in S} c_j : S \in F\}.$$

Given the discrete nature of combinatorial optimization problems, they are generally formulated as integer programs where some or all the variables must be integer-valued. Consider the linear mixed integer programming problem defined as follows.

Definition 2.8. *If some but not all variables are integer, we have the (linear) mixed integer program (MIP)*

$$\min\{cx + dy : Ax + By \leq b, x \in \mathbb{R}^n_+ \text{ and } y \in \mathbb{Z}^p_+\},$$

where $x = (x_1, \cdots, x_n)$ is a vector of real variables and $y = (y_1, \cdots, y_p)$ is a vector of integer variables. An *instance* of MIP is specified by the data (c, d, A, B, b). The feasible region of MIP is given by the set $S = \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+ : Ax + By \leq b\}$. The function $z = cx + dy$ is called the objective function and an *optimal solution* $z^* = (x^*, y^*)$ is a feasible point for which the objective value is minimum, i.e.,

$$cx^* + dy^* \leq cx + dy, \quad \forall (x, y) \in S.$$

An special case of MIP is the following linear program.

Definition 2.9. *If all variables are in* $\mathbb{R}_+$*, we have the linear program (LP)*

$$\min\{cx : Ax \leq b \text{ and } x \in \mathbb{R}^p_+\}.$$

Also, a linear programming problem can be described as the optimization problem of a linear objective function while satisfying a set of linear equality or inequality constraints. Classical techniques such as the simplex method as well as more sophisticated polynomial time algorithms, such as interior point methods, are nowadays capable of solving large-scale LP problems with millions of variables and constraints.

When we restrict all the variables to be integer-valued, we have another special case of a MIP.

Definition 2.10. *If all variables are in $\mathbb{Z}_+$, we have the (linear) integer program (IP)*

$$\min\{cx : Ax \leq b \text{ and } x \in \mathbb{Z}_+^n\}.$$

In general, the solution of integer programming models is a challenging task. We cannot use classical machinery of convex optimization because $S = \{x \in \mathbb{Z}_+^n : Ax \leq b\}$ is not a convex set and a linear function over $\mathbb{Z}_+^n$ is convex but non-differentiable. Therefore, we must resort in other mathematical techniques to prove that a particular solution is optimal by arguments other than convexity and differentiability.

When using integer programming, the first step is usually to represent the set of feasible solutions of an optimization problem with a polyhedron.

Definition 2.11. *A polyhedron $P \subseteq \mathbb{R}^{n+p}$ is a formulation for a set $X \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ if and only if $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$.*

From the previous definition, it is clear that there is not a unique formulation for an optimization problem. This leads us to the following definition in order to be able to compare between the different formulations of a particular problem.

Definition 2.12. *Given a set $X \subseteq \mathbb{R}^n$ and two formulations P_1 and P_2 for X then we say that P_1 is a better formulation than P_2, if $P_1 \subset P_2$.*

According to this definition, we can conclude that no other formulation for a given problem X is better than $conv(X)$. Given that $conv(X)$ is a polyhedral set, it can be represented by a finite set of linear constraints and thus, solved as a linear program. For some particular classes of combinatorial optimization problems the characterization of $conv(X)$ is known. However, for the vast majority of the problems that belong to the class *NP*-hard, the representation of $conv(X)$ remains unknown.

Several algorithms have been developed to solve MIPs. The key idea to this methods is usually to construct a sequence of lower bounds $\underline{z} \leq z^*$ and upper bounds $\overline{z} \geq z^*$ such that $z = \underline{z} = \overline{z}$. In practice, algorithms find an (not necessarily) increasing sequence of lower bounds and a (not necessarily) decreasing sequence of upper bounds, and stop when the difference between the lower bound and the upper bound is within a threshold value. We then need to find ways for obtaining such bounds. Following this idea, we must have in mind that in real applications we usually look for a balance between the time consumed by a model and its exactness. This means, if we can get a good approximation efficiently, sometimes is better than taking a long time to find the exact optimal solution. Obviously, this depends on the requirements of the problem and the particular application.

Many integer programming techniques use the simple idea of replacing a difficult MIP by an easier optimization problem whose optimal solution value is a lower bound for the MIP optimal solution value.

Definition 2.13. *A problem (R)* $z_R = \min\{f(x) : x \in T \subseteq \mathbb{R}^n\}$ *is a relaxation of the integer program (IP)* $z = \min\{g(x) : x \in X \subseteq \mathbb{R}^n\}$ *if the following two conditions are satisfied:*

- $X \subseteq T$
- $f(x) \leq g(x)$ *for all* $x \in X$.

An immediate result of this definition is that if (R) is a relaxation of (IP) then $z_R \leq z$. This means that any relaxation of the original problem will give us a lower bound if we are minimizing (an upper bound if we are maximizing). In the case of upper bounds when minimizing, every feasible solution is an upper bound and the problem lies in finding the smallest one. We present some methods to find upper bounds in later sections.

One of the most common relaxations of integer programs consists in dropping the integrality conditions.

Definition 2.14. *For an (IP)* $z = \min\{cx : x \in S\}$ *with* $S = X \cap \mathbb{Z}^n_+$ *the linear programming relaxation is given by* $z^{LP} = \min\{cx : x \in X\}$.

We present in the next proposition a known result that links the comparison between formulations and the LP relaxations.

Proposition 2.1. *Consider* P_1 *and* P_2 *two different formulations for an integer program and assume that* P_1 *is better than* P_2. *If* $z_i^{LP} = \min\{cx : x \in P_i\}$ *are the values of the associated linear programming relaxations, then* $z_1^{LP} \geq z_2^{LP}$.

Another relaxation widely used for MIP is called the Lagrangean relaxation. The main idea lies in the fact that sometimes it is "easy" to handle a set of constraints but we face a set of "difficult" constraints as well. In this case, we drop the difficult constraints and we add them to the objective function using Lagrangean multipliers. So, if we consider the (IP) $z = \min\{cx : Dx \leq d \text{ and } x \in X\}$ then for any value of $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m)$ we define the problem $IP(\lambda)$:

$$z(\lambda) = \min\{cx + \lambda(Dx - d) : x \in X\}.$$

Proposition 2.2. *The problem IP(*λ*) is a relaxation of the original IP for all* $\lambda \geq 0$.

Definition 2.15. *The problem of finding the best (largest) Lagrangean lower bound on the optimal solution value of IP is*

$$(LR) \qquad \max_{\lambda \geq 0} IP(\lambda).$$

It is called the Lagrangean dual of (IP).

Proposition 2.3. *The Lagrangean dual (LR) is equivalent to the primal relaxation (PR)*

$$\begin{aligned} &\textit{minimize} && cx \\ &\textit{subject to} && Dx - d \\ & && x \in conv(\{x \in \mathbb{Z}^n_+ : Ax \leq b\}) \end{aligned}$$

in the sense that $z^*_{LR} = z^*_{PR}$.

Definition 2.16. *We say that (LR) has the Integrality Property if*

$$conv(\{x \in \mathbb{Z}^n_+ : Ax \leq b\}) = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$$

These definitions and results imply that if a Lagrangean dual (LR) has the integrality property then $z^*_{LP} = z^*_{PR} = z^*_{LR} \leq z^*_{IP}$.

Given that the LP relaxations are generally easier that the original MIP, and the fact that $z^{LP} \leq z$, we are interested in knowing under which conditions a solution for the LP relaxation will coincide with the solution of the IP. We next present some definitions that will help us characterize some sufficient conditions.

Definition 2.17. *A matrix A is totally unimodular (TU) if every square sub-matrix of A has a determinant +1,-1 or 0.*

We present sufficient conditions for a matrix to be TU in the following proposition.

Proposition 2.4. *A matrix A is TU if*

- $a_{ij} \in \{1, 0, -1\}$ *for all* i, j *and*
- *for any subset M of the rows, there exists a partition (M_1, M_2) of M such that each column j satisfies* $|\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij}| \leq 1$.

Finally, we present the following result where sufficient and necessary conditions are given in order to solve an integer program using its LP relaxation.

Proposition 2.5. *The linear program* $\min\{cx : Ax \leq b, x \in X\}$ *has an integral optimal solution for all integer vectors b for which it has a finite optimal value if and only if A is totally unimodular.*

2.3 Solution Methodologies

In this section we review some of the most common and broadly used methods to solve IPs and MIPs. In particular, we present a brief description of approximate methods such as heuristic and meta-heuristic as well as exact algorithms such as branch-and-bound and cutting plane methods.

A heuristic method can be described as a procedure that is based on a set of rules that seem to be of some utility for finding feasible solutions for a given problem. The most simple heuristics are greedy and local search procedures. The aim of a greedy heuristic is to construct an initial feasible solution, while in a local search procedure is to improve some initial solution. A greedy method works on the basis of a rule which serves to identify elements, one at a time, to build a feasible solution for a problem. It has the inconvenient of being myopic, in the sense that earlier selections of promising elements could yield to the selection of very bad elements at

some point of the constructive procedure. A local search procedure is an iterative method that terminates when there are no solutions immediately accessible that could improve the last one found. However, it has the disadvantage of getting trapped in local optima. A lot of research has been done to develop heuristic methods that overcome local optimality. This search for improved heuristics combined with the development of computer technology have given rise to the so-called meta-heuristic methods. They refer to a master strategy that seeks to overcome local optimality and provide a general framework for the development of solution methods (see Gendreau and Potvin, 2010). One of the most used meta-heuristics is the greedy randomized adaptive search procedure (Festa and Resende, 2011).

One can find in theory the optimal solution for a given combinatorial optimization problem by enumerating all feasible solutions and evaluating the objective function. However, this is not efficient for most cases because usually, there is a huge number of feasible solutions. Therefore, exact methodologies have been developed to solve IPs without having to resort on a complete enumeration. Among them, we find the branch-and-bound method and the cutting plane algorithm. Broadly speaking, the branch-and-bound algorithm consists of dividing the set of feasible solutions, so that we have to handle smaller ("easier") problems, while the cutting plane algorithm looks for additional constraints that might reduce the set of feasible solutions by "cutting" parts of the set that are not part of the convex hull of integer solutions.

Proposition 2.6. *Let* $z = \min\{cx : x \in S\}$ *and* $S = S_1 \cup \cdots \cup S_k$ *be a decomposition of* S *into smaller sets, and let* $z^k = \min\{cx : x \in S_k\}$ *for* $k = 1, \cdots, K$. *Then,* $z = \min_k z^k$.

The branch-and-bound is interpreted as a tree, where each node corresponds to a given set S_i and the root node is the entire set S. Usually this method involves the use of LP relaxations to find lower bounds on each set S_i and compare them with the

best upper bound found to that point, in order to discard sets of feasible points (i.e. when $\underline{z}^k > \overline{z}$) and thus, reduce the size of the problem.

We now present a few definitions necessary for the the cutting plane method. We begin with the concepts of valid inequalities and separation problems.

Definition 2.18. *An inequality $\alpha x \leq \alpha_0$ is a valid inequality for $X \subseteq \mathbb{R}^n$ if $\alpha x \leq \alpha_0$ for all $x \in X$.*

Definition 2.19. *The separation problem associated to a combinatorial optimization problem is the following: given $x^* \in \mathbb{R}^n$, is $x^* \in conv(X)$? If not, find an inequality $\alpha x \leq \alpha_0$ satisfied by all points in X but violated by the point x^*.*

In order to explain the cutting plane algorithm, we must have in mind that for a given (IP) $\min\{cx : x \in X\}$ where $X = \{x \in \mathbb{Z}^n_+ : Ax \leq b\}$ we can in theory, find the convex hull of X, i.e., $conv(X) = \{x \in \mathbb{Z}^n_+ : \hat{A}x \leq \hat{b}\}$, which would lead us to simply solve its LP relaxation. However, finding $conv(X)$ is not generally easy (or efficient) task. What we can do in practice is to reduce the size of the set of feasible solutions by means of valid inequalities. Valid inequalities that are useful are the ones that are valid for X but are violated for the linear programming relaxation of (IP). Suppose that $X = P \cap \mathbb{Z}^n$ and let V be a set of valid inequalities of X.

Cutting Plane Algorithm.

Start: Set $i = 0$, $T = 0$ and $P^0 = P$.

while$(T \neq 1)$ **do**
Solve the linear program $z^i = \min\{cx : x \in P^i\}$.
if(The optimal solution x^i is integer)
T=1.
else
Solve the separation problem for x^i and V.
if (There is an inequality that cuts off x^i)
$P^{i+1} = P^i \cap \{x : \alpha^i x \leq \alpha_0^i\}$ and $i = i + 1$.
else
$T = 1$.
end if
end if
end do

If the output of the separation problem is a valid inequality that cuts off the optimal point for the current linear program, we will improve the formulation. The issue is that sometimes the separation problem is also *NP*-hard and therefore, is not easy to solve. This is usually handled by using approximation techniques, so that we find violated inequalities but not necessarily the most violated one.

A very interesting process to find the convex hull of a given set for any IP is the so-called Chvátal-Gomory procedure (Chvátal, 1973; Gomory, 1958), which constructs, in a finite number of iterations, the $conv(X)$ through sets of valid inequalities that at every step improve the formulation. The problem with this procedure is that in practice it can take a very large number of iterations to converge to an optimal solution.

2.4 Complexity Theory

One of the most studied open questions for the last decades in mathematics and computer science, particularly in complexity theory, is the P vs NP problem. Complexity theory is related to the classification of optimization problems in terms of their difficulty to solve them. For each optimization problem, an associated (YES-NO) decision problem, of the form: "Is there an $x \in S$ with value $cx \leq k$ for a given k?" is used to define the class of legitimate problems.

Definition 2.20. *NP is the class of decision problems with the property that: for any instance for which the answer to the problem is yes, there is a ("quick") polynomial proof of the yes.*

Definition 2.21. *P is the class of decision problems in NP for which there exits a polynomial algorithm to answer the problem.*

Definition 2.22. *If $Q, R \in NP$ and if an instance of Q can be converted in polynomial time to an instance of R, then Q is polynomially reducible to R.*

Definition 2.23. *The class of NP-Complete problems, is the subset of problems $Q \in NP$ such that for all $R \in NP$, R is polynomially reducible to Q.*

Definition 2.24. *An optimization problem for which its corresponding decision problem is NP-Complete is called NP-hard.*

The problem of P vs. NP consists in determining whether $P = NP$ or $P \neq NP$. By definition we already know that $P \subseteq NP$. However, to this date, there are problems in NP for which an algorithm running in polynomial time that can answer them has not been found yet. This does not imply that they do not exist nor that they do, so the question remains open.

In Cook (1971) the author showed that the $satisfiability$ problem is NP-Complete, therefore the class is not empty. In order to prove that a given problem Q belongs to the class NP-Complete we have to prove that $Q \in NP$ and that an NP-Complete problem can be polynomially reduced to Q. We are particularly interested in describing to which class the FCHT belongs to.

Chapter 3

The Minimum Flow Cost Hamiltonian Tour Problem

In this chapter we first introduce the formal definition of the FCHT and show that the problem is *NP*-hard. We then describe several optimization problems that have been studied in the literature that relate to the FCHT. The main goal is not to provide a comprehensive literature review on these problems but to point out their similarities, differences and connections with the FCHT. Finally, some potential application for the FCHT are provided.

3.1 Problem Definition

Let $G = (V, E)$ be a complete and undirected graph where V is the set of vertices with $|V| = n > 3$ and E is the set of edges. Let c denote a given function on E such that c_{ij} is the cost (usually also called distance) between i and j. Also, for each pair of vertices $i, j \in V$, let w_{ij} be the amount of flow that need to be routed from origin i to destination j. When no flow exists for a given pair $i, j \in V$, $i \neq j$, we have $w_{ij} = 0$. We assume that $w_{ii} = 0$, for all $i \in V$. Now, for every Hamiltonian tour

H of G and for each pair of vertices $i, j \in V$, let $d_H(i,j)$ denote the length of the shortest path in H from i to j. Given that H is a cycle, we always have exactly two possible directed paths from i to j, and $d_H(i,j)$ is defined as the shortest of them. The flow cost from i to j in H is thus given by $w_{ij}d_H(i,j)$, and the total flow cost of H is the sum of the flow costs for all pairs of vertices. The FCHT consist of finding a Hamiltonian cycle that minimizes the total flow cost and thus, can be stated as the following combinatorial optimization problem,

$$\text{(FCHT)} \qquad \min_{H \subseteq E} \left\{ \sum_{i,j \in V} w_{ij} d_H(i,j) : H \text{ is a Hamiltonian tour} \right\}.$$

The following instance of the FCTH will be used throughout the subsequent sections to illustrate some results and differences with other problems found in the literature.

Example 3.1. *Consider a complete graph with $n = 4$ and the following cost matrix C and the corresponding flow matrix W.*

$$C = \begin{pmatrix} 0 & 10 & 20 & 10 \\ 10 & 0 & 10 & 5 \\ 20 & 10 & 0 & 10 \\ 10 & 5 & 10 & 0 \end{pmatrix} \quad W = \begin{pmatrix} 0 & 10 & 10 & 100 \\ 0 & 0 & 100 & 200 \\ 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

In this case we can easily enumerate the three feasible Hamiltonian cycles for the FCHT (see, Figure 3.1). For each one, we can compute the shortest path $d_H(i,j)$ for every pair of vertices and evaluate the total flow cost by multiplying the shortest distance times the flow requirement. The optimal solution is given by the Hamiltonian tour formed by the edges $\{\{1,3\},\{3,4\},\{4,2\},\{1,2\}\}$ with an objective value of

$3,500$.

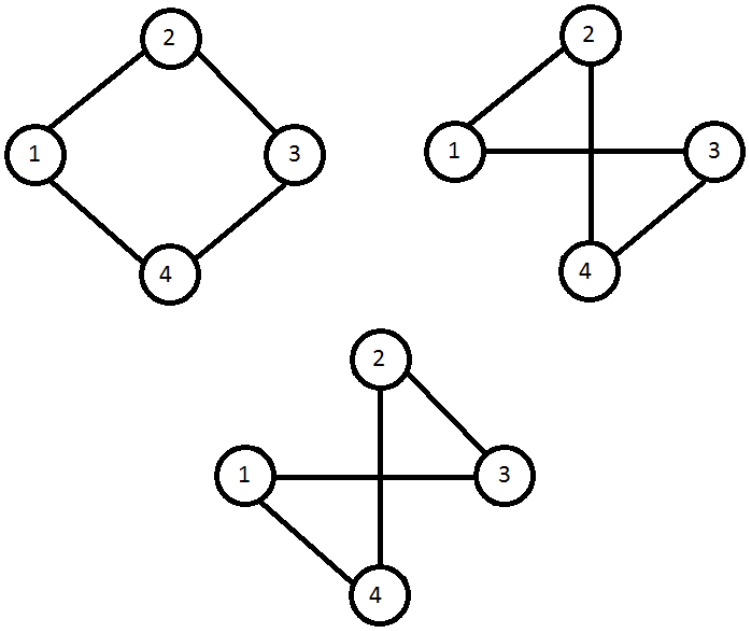


Figure 3.1: Feasible solutions.

In order to study the complexity of the FCHT we need first to introduce the decision version of the problem, denoted by FCHT-D, which can be stated as: is there a Hamiltonian cycle with total flow cost less than or equal to a given value k? Also, consider the Hamiltonian Cycle (HC-D) decision problem as: given an undirected graph, does it have a Hamiltonian cycle?

This problem is known to be *NP*-Complete (see Garey and Johnson (1990) and Cook et al. (1998)) and can be used to prove the following result.

Proposition 3.1. *The FCHT-D is NP-Complete.*

Proof. We give a polynomial-time reduction of the HC-D to the FCHT-D. Let $Q = (A, B)$ be an undirected graph. Let Q' be the complete graph on A. Define the flows for each pair of vertices as a positive constant w and the costs of the edges in Q' as

follows. Let $c_e = 0$ for each $e \in (Q \cap Q')$ and $c_e = 1$ for each $e \in Q' \backslash Q$. Then Q has a Hamiltonian cycle, if and only if there exists a Hamiltonian cycle in Q' of total flow cost less than or equal to zero. □

As a result, the FCHT belongs to the challenging class of *NP*-hard optimization problems and ad hoc mathematical models and solution methods should be used to solve it.

3.2 Related Optimization Problems

Network optimization is probably one of the most important and heavily studied research areas within combinatorial optimization. Broadly speaking, *network design problems* (NDP) consist in identifying an optimal subgraph of an undirected graph and routing flows between origin and destination vertices, subject to some feasibility conditions (Johnson et al., 1978; Ahuja et al., 1993). These problems frequently arise in the design of transportation or telecommunication networks, where commodities between pairs of vertices must be routed through the network. Two types of costs are usually considered in NDPs. The first one is the design cost, which is related to the activation (or construction) of the edges and/or vertices of the network. The second one is the operational cost, which corresponds to the routing of flow through the network. Some NDPs such as the *minimum spanning tree problem* (Kruskal, 1956) and the *maximum weight matching problem* (Edmonds, 1965), focus on the design costs. Other NDPs such as the *shortest path problem* (Bellman, 1958; Dijkstra, 1959), the *minimum flow cost problem* (see Ahuja et al., 1993), and the *optimum communication spanning tree problem* (Hu, 1974), focus on the operational costs of the network. A combination of both, design and operational costs has also been considered in several NDPs such as the *fixed-charge network flow problem* (Rardin and Wolsey, 1993) and

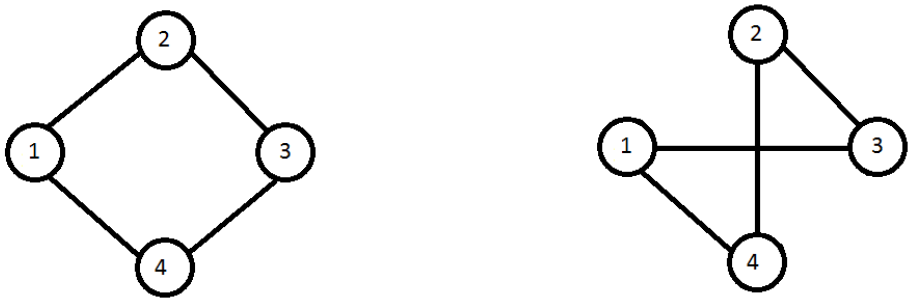
the *multicommodity network design problem* (Gendron et al., 1999).

As mentioned, one of the most well studied *NP*-hard problems is the *Traveling Salesman Problem* (see Dantzig et al., 1954; Öncan et al., 2009; Cook, 2011) and can be defined as follows. Let $G = (V, E)$ be an undirected graph where $|V| = n$ and consider a cost function c such that c_{ij} represents the cost between i and j. If $c_{ij} = c_{ji}$ $\forall i, j \in V$, the *symmetric traveling salesman problem* (STSP) can be defined as:

$$\text{(STSP)} \quad \min_{H \subseteq E} \left\{ \sum_{\{i,j\} \in H} c_{ij} : H \text{ is a Hamiltonian tour} \right\}.$$

The FCHT and TSP have the same set of feasible solutions, i.e., the set of all Hamiltonian cycles in G. However, an optimal solution for one problem may not be optimal for the other. Let us illustrate this situation with the instance used in example 3.1.

Example 3.2. *As previously stated we can enumerate the three Hamiltonian cycles corresponding to the feasible solutions for both the FCHT and STSP. Evaluating those solutions into the STSP objective function we observe that the optimal cycle is formed with the edges* $\{\{1,2\},\{2,3\},\{3,4\},\{4,1\}\}$ *and with an objective value equal to* 40*. From Example 3.1, we know that optimal cycle for the FCHT contains the edges* $\{\{1,3\},\{3,4\},\{4,2\},\{1,2\}\}$*. In Figure 3.2 the Hamiltonian cycles corresponding to the optimal solutions of both problems are shown.*



(a) Optimal Solution TSP (b) Optimal Solution FCHT

Figure 3.2: Comparison TSP and FCHT

From the previous example, we observe that the set of optimal solutions of the TSP and FCHT do not necessarily coincide and thus, an optimal solution for one problem may not be optimal for the other. In Table 3.1 we compare the objective values for the total flow cost as well as the total design cost for the three Hamiltonian tours.

	Design Cost	Flow Cost
TSP solution	40	6400
FCHT solution	45	3500
3rd solution	45	4400

Table 3.1: Objective values for all feasible solutions in Example 3.1.

Due to the importance of having Hamiltonian cycle structures, several variants, extensions and generalizations of the TSP have been studied. For instance, the *generalized traveling salesman problem* (Laporte et al., 1987), the *bottleneck traveling salesman problem* (Garfinkel and Gilbert, 1978), and *vehicle routing problems* (see Dantzig and Ramser, 1959; Solomon and Desrosiers, 1988; Laporte, 2009), are well known variants and generalizations of the TSP. A review on network design problems involving the Hamiltonian cycle structure can be found in Laporte and Rodríguez-Martín (2007). Lawler et al. (1985); Gutin and Punnen (2002); Applegate et al. (2006) are books providing an in-depth treatment on the TSP.

Another combinatorial optimization problem closely related to the FCHT is the *optimum communication spanning tree problem* (OCT). It considers the same operational costs as in the FCHT but has as the set of feasible solutions the set of spanning trees. Let $G = (V, E)$ be an undirected graph and let T be a tree associated with G. Consider the same definitions of W and C as before and let $d_T(i, j)$ denote the cost of the unique path from i to j in T. The OCT consists of finding a spanning tree that minimizes the total flow cost. The OCT can thus be defined as (see Hu, 1974):

$$(OCT) \quad \min_{T} \sum_{\{i,j\} \in E} w_{ij} d_T(i, j).$$

This problem is also known to be *NP*-hard. However, it is important to note that the *minimum spanning tree problem* (MST), which focuses on the design costs rather than on the operational costs, can be solved in polynomial time with several greedy algorithms (see Ahuja et al., 1993). The OCT was introduced by Hu (1974) and several works have focused on developing approximate and solution methods for it (Ahuja and Murty, 1987; Rothlauf, 2009; Contreras et al., 2010).

Another similar problem to the FCHT is the well-known *quadratic assignment problem* (QAP). Let F be a set of facilities and L a set of locations. Assume $|F| = |L| = n$ and also consider the matrices $W = [w_{ij}]_{n \times n}$ and $D = [d_{ij}]_{n \times n}$ corresponding to the flow between facilities and the distances between locations, respectively. That is, $w : F \times F \to \mathbb{R}$ and $d : L \times L \to \mathbb{R}$. The QAP considers the assignment of all facilities to different location with the objective of minimizing the sum of the distances times the corresponding flows. (see Koopmans and Beckmann, 1957; Frieze and Yadegar, 1983; Loiola et al., 2007). It can thus be defined as

$$(QAP) \qquad \min_{\phi \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} d_{\phi(i)\phi(j)},$$

where S_n is the set of all possible permutations of the set $\{1, \cdots, n\}$.

We note that the definition of the FCHT and the QAP are quite similar. However, notice that the distance in the QAP is defined between locations and in the FCHT we define it between vertices (facilities). To better illustrate this difference consider the following interpretation of the QAP. Let F be a set of professors and L a set of offices located in a circular hall. There is a flow or transit (w_{ij}) between every pair of professors and there is a distance between every pair of offices d_{ij}. The objective is to minimize the total flow cost of locating all professors to the offices. Take for instance, that we locate professors 1, 3 and 7 in the first 3 offices, say 1, 2 and 3, respectively. Then the distance $d_{\phi(1)\phi(7)} = d_{13}$. Now, if we switch professor 3 with professor 6 the distance between professors 1 and 7 will not change (d_{13}). On the other hand, if we do the same exercise with the FCHT we see that if we change one vertex of its ordering position it will affect the costs between every other pair of vertices since we have to recalculate the costs in the new Hamiltonian cycle. It is worth mentioning that until today, the QAP is considered one of the most difficult problems in combinatorial optimization. Instances with only 30 vetices were finally solved to optimality just a year ago (see Nyberg and Westerlund, 2012) and some instances with 60 vertices and one instance with 128 vertices were solved recently (see Fischetti et al., 2012).

The FCHT is also related to the *uncapacitated fixed-charge network flow problem* (UFC). This problem has been studied in the literature (see Rardin and Wolsey, 1993; Ortega and Wolsey, 2003) and it is well known to be *NP*-hard as well. Consider a digraph $D = (V, A)$, where V is the set of vertices and A is the set of arcs. One of the costs involved is the fixed cost f_{ij} of using the arc (i, j) to send flow and the other is a variable cost c_{ij} dependent on the amount of flow sent through the arc (i, j). The

objective is to minimize the total cost. The UFC can be stated as:

$$(UFC) \quad \min_{B \subseteq A} \sum_{i,j \in V} w_{ij} d_B(i,j) + \sum_{(i,j) \in B} f_{ij},$$

where $d_B(i,j)$ is the distance between i and j in the subset of arcs B.

If we set all fixed costs equal to zero and we add the constraint that the subset B should be a Hamiltonian cycle, we have that the FCHT is indeed a particular case of the UFC. Several mathematical models as well as different approximate and exact solution techniques have been proposed for solving UFCs (Gendron et al., 1999).

3.3 Applications

Potential applications of the FCHT arise naturally in telecommunications network design and in rapid transit systems planning. In the former case, cycle topologies are usually preferred when designing reliable networks. If an edge connecting two vertices fails for some reason, a cycle topology guarantees connectivity of the remaining subnetwork, and allows flows to be routed through alternative paths. For these problems usually it is assumed that a forecast of the amount of communication requirements between origin/destination pairs is known in advance and the objective is to minimize the communication cost after the network is built (see, Xu et al., 1998, for an example in data service design). Also, an extensive review of models and telecommunications applications considering the location of a cycle topology is given in Laporte and Rodríguez-Martín (2007). In the case of rapid transit systems, bus routes and metro lines are sometimes designed with a cycle structure (see Tanash et al., 2013) and the objective is to minimize the total travel time to serve users. The design of automated guided vehicles (AGV) networks is another relevant transportation application of the FCHT. The design of AGV networks consists in selecting

the optimal route for an automated vehicle that will visit a set of stations within a manufacturing or distribution facility. The network must provide a connection function (not necessarily in a direct way) between all pairs of stations to send and receive various commodities and the objective is to minimize the total flow cost. For some examples and a review of advantages of a cycle topology in AGV systems, such as the easiness to handle vehicle conflicts and having a less complicated network, we refer the reader to Asef-Vaziri and Goetschalckx (2008), Asef-Vaziri et al. (2007) and Qiu et al. (2002).

Chapter 4

Integer Programming Formulations

In this chapter we present five different mixed integer programming formulations for the FCHT. We analytically compare the quality of their linear programming relaxation bounds to establish a dominance (or equivalence) relationship between all of them.

We define the *graph of flows* G_F, as the undirected graph with vertex set V and an edge associated with each pair $(i, j) \in V \times V$ such that $w_{ij} + w_{ji} > 0$. We assume that G_F has one single connected component since, otherwise, the problem can be decomposed into several independent FCHTs, one for each connected component in G_F. In the case that a particular application requires one single cycle and the graph of flow contains more than one connected component, we would replace those flows equal to zero by $w_{ij} = \epsilon$ sufficiently small.

The objective of the FCHT is the minimization of the total cost of sending flow between pairs of vertices on an undirected Hamiltonian cycle. Contrary to the TSP, knowing the set of edges that define the cycle topology is not enough to evaluate the objective function. The length of the paths between pairs of vertices have to be computed to calculate the flow cost. For each pair of vertices, there are exactly

two possible paths on the cycle and, given that there are no capacity restrictions on the amount of flow routed on each edge, each flow has to be routed through the shortest path (among the two possible) containing an undetermined number of edges. Therefore, the formulations must be able to model (or reproduce) the shortest path used for routing each flow and guarantee that the subgraph associated with the arcs used in all the paths defines a Hamiltonian cycle. We present five different ways to do so.

4.1 A Path Based Formulation

One way of modeling the origin/destination (O/D) paths is by using path-based variables commonly used in network design problems (see for instance, Gendron et al., 1999). In particular, we consider two sets of binary decision variables, one to determine which arcs are used in a specific path and the other to determine whether an edge will be in the Hamiltonian tour or not. For each $i, j, r, s \in V$, we define binary variables

$$x_{ij}^{rs} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is used in the path between vertices } r \text{ and } s, \\ 0 & \text{otherwise.} \end{cases}$$

and for each edge $\{i, j\} \in E$, we introduce binary variables

$$y_{ij} = \begin{cases} 1 & \text{if edge } \{i,j\} \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that the definition of the x (path defining) variables are directed while the y (design) variables are not. The FCHT can be formulated as follows:

$$
\begin{aligned}
\text{(PBF)} \qquad \text{minimize} \quad & \sum_{r\in V}\sum_{s\in V}\sum_{i\in V}\sum_{j\in V} w_{rs}c_{ij}x_{ij}^{rs} & & \\
\text{subject to} \quad & \sum_{i\in V-\{r\}} x_{ri}^{rs} = 1 \qquad \forall r,s \in V & & (4.1)\\
& \sum_{i\in V-\{j\}} x_{ji}^{rs} - \sum_{i\in V-\{j\}} x_{ij}^{rs} = 0 \quad \forall j,r,s \in V,\ j \neq r,s & & (4.2)\\
& \sum_{i\in V-\{s\}} x_{is}^{rs} = 1 \qquad \forall r,s \in V & & (4.3)\\
& x_{ij}^{rs} + x_{ji}^{rs} \leq y_{ij} \qquad \forall r,s,i,j \in V,\quad j > i & & (4.4)\\
& \sum_{j>i} y_{ij} + \sum_{j<i} y_{ji} = 2 \qquad \forall i \in V & & (4.5)\\
& x_{ij}^{rs} \in \mathbb{R}_{+} \qquad \forall r,s,i,j \in V & & (4.6)\\
& y_{ij} \in \{0,1\} \qquad \forall i,j \in V. & & (4.7)
\end{aligned}
$$

Constraints (4.1) – (4.3) are the well-known flow conservation constraints for each pair of vertices $r, s \in V$. The set of inequalities in (4.4) ensure that if the edge $\{i, j\}$ is not used in the final network then no flow must pass through it. Constraints (4.5) ensure that there will only be n edges in the final network and that each vertex must have degree equal to two. The combination of all constraints (4.1)–(4.7) will create paths between all pair of vertices and will be a subgraph with a single connected component with exactly n edges, and thus any feasible solution of this system of constraints will be a Hamiltonian cycle of G. As a consequence, classical subtour elimination constraints, commonly required to model the TSP, are not necessary when using this formulation. Finally, note that this formulation requires $O(n^4)$ variables and $O(n^4)$ constraints to model the problem.

In order to illustrate the previous formulation we refer to the instance of Example 3.1. For this instance, the x variables that take value one in the optimal solution are $\{x_{14}^{12},\ x_{42}^{12},\ x_{13}^{13},\ x_{14}^{14},\ x_{23}^{23},\ x_{24}^{24},\ x_{31}^{34},\ x_{14}^{34}\}$ and the rest of them are equal to zero.

4.2 A Second Path Based Formulation

This formulation also considers decision variables to model the path between pairs of vertices. However, we now define the set of all possible paths between any two vertices r and s, denoted as $P_{rs} = \{p_1, \ldots, p_k, \ldots, p_q\}$. For each $r, s \in V$ and each $p_k \in P_{rs}$, we define binary decision variables

$$z_{p_k}^{rs} = \begin{cases} 1 & \text{if path } p_k \text{ is used to connect vertices } r \text{ and } s, \\ 0 & \text{otherwise.} \end{cases}$$

Combining these variables with the y design variables previously presented, the FCHT can be stated as:

$$\begin{aligned}
\text{(PBF2)} \quad \text{minimize} \quad & \sum_{r,s \in V} \sum_{p_k \in P_{rs}} w_{rs} \left(\sum_{(i,j) \in p_k} c_{ij} \right) z_{p_k}^{rs} \\
\text{subject to} \quad & \sum_{p_k \in P_{rs}} z_{p_k}^{rs} = 1 \qquad \forall r, s \in V & (4.8) \\
& \sum_{p_k \in P_{rs} : (i,j) \in p_k} z_{p_k}^{rs} \leq y_{ij} \qquad \forall r, s, i, j \in V & (4.9) \\
& \sum_{j>i} y_{ij} + \sum_{j<i} y_{ji} = 2 \qquad \forall i \in V & (4.10) \\
& z_{p_k}^{rs} \in \mathbb{R}_+ \qquad \forall r, s \in V \text{ and } \forall p_k \in P_{rs} & (4.11) \\
& y_{ij} \in \{0, 1\} \qquad \forall i, j \in V. & (4.12)
\end{aligned}$$

Constraints (4.8) ensure that exactly one path is assigned for each pair of vertices while constraints (4.9) ensure that if the edge $\{i, j\}$ is not in the final network, then no path containing (i, j) will be used. Constraints (4.10) are the same as defined in PBF and FBF to guarantee the degree of each vertex. Similar to the previous formulation, the combination of all constraints (4.8)–(4.12) will create paths between

all pair of vertices and thus, subtour elimination constraints are not necessary for this formulation. Finally, observe that the number of variables in this formulation grows exponentially with the number of vertices. In particular, there is one variable for each possible path between vertices r and s, and the number of simple paths between two vertices in a complete graph (worst-case scenario) is given by $\sum_{i=0}^{n-2} \frac{(n-2)!}{(n-2-i)!}$ (Rosen, 2007). Therefore, the formulation contains $O(n^{n-2})$ variables and thus column generation techniques are needed to handle it.

Taking the instance of Example 3.1, the $z_{p_k}^{rs}$ variables that take value one in the optimal solution are $\{z_{p_k}^{12}, z_{p_k}^{13}, z_{p_k}^{14}, z_{p_k}^{23}, z_{p_k}^{24}, z_{p_k}^{34}\}$ with the corresponding paths shown in table 4.1. The rest of the $z_{p_k}^{rs}$ variables take value zero.

r	s	p_k
1	2	{{1,4},{4,2}}
1	3	{{1,3}}
1	4	{{1,4}}
2	3	{{2,3}}
2	4	{{2,4}}
3	4	{{3,1},{1,4}}

Table 4.1: Path variables

4.3 A Flow Based Formulation

Another common way to model network design problems is to use flow based variables that compute the amount of flow that is routed on a particular arc (see Frangioni and Gendron, 2007). In particular, for each $r, i, j \in V$ we define the real-valued decision variables

$$x_{rij} : \text{amount of flow going through arc } (i, j) \text{ originated at vertex } r.$$

Combining these variables with the y design variables previously presented, the FCHT can be formulated as:

$$\text{(FBF)} \quad \text{minimize} \quad \sum_{r\in V}\sum_{i\in V}\sum_{j\in V, j\neq i} c_{ij}x_{rij}$$

$$\text{subject to} \quad \sum_{i\in V} x_{rij} - \sum_{i\in V} x_{rji} = w_{rj} \qquad \forall r, j \in V r \neq j \quad (4.13)$$

$$x_{rji} + x_{rij} \leq M_r y_{ij} \qquad \forall r, i, j \in V \quad i < j \quad (4.14)$$

$$\sum_{j>i} y_{ij} + \sum_{j<i} y_{ji} = 2 \qquad \forall i \in V \quad (4.15)$$

$$x_{rij} \in \mathbb{R}_+ \qquad \forall r, i, j \in V \quad (4.16)$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j \in V, \quad (4.17)$$

where $M_r = \sum_{s\in V} w_{rs}$ is the total outgoing flow from vertex r. Constraints (4.13) are the flow conservation constraints for each pair of vertices $r, j \in V$ while constraints (4.14) ensure that if we do not use the edge between i and j in the final network then it will not be used to send any amount of flow. The set of constraints (4.15) are the same ones as in the previous formulations. The assumption that the graph of flows G_F contains a single connected component together with constraints (4.13)–(4.17) eliminates the need of including subtour elimination constraints to obtain a valid formulation. Finally, note that this formulation requires $O(n^3)$ variables and $O(n^3)$ constraints to model the problem, a considerable reduction with respect to previous formulations.

Using the instance of Example 3.1, the x_{rij} variables with a non-zero value are shown in Table 4.2.

r	i	j	x_{rij}
1	1	4	110
1	1	3	10
1	4	2	10
2	2	3	100
2	2	4	200
3	2	4	10
3	3	2	10

Table 4.2: Flow variables

4.4 A Two-index Formulation

The following formulation is able to further reduce the number of variables to $O(n^2)$ at the expense of considerably increasing the number of constraints. However, for this formulation to be valid, we need to assume that costs are symmetric, i.e., $c_{ij} = c_{ji}$. The main idea of this formulation is to construct a directed Hamiltonian cycle and to compute the length of the unique directed path between all pairs of vertices using a set of continuous (real-valued) decision variables. Then, another set of variables will determine which path is the shortest one to go from an origin vertex r to a destination vertex s, the directed path from r to s or the directed path from s to r. To this end, for each $r, s \in V$ we define the real-valued decision variables

$$d_{rs} : \text{directed distance from vertex } r \text{ to vertex } s \text{ on the cycle}$$

For each $i, j \in V$, we also define binary decision variables

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that the new (design) variables x_{ij} are directed while the previous y_{ij} variables are not. The FCHT can then be formulated as:

$$\text{(NL2IF)} \quad \text{minimize} \quad \sum_{r\in V}\sum_{s\in V} w_{rs}\min\{d_{rs}, d_{sr}\} \tag{4.18}$$

$$\text{subject to} \quad \sum_{i\in V} x_{ij} = 1 \qquad \forall j \in V \tag{4.19}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \tag{4.20}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \qquad \forall S \subseteq V,\ \ 2 \leq |S| < n \tag{4.21}$$

$$d_{rs} \geq \left(1 - |p_k| + \sum_{(i,j)\in p_k} x_{ij}\right)\left(\sum_{(i,j)\in p_k} c_{ij}\right)$$
$$\forall r, s \in V \text{ and } \ \forall p_k \in P_{rs} \tag{4.22}$$

$$d_{rs} \in \mathbb{R}_+ \qquad \forall r, s \in V \tag{4.23}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall i, j \in V \tag{4.24}$$

where P_{rs} denotes the set of all possible paths between vertices r and s and $|p_k|$ denotes the number of edges in the path p_k. Constraints (4.19), (4.20) are the degree constraints stating that each vertex has exactly one arc entering and one arc leaving the vertex. Constrains (4.21) are the so-called subtour elimination constraints that forbid cycles containing less than n arcs. Constraints (4.22) compute the directed distance between two vertices, given by the sum of the individual costs of all edges contained in the solution path.

It is worth mentioning that constraints (4.19)–(4.21), (4.24) are known as the TSP polytope which is precisely the classical formulation of the TSP (Dantzig et al., 1954). However, in the case of the FCHT constraints (4.22) are needed in order to compute the flow cost and they grow exponentially with the number of vertices, since there is one for each possible path between every pair of vertices. In addition, the objective

function (4.18) is nonlinear due to the min function. The solution of nonlinear integer programs with general purpose methods is still very limited and a common approach is to linearize the nonlinear terms. This can be done in our case by adding some extra decision variables and constraints. For each $r, s \in V$ we define the real-valued decision variables

$$d_{rs}^{min} : \text{minimum undirected distance between vertices } r \text{ and } s \text{ on the cycle,}$$

and the binary decision variables

$$z_{rs} = \begin{cases} 1 & \text{if the flow from } r \text{ to } s \text{ is sent according with the direction of the cycle,} \\ 0 & \text{otherwise.} \end{cases}$$

The FCHT can be formulated as the following linear integer program:

$$\begin{aligned}
\text{(2IF)} \quad \text{minimize} \quad & \sum_{r \in V} \sum_{s \in V} w_{rs} d_{rs}^{min} \\
\text{subject to} \quad & (4.19) - (4.24) \\
& d_{rs}^{min} \geq d_{rs} - M(1 - z_{rs}) \qquad \forall r, s \in V,\ r \neq s \qquad (4.25) \\
& d_{rs}^{min} \geq d_{sr} - M(1 - z_{sr}) \qquad \forall r, s \in V,\ r \neq s \qquad (4.26) \\
& z_{rs} + z_{sr} = 1 \qquad \forall r, s \in V,\ r \neq s \qquad (4.27) \\
& d_{rs}^{min} \in \mathbb{R}_+ \qquad \forall r, s \in V. \qquad (4.28)
\end{aligned}$$

where M is a sufficiently large number, in this case $M = \sum_{i,j \in V : i<j} c_{ij}$. The sets of constraints (4.25) and (4.26) ensure that for each pair of vertices we consider the distance cost of one of the two possible paths in the solution cycle. Constraints (4.27) ensure that exactly one of them will be selected, which together with the fact that we are minimizing the objective (4.25), the shortest path is always going to be selected.

To illustrate this formulation consider the instance of Example 3.1. The optimal solution values for the d_{rs}^{min} and z_{rs} variables are

r	s	d_{rs}^{min}	z_{rs}
1	2	15	0
1	3	20	1
1	4	10	0
2	3	10	0
2	4	5	1
3	4	15	1

Table 4.3: Solution example 3.1

4.5 A Second Two-index Formulation

This last formulation is able to model the problem with only $O(n^2)$ variables and $O(n^2)$ constraints, a considerable reduction with respect to the previous formulations. It is based on the idea of using an arbitrary origin vertex and compute the directed distance on the solution cycle from this vertex to the rest of them. By doing so we can model the directed distance between pair of vertices without using an exponentially growing number of constraints that considers all possible paths. In particular, for each $j \in V$ we define the real-valued decision variables

$$g_j : \text{distance between vertex 1 and vertex } j.$$

Also we use x_{ij} and d_{rs} variables previously used in section 4.4. The FCHT can be stated as:

$$\text{(NL2IF2)} \quad \text{minimize} \quad \sum_{r\in V}\sum_{s\in V} w_{rs}\min\{d_{rs}, d_{sr}\} \tag{4.29}$$

$$\text{subject to} \quad \sum_{i\in V} x_{ij} = 1 \qquad \forall j \in V \tag{4.30}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \tag{4.31}$$

$$g_j \geq (g_i + c_{ij})x_{ij} \qquad \forall i, j \in V, j \neq 1 \tag{4.32}$$

$$d_{rs} = \begin{cases} g_s - g_r, & \text{if } g_s \geq g_r, \\ \sum\limits_{i\in V}\sum\limits_{j\in V} c_{ij}x_{ij} - (g_r - g_s), & \text{if } g_s < g_r. \end{cases} \tag{4.33}$$

$$\forall r, s \in V$$

$$d_{rs} \in \mathbb{R}_+ \qquad \forall r, s \in V \tag{4.34}$$

$$g_j \in \mathbb{R}_+ \qquad \forall j \in V \tag{4.35}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V \tag{4.36}$$

Constraints (4.30)–(4.31) have the same meaning as in the previous formulation. Constraints (4.32) compute the cumulative cost between vertex 1 and the rest of the vertices. Constraints (4.33) evaluate the directed distance between two vertices using the cumulative distance given in g_j variables.

This model is nonlinear due to the objective function (4.29) and the set of constraints (4.32) and (4.33). We linearize the objective function as in the previous formulation. To linearize the constraints, we need to define additional variables. For each $r, s, \in V$, we define binary decision variables

$$f_{rs} = \begin{cases} 1 & \text{if vertex } r \text{ precedes vertex } s \text{ in the solution cycle,} \\ 0 & \text{otherwise.} \end{cases}$$

Using as well the d_{rs}^{min} and z_{rs} variables presented in the previous model, the FCHT can be stated as:

$$\begin{aligned}
\text{(2IF2) minimize} \quad & \sum_{i \in V} \sum_{j \in V} w_{rs} d_{rs}^{min} \\
\text{subject to} \quad & (4.19) - (4.20), (4.25) - (4.27) \\
& g_j \geq g_i + c_{ij} - M(1 - x_{ij}) \qquad \forall i, j \in V && (4.37) \\
& g_j \leq g_i + c_{ij} + M(1 - x_{ij}) \qquad \forall i, j \in V && (4.38) \\
& d_{rs} \geq g_s - g_r \qquad \forall r, s \in V && (4.39) \\
& d_{rs} \geq \sum_{i,j \in V} c_{ij} x_{ij} - (g_r - g_s) - M(1 - f_{sr}) \quad \forall r, s \in V && (4.40) \\
& f_{rs} \geq \frac{g_s - g_r}{M} \qquad \forall r, s \in V && (4.41) \\
& f_{rs} \leq 1 + \frac{g_s - g_r}{M} \qquad \forall r, s \in V && (4.42) \\
& d_{rs}, d_{rs}^{min} \in \mathbb{R}_+ \qquad \forall r, s \in V && (4.43) \\
& x_{ij}, z_{rs}, f_{rs} \in \{0, 1\} \qquad \forall r, s, i, j \in V && (4.44)
\end{aligned}$$

where M is a sufficiently large number, in this case $M = \sum_{i,j \in V : i<j} c_{ij}$. Constraints (4.37) and (4.38) compute the accumulated cost from vertex 1 to every other vertex j. Constraints (4.39) and (4.40) evaluate the cost of the two possible paths in the final network between vertices r and s. Constraints (4.41) and (4.42) ensure the correct computation of the distances for all pairs of vertices by considering only the case when $f_{ij} = 1$, i.e., when $g_j \geq g_i$. Note that constraints (4.37) are similar to the sub-tour elimination constraints proposed in Miller et al. (1960). It is known that these guarantee that there are no subtours when working on a complete graph G, since $c_{ij} \neq 0$ for $i \neq j$ ensures the ordering of the vertices by their corresponding accumulated distance to vertex 1.

Using the instance of Example 3.1 we obtain the following optimal solution values for the variables:

r	s	d_{rs}^{min}	z_{rs}	f_{rs}	g_s
1	2	15	0	1	30
1	3	20	1	1	20
1	4	10	0	1	35
2	3	10	0	0	
2	4	5	1	1	
3	4	15	1	1	

Table 4.4: Solution example 3.1

4.6 A Comparison of Bounds

We now analytically compare the quality of the LP relaxation bounds of the five MIP formulations previously introduced. By doing so, we provide a dominance (or equivalence) relationship between all of them. For this purpose we use the definitions presented in Section 2.2.

Proposition 4.1. *The value of the LP relaxation for the PBF coincides with the value of the LP relaxation for PBF2.*

Proof. Let $L^1(\lambda)$ be the following Lagrangian relaxation for PBF.

$$L^1(\lambda) = \min \quad \sum_{r\in V}\sum_{s\in V}\sum_{i\in V}\sum_{j\in V} w_{rs}c_{ij}x_{ij}^{rs} \tag{4.45}$$
$$+\sum_{r\in V}\sum_{s\in V}\sum_{i\in V}\sum_{j\in V} \lambda_{ij}^{rs}(x_{ij}^{rs} + x_{ji}^{rs} - y_{ij})$$

$$\text{subject to} \quad \sum_{i\in V-\{r\}} x_{ri}^{rs} = 1 \qquad \forall r, s \in V \tag{4.46}$$

$$\sum_{i\in V-\{j\}} x_{ji}^{rs} - \sum_{i\in V-\{j\}} x_{ij}^{rs} = 0 \quad \forall r, s, j \in V \text{ for } j \neq r, s \tag{4.47}$$

$$\sum_{i\in V-\{s\}} x_{is}^{rs} = 1 \qquad \forall r, s \in V \tag{4.48}$$

$$\sum_{j>i} y_{ij} + \sum_{j<i} y_{ji} = 2 \quad \forall i \in V \tag{4.49}$$

$$x_{ij}^{rs} \in \mathbb{R}_+ \qquad \forall r, s, i, j \in V \tag{4.50}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j \in V. \tag{4.51}$$

Note that $L^1(\lambda)$ can be separated into two subproblems: (i) a problem in the x variables (4.46)–(4.48) and (4.50), and (ii) a problem in the y variables (4.49) and (4.51). Subproblem (i) can be decomposed into $|R|$ shortest path subproblems, each one has to select between two possible paths, where $R = \{(r, s) : r, s \in V,\ w_{rs} > 0\}$. It is well known that the LP relaxation of the given formulation for (i) will coincide with the integer optimal solution (Wolsey, 1998). Furthermore, constraints of subproblem (ii) can be rewritten as $\sum_{j\neq i} y_{ij} = 2 \ \ \forall i \in V$ and we can show that the associated matrix is totally unimodular. To prove this, we need to check that all entries in the matrix are 0 or 1 and that for any subset F of the rows, there is a partition $(F1, F2)$ of F such that each column j satisfies $|\sum_{i\in F1} a_{ij} - \sum_{i\in F1} a_{ij}| \leq 1$. Take for instance $F1 = F$ and $F2 = \emptyset$. Then, this Lagrangian problem has the integrality property and thus, the bound obtained with the Lagrangian dual coincides with the value of the LP relaxation of PBF.

On the other hand, consider the following Lagrangian problem resulting from relaxing constraints (4.9) in PBF2.

$$L^2(\lambda) = \text{minimize} \quad \sum_{r,s\in V}\sum_{p_k\in P_{rs}}\sum_{(i,j)\in p_k} w_{rs}c_{ij}z^{rs}_{p_k} + \sum_{r,s\in V}\sum_{i,j\in V}\lambda^{rs}_{ij}\left(\sum_{p_k\in P_{rs}:(i,j)\in p_k} z^{rs}_{p_k} - y_{ij}\right) \tag{4.52}$$

$$\text{subject to} \quad \sum_{p_k\in P_{rs}} z^{rs}_{p_k} = 1 \qquad \forall r,s \in V \tag{4.53}$$

$$\sum_{j>i} y_{ij} + \sum_{j<i} y_{ji} = 2 \qquad \forall i \in V \tag{4.54}$$

$$z^{rs}_{p_k} \in \{0,1\} \qquad \forall r,s \in V \tag{4.55}$$

$$y_{ij} \in \{0,1\} \qquad \forall i,j \in V \tag{4.56}$$

Note that this problem can also be decomposed into two subproblems: (i) a subproblem in the space of the z variables with (4.53) and (4.55), and (ii) a subproblem in the y variables with (4.54) and (4.56). Subproblem (ii) was explained in the previous part of the proof for $L^1(\lambda)$. Now, subproblem (i) can be decomposed into $|R|$ semi-assignment problems, one for each pair $\{r,s\}$ with $w_{rs} > 0$. It is known that for this formulation of (i) the LP solution is integral. Then, this Lagrangian problem also has the integrality property and thus, the bound obtained with its Lagrangian dual coincides with the value of the LP relaxation of PBF2.

Finally, since both Lagrangean relaxations coincide with their corresponding LP relaxations, we compare L^1 and L^2 in order to complete the proof. Given a binary vector $\hat{y}$, let $T_{rs} = \{p_k \in P_{rs} : \text{conditions (4.49) are satisfied}\}$ be a family of feasible paths. This uniquely defines a feasible solution $\hat{x}$ to $L^1(\lambda)$ and a feasible solution $\hat{z}$ to $L^2(\lambda)$. In particular, $\hat{x}^{rs}_{ij} = 1$ if and only if $(i,j) \in p_k \in T_{rs}$ and $\hat{z}^{rs}_{p_k} = 1$ if and only

if $p_k \in T_{rs}$. Now, the contribution of $\hat{x}$ to the objective function of $L^1(\lambda)$ is given by

$$\sum_{r,s \in V} \sum_{(i,j) \in p_k : p_k \in T_{rs}} \left(w_{rs} c_{ij} + \lambda_{ij}^{rs} \right)$$

which coincides with the contribution of the $\hat{z}$ variables to $L^2(\lambda)$ and the result follows. □

Proposition 4.2. *The value of the LP relaxation of the PBF is at least as good as the value of the LP relaxation of the FBF. That is, $LP_{FBF} \leq LP_{PBF}$.*

Proof. To prove this result we need to show that every feasible solution that is in the LP relaxation of PBF is also in the LP relaxation of FBF. Let $\hat{x}$ be a feasible solution to the LP relaxation of PBF. After multiplying (4.2) and (4.3) by w_{rs} we obtain the following result

$$w_{rs} \left(\sum_{i \in V \setminus \{j\}} \hat{x}_{ji}^{rs} - \sum_{i \in V \setminus \{j\}} \hat{x}_{ij}^{rs} \right) = 0 \qquad \forall r, s, j \in V \quad \text{with } j \neq r, s \tag{4.57}$$

$$w_{rj} \left(\sum_{i \neq s} \hat{x}_{is}^{rs} \right) = w_{rj} \qquad \text{for } \; j = s \quad \forall r, s, j \in V \tag{4.58}$$

Summing (4.57) over all $s \in V \setminus \{j\}$ and then adding (4.58) we have

$$\begin{aligned}
w_{rj} &= \sum_{s \neq r,j} \left[w_{rs} (\sum_{i \neq j} \hat{x}_{ji}^{rs} - \sum_{i \neq j} \hat{x}_{ij}^{rs}) \right] + w_{rj} \sum_{i \neq j} \hat{x}_{ij}^{rj} \\
&= \sum_{s \neq r,j} \left[w_{rs} (\sum_{i \neq j} \hat{x}_{ji}^{rs} - \sum_{i \neq j} \hat{x}_{ij}^{rs}) \right] + w_{rj} \left[\sum_{i \neq j} \hat{x}_{ij}^{rj} - 0 \right] \\
&= \sum_{s \neq r,j} \left[w_{rs} (\sum_{i \neq j} \hat{x}_{ji}^{rs} - \sum_{i \neq j} \hat{x}_{ij}^{rs}) \right] + w_{rj} \left[\sum_{i \neq j} \hat{x}_{ij}^{rj} - \sum_{i \neq j} \hat{x}_{ji}^{rj} \right] \\
&= \sum_{s \neq r} \left[w_{rs} (\sum_{i \neq j} \hat{x}_{ji}^{rs} - \sum_{i \neq j} \hat{x}_{ij}^{rs}) \right] = \sum_{i \neq j} \sum_{s \neq r} w_{rs} \hat{x}_{ij}^{rs} - \sum_{i \neq j} \sum_{s \neq r} w_{rs} \hat{x}_{ji}^{rs}.
\end{aligned}$$

Recall that in Section 4.3 we define the flow variables for FBF as x_{rij}. Abusing the notation, in this proof we use X_{rij} to represent the same variables in order to avoid any conflict with the x_{ij}^{rs} variables. Thus, we consider the term

$$\hat{X}_{rij} = \sum_{s \neq r} w_{rs} \hat{x}_{ij}^{rs}, \tag{4.59}$$

then we can rewrite the previous expression as

$$w_{rj} = \sum_{i \neq j} \hat{X}_{rij} - \sum_{i \neq j} \hat{X}_{rji} \qquad \forall r, j \in V \text{ with } \ r \neq j.$$

Which corresponds to the set of constraints given in (4.13). On the other hand, we have that for all $r, s, i, j \in V$ with $i < j$ and $r \neq s$, after multiplying (4.4) by $w_{rs} (\geq 0)$ the following expression holds

$$w_{rs} \hat{x}_{ij}^{rs} + w_{rs} \hat{x}_{ji}^{rs} \leq w_{rs} \hat{y}_{ij}.$$

Summing over all $s \in V$ we obtain

$$\sum_{s \neq r} w_{rs} \hat{x}_{ij}^{rs} + \sum_{s \neq r} w_{rs} \hat{x}_{ji}^{rs} \leq \sum_{s \neq r} w_{rs} \hat{y}_{ij} = \hat{y}_{ij} M_r,$$

and after rewriting the expression using (4.59) we obtain

$$\hat{X}_{rij} + \hat{X}_{rji} \leq M_r \hat{y}_{ij}.$$

Since any feasible solution of the LP relaxation of PBF defines a feasible solution for the LP relaxation of FBF, the result follows. □

Proposition 4.3. *The value of the LP relaxation of the 2IF is equal to zero.*

Proof. Note that, without loss of generality, we can transform an instance with an even number of vertices in V to an instance with an odd number of vertices. This may be done by adding an extra vertex with cost $c_{1n+1} = 0$ and the same distances as vertex 1 to the others vertices (i.e. $c_{1j} = c_{(n+1)j}$). The flow might be considered as a constant w_{1n+1} to vertex one but zero to the rest. This process will produce an equivalent problem with the same objective value and the same optimal solution. For this reason, we only prove the result when $|V|$ is odd. We recall that the FCHT is defined for $n > 3$.

For any instance with $|V| > 3$ and odd, we define for each $i \in V$ the x_{ij} variables with a strictly positive value as follows

$$\hat{x}_{ii+1} = \hat{x}_{ii+3} = \cdots = \hat{x}_{ii+(n-2)} = \hat{x}_{ii-2} = \hat{x}_{ii-4} = \cdots = \hat{x}_{ii-(n-1)} = \frac{1}{(n-1)/2}, \quad (4.60)$$

and the rest of the x_{ij} variables are set to zero.

We now prove that for any odd number n, constraints (4.19), (4.20) and (4.21), are satisfied with the previous definition of $\hat{x}$. Note that for each $j \in V$ there are exactly $\frac{n-1}{2}$ non-zero x_{ij} (and x_{ji}) variables.

$$\sum_{i \in V} \hat{x}_{ij} = \frac{n-1}{2}\frac{2}{n-1} = 1 \qquad \forall j \in V,$$

$$\sum_{j \in V} \hat{x}_{ij} = \frac{n-1}{2}\frac{2}{n-1} = 1 \qquad \forall i \in V.$$

Also we have that for any set $S \subseteq V$ with $2 \leq |S| < n$,

$$\sum_{i \in S}\sum_{j \in S} \hat{x}_{ij} \leq \frac{(|S|-1)|S|}{2}\frac{2}{(n-1)} \leq \frac{(|S|-1)(n-1)}{2((n-1)/2)} = |S| - 1.$$

We now construct the sets of the remaining variables (i.e., d_{rs}^{min}) that will give an

objective value equal to zero. Notice that for any pair of vertices r and s and any path p_k between them, when $|p_k| > 1$, the distance constraint (4.22) becomes $d_{rs} = 0$, since $0 \leq \sum_{(i,j)\in p_k} \hat{x}_{ij} \leq |p_k| - 1$, because for each edge added to the path the sum of the $\hat{x}$ variables in the path will increase at most $2/(n-1) < 1$. Considering also the non-negativity constraints for the x variables, the inequality will then take value $d_{rs} = 0$. For the case of $|p_k| = 1$, i.e., when the vertices r and s are directly connected, the value of the directed distance d_{rs} will be strictly greater than zero $(2/(n-1))$. However the distance d_{sr} will be zero because by definition of the $X = [\hat{x}_{ij}]$ matrix if r and s are connected by an arc then s and r are not, which means there will be more than one arc in the path from s to r, therefore $d_{rs}^{min} = 0$ and the total objective value will also be zero. □

For instance, for $n = 5$, using (4.60) the X matrix is given by:

$$\begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 & 0 \end{pmatrix}$$

Although the previous proof does not require us to compute the X matrix for the case when n is even, we have obtained the result for a few instances with n even. However, we have not yet been able to generalize such matrix. One solution for $n = 6$ would be

$$\begin{pmatrix} 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & 2/3 & 0 & 1/3 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 2/3 \\ 0 & 1/3 & 0 & 0 & 2/3 & 0 \\ 2/3 & 0 & 1/3 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 2/3 & 0 & 0 \end{pmatrix}$$

Proposition 4.4. *The value of the LP relaxation for the 2IF2 is zero.*

Proof. We recall that for this formulation we assume $c_{ij} \neq 0$ for $i \neq j$, and $M = \sum_{i,j \in V: i<j} c_{ij}$. We define the following fractional values for the x_{ij} (structural) variables.

$$\hat{x}_{ii} = 0 \qquad \forall i = 1, \cdots n.$$

For $q = 2, \cdots, n-2$ we have

$$\hat{x}_{i(i+q)} = \hat{x}_{nq} = \hat{x}_{(n-1)(q-1)} = \cdots = \hat{x}_{(n-(q-1))1} = \sum_{k>q+1} c_{(q+1)k}/M \qquad \forall i = 1, \cdots n-q.$$

Also, for $q = 1$ and $q = n-1$ we have, respectively,

$$\hat{x}_{n1} = \hat{x}_{i(i+1)} = \sum_{k>2} (c_{2k}/M) + c_{1n}/M \qquad \forall i = 1, \cdots, n-1,$$

$$\hat{x}_{1n} = \hat{x}_{i(i-1)} = \sum_{k>1} (c_{1k}/M) - c_{1n}/M \qquad \forall i = 2, \cdots, n.$$

We need this particular definition of the $\hat{x}_{1n}$ and $\hat{x}_{n1}$ variables when proving that the accumulated distances are zero.

We first prove that constraints (4.30) and (4.31) are satisfied using this definition.

It is possible to check that for each row and for each column in the $X = [x_{ij}]_{n\times n}$ matrix there is exactly one variable with value $\sum_{k>\alpha} c_{\alpha k}$ for $\alpha \in \{2, \cdots, n-1\}$ and one variable with the corresponding values of x_{1n} and x_{n1}. Furthermore, each row and each column contain all possible values of $\sum_{k>\alpha} c_{\alpha k}$, then if we take the respective sums we will get $\sum_{i\in V} x_{ij} = M/M = 1$ and $\sum_{j\in V} x_{ij} = M/M = 1$

Now, we show that the objective value for the LP relaxation is zero using the $\hat{x}$ variables and the fact that $g_1 = 0$. For all $j \neq 1$ we have that (4.37) holds, i.e.

$$g_j \geq g_1 + c_{1j} - M(1 - x_{1j}).$$

This implies that for $j \neq n$, using the $\hat{x}$ variables, we get

$$g_j \geq c_{1j} - M + M(\sum_{k>\alpha} c_{\alpha k}/M)$$

$$g_j \geq c_{1j} - M + \sum_{k>\alpha} c_{\alpha k} \quad \text{with} \quad \alpha \in \{2, \cdots, n-1\}.$$

Since $c_{1j} - M + \sum_{k>\alpha} c_{\alpha k} \leq 0$ then $g_j = 0$ due to the non-negativity constraints. This is also the case for $j = n$ because we define x_{1n} without c_{1n} ensuring that the total value is less than or equal to zero. Therefore all $g_j = 0$ and thus the $d_{rs}^{min} = 0$. □

Combining all the results from this section, we obtain the following corollary.

Corollary 4.1. $0 = LP_{2IF1} = LP_{2IF2} \leq LP_{FPF} \leq LP_{PBF} = LP_{PBF2}$.

This means, that in terms of the linear programming relaxation bounds, the best formulations are the path based formulations (PBF) and (PBF2). Also, the two-index formulations have the worst lower bound possible, even if we have an number of constraints growing exponentially with n.

Chapter 5

Solution Methods

In this chapter we present some approximate and exact solution methods to solve the FCHT. The need to develop specialized methods arises not only from the fact that the problem is *NP*-hard, but because as it is shown in the next chapter, state-of-the-art commercial solvers can only solve small size instances. We first present a combinatorial bound based on shortest paths and two heuristic methods that can be used to obtain lower and upper bounds, respectively, on the optimal solution of the problem. We then present an exact branch-and-cut method to obtain the optimal solution for the FCHT.

5.1 Combinatorial Bounds

The main idea behind this combinatorial bound is to use the information on the length of shortest paths between all pair of vertices. In particular, we compute the shortest path for every $r, s \in V$ and then we use this value as the distance to compute the flow cost for routing the flow w_{rs}. We define the shortest path distance between r and s on G as d^*_{rs}.

Proposition 5.1. *The shortest path relaxation (SPR) problem*

$$
\begin{aligned}
\sum_{r,s\in V} w_{rs}d^*_{rs} \quad \equiv \quad & \textit{minimize} \sum_{r,s\in V} w_{rs}\left(\sum_{i,j\in V} c_{ij}x^{rs}_{ij}\right) \\
\textit{subject to} \quad & \sum_{i\in V-\{r\}} x^{rs}_{ri} = 1 \qquad \forall r,s \in V \\
& \sum_{i\in V-\{j\}} x^{rs}_{ji} - \sum_{i\in V-\{j\}} x^{rs}_{ij} = 0 \qquad \forall j,r,s \in V\, j \neq r,s \\
& \sum_{i\in V-\{s\}} x^{rs}_{is} = 1 \qquad \forall r,s \in V \\
& x^{rs}_{ij} \in \{0,1\} \qquad \forall r,s,i,j \in V
\end{aligned}
$$

is a relaxation of FCHT and thus, $\sum_{r,s\in V} w_{rs}d^*_{rs}$ *is a lower bound on the optimal solution value of FCHT.*

Proof. Following Definition 2.13, the first condition is satisfied trivially. In order to prove that $\sum_{r,s\in V} w_{rs}d^*_{rs}$ is always less than or equal to the optimal solution of FCHT, observe that the product of the flow w_{rs} times the distance of the shortest path for any two vertices r and s in the graph G is the smallest possible flow cost for the pair r and s in any FCHT solution and since we are adding up all the flow costs the result follows. □

Note that in the SPR we do not consider the structure of the solution network (a cycle) and we basically end up with an all pair shortest path problem, which can be efficiently solved in $O(n^3)$ (see, Floyd, 1962; Warshall, 1962). For instance, in Example 3.1 we have that the solution for the SPR corresponds to:

$$\sum_{r,s\in V} w_{rs}d^*_{rs} = 10*10 + 10*20 + 100*10 + 100*10 + 200*5 + 10*10 = 3,400.$$

The optimal solution value of this instance is 3,500 and thus the combinatorial bound turns out to be quite close to the optimal.

5.2 Approximate Methods

In this section we present two different heuristic methods to obtain feasible solutions of the FCHT and thus, upper bounds on the optimal solution value. In this case, we cannot guarantee the optimality of the solution obtained. The first heuristic consists of a greedy deterministic constructive procedure to obtain an initial feasible solution, which is later improved with a simple local search procedure. The second heuristic is a greedy randomized adaptive search procedure (GRASP), a multi-start procedure that uses a greedy randomized constructive phase to obtain an initial feasible solution that is later improved with a local search.

5.2.1 Greedy Deterministic Heuristic

In order to obtain an initial feasible solution, we use a greedy deterministic iterative procedure in which a Hamiltonian cycle is constructed by adding one edge at every iteration. It starts from an empty solution and selects one vertex at each iteration. The first vertex added is vertex 1 and the corresponding vertex j to the cheapest edge with cost c_{1j}. The next iterations consist of selecting the vertex associated to the edge with the lowest cost c_{ij} adjacent to at least one of the two current vertices with degree one in the partial solution. The procedure continues until selecting n edges. Observe that in this way, we make sure that the set of edges will define a Hamiltonian circuit. To evaluate the objective function, we still need to compute the shortest path between all pairs of vertices. This can be done efficiently in $O(n^3)$ time by using the Floyd-Warshall algorithm (see Floyd, 1962; Warshall, 1962).

The initial solution is then used as the starting point for a local improvement procedure. We use one of the most common local search techniques for cycle topologies, the 2-opt algorithm (Croes, 1958; Wolsey, 1998). It consists of iteratively removing

two non-adjacent edges and replacing them with two different edges that will create a new cycle (see Figure 5.1). If a new improved solution is found, the algorithm moves to that solution and continues the edge-interchange movement until the method finishes to explore all possible changes and there is no further improvement.

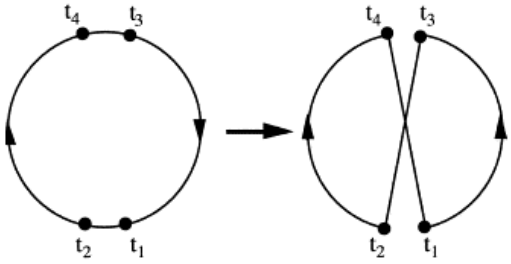


Figure 5.1: 2-Opt algorithm

The general description of the greedy deterministic constructive phase is shown in Algorithm 1.

Algorithm 1: Greedy Deterministic Constructive Procedure

$H = \{1\}$.
$k_1 = \arg \min_{j \in V \ j \neq 1} c_{1j}$.
$e_1 = 1 \quad e_2 = k_1$.
while $(|H| < n)$ **do**
  $k_1 = \arg \min_{j \in V \ j \notin H} c_{e_1 j}$ and $k_2 = \arg \min_{j \in V \ j \notin H} c_{e_2 j}$.
  if$(c_{e_1 k_1} < c_{e_2 k_2})$
    $H \leftarrow H \cup \{k_1\}$
    $e_1 = k_1$.
  else
    $H \leftarrow H \cup \{k_2\}$
    $e_2 = k_2$.
  end if
end do

5.2.2 GRASP

The GRASP meta-heuristic is an iterative procedure consisting of two phases: the first one is a greedy randomized procedure to construct an initial feasible solution while the second is a local search procedure used to improve the initial solution.

At each iteration of the construction phase an edge is randomly selected from a subset of edges, called the restricted candidate list (RCL), adjacent to vertices with degree equal to one. The elements in the RCL are defined as the edges with cost c_{ij} less than or equal to a threshold value

$$\bar{c} = \alpha \left(\sum_{j \in V \setminus H} \left(c_{e_1 j} + c_{e_2 j} \right) / (n - |H|) \right),$$

where e_1 and e_2 are the only two vertices with degree one at each iteration and α is a parameter used to control the size of the RLC. The choice of the next edge to enter the solution is determined by randomly selecting one element from the RCL. The construction phase terminates when n edges are selected.

The randomized greedy algorithm is depicted in Algorithm 2.

Algorithm 2: Constructive phase of GRASP

$r \leftarrow$ random number between 1 and n.
$H = \{r\}$.
$\bar{c} = \sum_{j \in V \;\; j \neq 1} \frac{c_{rj}}{n-1}$.
$RCL = \{i \in V : c_{ri} \leq \bar{c} \text{ and } \; i \neq r\}$
Select a random element k from RCL.
$e_1 = r \quad e_2 = k$.
while $(|H| < n)$ **do**

$\bar{c} = \alpha \sum_{j \in V \setminus H} (c_{e_1 j} + c_{e_2 j}) / (n - |H|)$.

$RCL = \{i \in V \setminus H : c_{e_1 i} \leq \bar{c} \text{ or } \; c_{e_2 i} \leq \bar{c}\} \cup \arg \min_{i \in V \setminus H} \{c_{e_1 i}, c_{e_2 i}\}$

Select a random element k from RCL.
$H \leftarrow H \cup \{k\}$
if$(c_{e_1 k} < c_{e_2 k})$
$e_1 = k$.
else
$e_2 = k$.
end if
end do

The local search phase of the GRASP is also the 2-opt algorithm applied in the same way as in the previous heuristic. Given that GRASP is a multi-start algorithm, it is applied a number of iterations and due to the randomized nature of the constructive phase, it will usually generate different initial feasible solutions. The algorithm also uses different values of $\alpha \in \{0.1, 0.2, \cdots, 1.0\}$, and it intensifies the search with more iterations with the two values of α in which the best solutions are obtained. Notice that when $\alpha = 0$ we are in the case of the deterministic greedy heuristic described in the previous section. The use of different α values is a diversification scheme that permits the method to sample more areas of the solution space, whereas the use of the two most promising α values is an intensification scheme that focuses on particular

areas of the solution space.

5.3 An Exact Solution Method

In this section we present an exact branch-and-cut method to solve the FCHT. It is based on the flow-based formulation introduced in Section 4.3 plus some families of valid inequalities that are able to improve the quality of the LP relaxation bound. We first introduce the inequalities and show their validity. We then define the separation problem and some solution methods associated with these inequalities. Finally, we describe the branch-and-cut method.

5.3.1 Valid Inequalities

We present several families of valid inequalities for the FBF formulation of the FCHT. The first three families are an extension of the *dicut* and *mixed dicut* inequalities of the *uncapacitated fixed charge network flow problem* (see Rardin and Wolsey, 1993; Ortega and Wolsey, 2003). Consider the following dicut inequality. Recall that when we say it is valid for FBF we mean it is valid for the polyhedral set of (mixed 0-1) feasible solutions of the FBF. Let $\delta^-(S) = \{(i,j) \in A : j \in S, i \notin S\}$, where A denotes the set of directed arcs associated with the set of edges E and S is a subset of V.

Proposition 5.2. *Let $k, j \in V$ and $F \subseteq \delta^-(\{j\})$ then*

$$\sum_{(i,j)\in\delta^-(\{j\})\setminus F} x_{kij} + w_{kj} \sum_{(i,j)\in F} y_{ij} \geq w_{kj} \tag{5.1}$$

is valid for FBF.

Proof. Let $k, j \in V$ and $F \subseteq \delta^-(\{j\})$. We know that (4.13) is valid for FBF. Then

$$\begin{aligned} \sum_{i \in V} x_{kij} - \sum_{i \in V} x_{kji} &= w_{kj}, \\ \sum_{i \in V} x_{kij} &\geq w_{kj}, \\ \sum_{(i,j)\delta^-(\{j\})\setminus F} x_{kij} + \sum_{(i,j) \in F} x_{kij} &\geq w_{kj}. \end{aligned}$$

Now, given that in Hamiltonian cycles the degree of each vertex is equal to two, only one of the two summations in the left hand side of the last inequality can be different from zero because the flow will enter vertex j from exactly one edge. When the second term in the left hand side is zero, the first one must be greater than zero and satisfy the inequality. In terms of the flows, this means that the flow entering the destination vertex j through the arcs outside F must be greater than or equal to w_{kj}. On the other hand, if the second summation is positive, the first one will be zero. Then $\sum_{(i,j)\in F} x_{kij} \geq w_{kj}$ can be replaced by $w_{kj}\left(\sum_{(i,j)\in F} y_{ij}\right) \geq w_{kj}$ because we are certain that $\left(\sum_{(i,j)\in F} y_{ij}\right) \neq 0$ since at least one y_{ij} with $(i,j) \in F$ must be set to one and the result follows. □

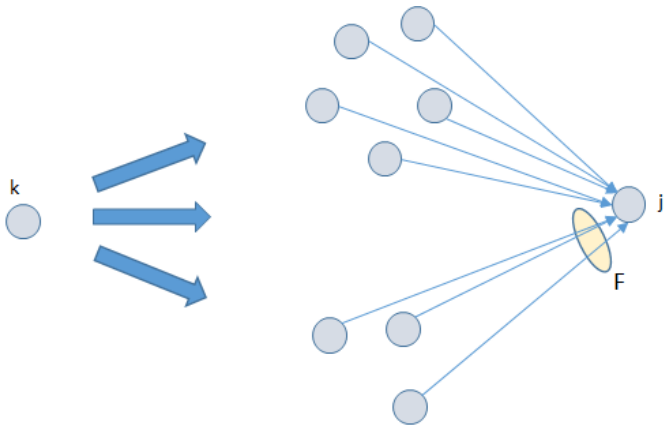


Figure 5.2: Graphical interpretation

A graphical interpretation of the dicut inequalities is given in Figure 5.2. It is important to note that although all feasible integer solutions of FBF satisfy constraints (5.1), fractional solutions may violate them. That is, there might be more than two arcs y_{ij} entering vertex j with a strictly positive value.

We now generalize the previous dicut inequalities by considering more than one destination vertex. These inequalities are known as mixed dicut inequalities.

Proposition 5.3. *Let $k \in V$, $S \subseteq V \backslash \{k\}$ and $F \subseteq \delta^-(S)$ the mixed dicut inequality*

$$\sum_{(i,j) \in \delta^-(S) \backslash F} x_{kij} + \left(\sum_{m \in S} w_{km} \right) \sum_{(i,j) \in F} y_{ij} \geq \sum_{m \in S} w_{km} \tag{5.2}$$

is valid for FBF.

Proof. Once more we start from the fact that (4.13) is valid for FBF, hence for all $k, j \in V$, $S \subseteq V \backslash \{k\}$ and for all $F \subseteq \delta^-(S)$ we have:

$$\sum_{i \in V} x_{kij} - \sum_{i \in V} x_{kji} = w_{kj} \qquad \forall j \in V$$

Then, summing over all $j \in S$ we get

$$\sum_{j \in S} \sum_{i \in V} x_{kij} - \sum_{j \in S} \sum_{i \in V} x_{kji} = \sum_{m \in S} w_{km}$$

$$\sum_{j \in S} \sum_{i \in V \setminus S} x_{kij} - \sum_{j \in S} \sum_{i \in V \setminus S} x_{kji} = \sum_{m \in S} w_{km}$$

$$\sum_{(i,j) \in \delta^-(S)} x_{kij} - \sum_{(i,j) \in \delta^-(S)} x_{kji} = \sum_{m \in S} w_{km}$$

$$\sum_{(i,j) \in \delta^-(S)} x_{kij} \geq \sum_{m \in S} w_{km}$$

$$\sum_{(i,j) \in \delta^-(S) \setminus F} x_{kij} + \sum_{(i,j) \in F} x_{kij} \geq \sum_{m \in S} w_{km}$$

$$\sum_{(i,j) \in \delta^-(S) \setminus F} x_{kij} + \left(\sum_{m \in S} w_{km} \right) \left(\sum_{(i,j) \in F} y_{ij} \right) \geq \sum_{m \in S} w_{km}$$

The last step can be justified in a similar manner as with the previous proposition. In this case, if at least one of the y variables in the set F is set to one then the inequality is satisfied. In case all the y variables in F are zero all the flow must enter through the x variables in the complement of F. □

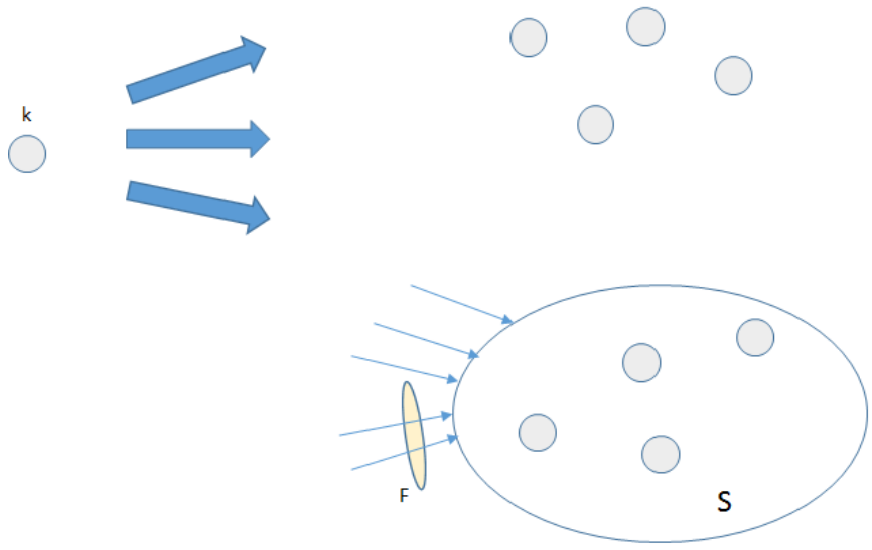


Figure 5.3: Graphical interpretation mixed dicuts

A graphical interpretation of the mixed dicut inequalities is given in Figure 5.3. A more general set of valid inequalities which represents an extension of the mixed dicut with outflow inequalities from Ortega and Wolsey (2003) is the following.

Proposition 5.4. *For $k \in V$, $S \subseteq V\backslash\{k\}$, $F \subseteq \delta^-(S)$ and $C \subseteq \delta^+(S)$, the mixed dicut with outflow inequality*

$$\sum_{(i,j)\in\delta^-(S)\backslash F} x_{kij} + \left(\sum_{m\in S} w_{km}\right)\left(\sum_{(i,j)\in F} y_{ij}\right) \geq$$

$$\sum_{m\in S} w_{km} + \sum_{(i,j)\in C} x_{kij} - \left(\sum_{m\in V} w_{km} - \sum_{m\in S} w_{km}\right)\left(\sum_{(i,j)\in C} y_{ij}\right)$$

is valid for FBF, with $\delta^+(S) = \{(i,j) \in A : i \in S, j \notin S\}$.

Proof. We know (4.13) is valid for FBF. Then for all $k, j \in V$, $S \subseteq V\backslash\{k\}$, $F \subseteq \delta^-(S)$

and for all $C \subseteq \delta^+(S)$ we have:

$$\sum_{i \in V} x_{kij} - \sum_{i \in V} x_{kji} = w_{kj}$$

Summing over all $j \in S$ we obtain

$$\begin{aligned}
\sum_{j \in S} \sum_{i \in V \setminus S} x_{kij} - \sum_{j \in S} \sum_{i \in V \setminus S} x_{kji} &= \sum_{m \in S} w_{km} \\
\sum_{(i,j) \in \delta^-(S)} x_{kij} - \sum_{(i,j) \in \delta^-(S)} x_{kji} &= \sum_{m \in S} w_{km} \\
\sum_{(i,j) \in \delta^-(S)} x_{kij} - \sum_{(i,j) \in \delta^+(S)} x_{kij} &= \sum_{m \in S} w_{km} \\
\sum_{(i,j) \in \delta^-(S) \setminus F} x_{kij} + \sum_{(i,j) \in F} x_{kij} &= \sum_{m \in S} w_{km} + \sum_{(i,j) \in \delta^+(S)} x_{kij}
\end{aligned}$$

Since $\forall k, i, j \in V$ we have that $x_{kij} \leq M_k y_{ij}$, then $\sum_{(i,j) \in F} x_{kij} \leq M_k \sum_{(i,j) \in F} y_{ij}$ and thus

$$\begin{aligned}
\sum_{(i,j) \in \delta^-(S) \setminus F} x_{kij} + M_k \sum_{(i,j) \in F} y_{ij} &\geq \sum_{m \in S} w_{km} + \sum_{(i,j) \in \delta^+(S)} x_{kij} \\
\sum_{(i,j) \in \delta^-(S) \setminus F} x_{kij} + M_k \sum_{(i,j) \in F} y_{ij} &\geq \sum_{m \in S} w_{km} + \sum_{(i,j) \in C} x_{kij}
\end{aligned}$$

Considering $\bar{x}_{kij} = M_k y_{ij} - x_{kij}$ and $\bar{y}_{ij} = 1 - y_{ij}$, and substituting for the variables in the summation over C we obtain

$$\sum_{(i,j) \in \delta^-(S) \setminus F} x_{kij} + M_k \sum_{(i,j) \in F} y_{ij} \geq \sum_{m \in S} w_{km} + |C| M_k - M_k \sum_{(i,j) \in C} \bar{y}_{ij} - \sum_{(i,j) \in C} \bar{x}_{kij}.$$

Applying the mixed integer rounding procedure (MIR) (Nemhauser and Wolsey, 1988)

we have

$$\sum_{(i,j)\in\delta^-(S)\setminus F} x_{kij} + \sum_{(i,j)\in C} \bar{x}_{kij} \geq \left(\sum_{m\in S} w_{km}\right)\left(1 + |C| - \sum_{(i,j)\in C} \bar{y}_{ij} - \sum_{(i,j)\in F} y_{ij}\right).$$

Finally, after substitution back we get

$$\sum_{(i,j)\in\delta^-(S)\setminus F} x_{kij} + \left(\sum_{m\in S} w_{km}\right)\left(\sum_{(i,j)\in F} y_{ij}\right) \geq \sum_{m\in S} w_{km} + \sum_{(i,j)\in C} x_{kij} - \left(\sum_{m\in V\setminus S} w_{km}\right)\left(\sum_{(i,j)\in C} y_{ij}\right).$$

□

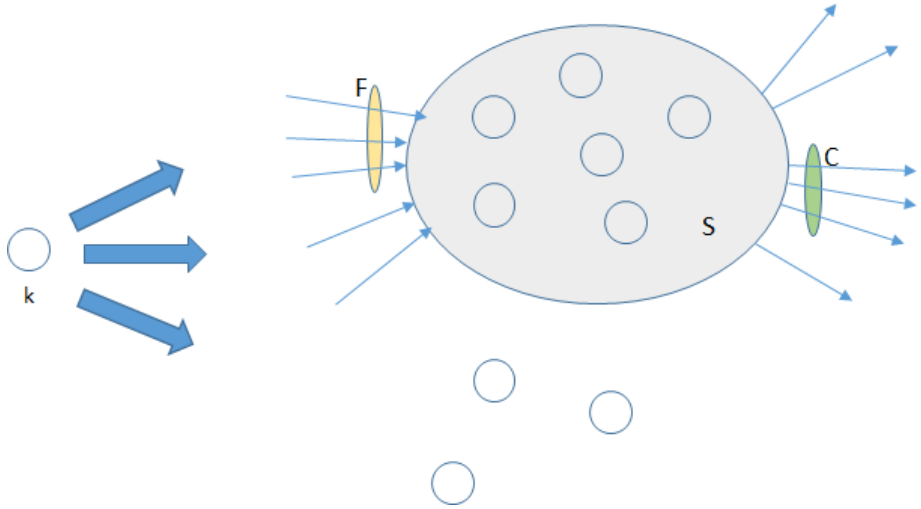


Figure 5.4: Graphical interpretation of mixed dicuts with outflow.

A graphical interpretation of the mixed dicut with outflow inequalities is given in Figure 5.4.

Finally, we present two more sets of valid inequalities where the cycle structure of the solution network is exploited. We present them in the following two propositions.

Proposition 5.5. *For all $k, i, j \in V$ with $k \neq i, j$ and $i < j$, the inequality*

$$x_{kij} \leq (M_k - w_{ki})y_{ij} \tag{5.3}$$

is valid for FBF.

Proof. Recall that $M_k = \sum_{s \in V} w_{ks}$. Also, we know that (4.14) is valid for FBF. Therefore, for all $k, i, j \in V$ with $k \neq i, j$ and $i < j$ we have:

$$\begin{aligned} x_{kij} + x_{kji} &\leq M_k y_{ij} \\ x_{kij} &\leq M_k y_{ij} \\ x_{kij} &\leq (M_k - w_{ki}) y_{ij} \end{aligned}$$

Since, the flow sent to vertex i with origin k will not be part of the flow going through arc (i, j) with origin k. □

Proposition 5.6. *For all $k, i, j \in V$ with $k \neq i, j$ and $i < j$, the inequality*

$$x_{kij} \leq (M_k - w_{ki}) y_{ij} - \sum_{q_1 \in V : q_1 \neq i,j} w_{kq_1} (y_{q_1 i} + y_{ij} - 1) - \sum_{q_1 \in V} \sum_{q_2 \in V} w_{kq_2} (y_{q_1 i} + y_{q_2 q_1} + y_{ij} - 2) \tag{5.4}$$

is valid for FBF.

Proof. The first term in the right hand side of the inequality correspond to the previous valid inequality. Following the same idea, we can say that in the case where edge $\{i, j\}$ is part of the final network, the flow sent from k going to the unique vertex adjacent to i (call it $\hat{q}$), that is not j, is not going to pass through arc (i, j). This is because if the shortest path to go from k to $\hat{q}$ is not using the edge $\{i, j\}$ then the result is obvious. On the other hand, if the shortest path to go from k to $\hat{q}$ contains the edge $\{i, j\}$ this means the flow will pass through arc (j, i) and not (i, j) which is the one in consideration. □

We can generalize this idea for a larger number of vertices. We would go from vertex i until we reach vertex k. For this particular case we only wrote the inequality of 2 vertices before i.

5.3.2 Separation of Inequalities

As we described in Chapter 2, given a point, a set of feasible solutions and a family of valid inequalities, the problem of finding the most violated inequality (if any) is called the separation problem. Given that the dicut inequalities are a particular case of the mixed dicut inequalities, we directly introduce the separation problem of the former ones. For the particular set of inequalities (5.1), given an LP solution $(\hat{x}, \hat{y})$ for every $k, j \in V$, we need to find a set $F \subseteq \delta^-(\{j\})$ such that the inequalities are violated, i.e.,

$$\sum_{(i,j)\in\delta^-(\{j\})\setminus F} \hat{x}_{kij} + w_{kj} \sum_{(i,j)\in F} \hat{y}_{ij} < w_{kj}.$$

In order to find such set F we define the following decision variables and the corresponding separation problem for each pair of vertices $k, j \in V$:

$$z_{ij} = \begin{cases} 1 & \text{if } (i,j) \in F, \\ 0 & \text{otherwise.} \end{cases}$$

$$\xi_{kj} = \min \left\{ \sum_{(i,j)\in\delta^-(\{j\})} \hat{x}_{kij}(1 - z_{ij}) + w_{kj} \sum_{(i,j)\in\delta^-(\{j\})} \hat{y}_{ij} z_{ij} : \; z_{ij} \in \{0,1\} \right\}.$$

There is going to be a violated inequality whenever $\xi_{kj} < 0$. Otherwise, all inequalities will be satisfied at that particular point $(\hat{x}, \hat{y})$. Notice that when the values $w_{kj}\hat{y}_{ij} - \hat{x}_{kij}$ are negative the variable z_{ij} will be set to one in order to minimize the function. Then, we can define the set

$$\hat{F} = \{(i,j) \in \delta^-(\{j\}) : \frac{\hat{x}_{kij}}{\hat{y}_{ij}} > w_{kj}\},$$

Hence, the optimal solution of ξ_{kj} is given by $F = \hat{F}$.

We now generalize this idea for the mixed dicut inequalities presented in (5.2). Recall that in this case we consider a set S of destination vertices and a set F of arcs incident to S. Therefore, the separation problem consists of finding the two sets S and F that minimize the value of the left hand side for a given LP solution $(\hat{x}, \hat{y})$. We define hereafter the value $Q = \sum_{m \in S} w_{km}$ to simplify the notation.

The following proposition indicates how to choose the best F subset for a given value of Q (i.e. for a given set S) and thus, to optimally solve the separation problem.

Proposition 5.7. *Let $k, j \in V$, $Q \geq 0$ and $(\hat{x}, \hat{y})$ be a given LP solution. Then, a set $\hat{F} \subseteq \delta^-(S)$ that minimizes the value of*

$$L(Q) = \min_{F \subseteq \delta^-(S)} \sum_{(i,j) \in \delta^-(S) \setminus F} \hat{x}_{kij} + Q \left(\sum_{(i,j) \in F} \hat{y}_{ij} \right)$$

is given by $\hat{F} = \{(i,j) \in \delta^-(S) : \dfrac{\hat{x}_{kij}}{\hat{y}_{ij}} \geq Q\}$.

Proof. The set $\hat{F}$ can be obtained by solving the following optimization problem

$$\min \left[\sum_{(i,j) \in \delta^-(S)} \hat{x}_{kij}(1 - \gamma_{ij}) + Q * \left(\sum_{(i,j) \in \delta^-(S)} \hat{y}_{ij} \gamma_{ij} \right) \right]$$

with $\gamma_{ij} \in \{0, 1\}$.

Then, the result follows since the objective function can be rewritten as

$$\sum_{(i,j) \in \delta^-(S)} \hat{x}_{kij} + \min \left[\sum_{(i,j) \in \delta^-(S)} \left(Q\hat{y}_{ij} - \hat{x}_{kij} \right) \gamma_{ij} \right].$$

□

Consequently, for a given value of Q, the function $L(Q)$ will be

$$\sum_{(i,j)\in\notin\hat{F}} \hat{x}_{kij} + Q * \left(\sum_{(i,j)\in\hat{F}} \hat{y}_{ij} \right)$$

which is a piecewise linear function. Therefore, we can define intervals $I_t = [A_t, B_t]$ for $t = 1, \cdots, l$ with $A_1 = 0$, $A_t = B_{t-1}$ and $B_l = \infty$ and a series of sets $\hat{F}^t$ such that for all $Q \in I_t$, $L(Q) = a^t + Qb^t$ with $a^t = \sum_{(i,j)\notin\hat{F}^t} \hat{x}_{kij}$ and $b^t = \sum_{(i,j)\in\hat{F}^t} \hat{y}_{ij}$.

In order to solve the separation problem we know that the possible values of Q are given by $Q = \sum_{m\in V} w_{km}\beta_m$ with $\beta_m \in \{0, 1\}$. Then, we can present the following proposition.

Proposition 5.8. *Let $k \in V$, and $(\hat{x}, \hat{y})$ be given. There exists a violated mixed dicut inequality by $(\hat{x}, \hat{y})$ if and only if $\eta < 0$, where*

$$\eta = \min L\left(\sum_{m\in V} w_{km}\beta_m \right) - \sum_{m\in V} w_{km}\beta_m$$
$$\beta_m \in \{0, 1\}.$$

Since the function $L(Q)$ is piecewise linear, the value of η depends on the interval I_t, where the value $\sum_{m\in V} w_{km}\beta_m$ lies. For the purpose of solving the separation problem we consider the value of $B_l = \sum_{m\in V} w_{km}$ which is the highest value it can take and we proceed as follows:

1. Define the intervals I_t that determine the expression of the piecewise linear function L as well as $\hat{F}^t$ which denote the optimal solution for a given Q in I_t.

2. For $t = 1, \cdots, l$ solve the optimization problem

$$\eta^t = \min \left[a^t + \left(\sum_{m \in V} w_{km} \beta_m \right) b^t \right] - \sum_{m \in V} w_{rm} \beta_m$$
$$A_t \leq \sum_{m \in V} w_{km} \beta_m \leq B_t$$
$$\beta_m \in \{0, 1\} \qquad \forall m \in V$$

3. If $\min\{\eta^t : 1, \cdots, l\} \geq 0$ then no inequality is violated by $(\hat{x}, \hat{y})$.

From the above analysis, the separation problem considers the solution of a number of independent knapsack problems with lower bounds which are known to be *NP*-hard. From a practical point of view it is also important to have efficient solution methods able to identify violated inequalities without optimally solving the separation problem. Hence, we develop a simple heuristic algorithm for approximately solving the separation problem. The main idea of this algorithm is to iteratively construct a set S and by checking the values $\hat{x}_{kij}, \hat{y}_{ij}$ we can obtain the best possible F for a given set S using the procedure previously presented. The algorithm does not guaranty optimality because it does not check all possible sets S.

For each vertex $k \in V$, we initially consider all the other vertices of V to be in S and we obtain the optimal set F associated with $Q = \sum_{m \in S} w_{km}$. Let

$$\Delta(S) = \sum_{(i,j) \in \delta^-(S) \setminus F} \hat{x}_{kij} + \left(\sum_{m \in S} w_{km} \right) \sum_{(i,j) \in F} \hat{y}_{ij} - \sum_{m \in S} w_{km}.$$

We then arbitrarily remove one element of S at a time if $\Delta(S)$ strictly decreases. After this process is completed, we add the corresponding inequality if violated, i.e., $\Delta(S) < 0$.

The separation problem for the mixed dicut inequalities with outflow is very similar to the one of the mixed dicuts. Analogously, in order to determine the best set

$C \subseteq \delta^+(S)$ for a given S, we simply add $(i,j) \in \delta^+(S)$ to C if and only if

$$\hat{x}_{kij} - \left(\sum_{m \in V \setminus S} w_{km} \right) \hat{y}_{ij} > 0.$$

5.3.3 A Branch-and-Cut Algorithm

We next introduce an exact solution method for the FCHT based on the flow-based formulation FBF presented in Chapter 4. It is a branch and cut method that uses the valid inequalities and their corresponding separation problems introduced in the previous sections to tighten the LP bound at some nodes of the enumeration tree in the branch-and-bound process.

The idea is to first solve the LP relaxation of (4.13)–(4.17) at the root node of the tree. We then iteratively add valid inequalities that are violated by the current LP solution. When no more violated inequalities exist, we resort to CPLEX for solving the resulting formulation by enumeration, using a call-back function for generating additional violated constraints at some nodes of the enumeration tree. In the computational experiments section, we provide additional details on which subset of inequalities are added and their impact in terms of improving the LP bound and the CPU time.

We use the best feasible solution obtained with the heuristics presented in Section 5.2 as an initial upper bound for the branch-and-cut. The heuristic introduced in section 5.3.1 for the separation problem of the mixed dicut inequalities is used to quickly find violated inequalities. We also added a tolerance parameter in order to determine the minimum absolute violation an inequality should have to add to the current LP relaxation. If the tolerance is set to zero, then all violated valid inequalities would be added.

Chapter 6

Computational Experiments

A computational study has been carried out in order to test the performance of the models and solution methodologies introduced in previous sections. In the first part of the computational experiments, we focus on a comparison of the three most promising MIP formulations of Chapter 3. In the second part we analyze the performance of the different heuristic methods presented in Section 5.2. In the final part, we present the results of the branch-and cut method and the effect of adding different families of valid inequalities. The algorithms were coded in C and run on Windows with a Pentium Dual-Core processor at 2.80 GHz and 4GB of RAM. The formulations and the branch-and-cut method were implemented using the callable library of CPLEX 12.5.

We have performed the computational experiments using well-known instances for the closely related optimum communication spanning tree problem and hub location problems as well as some randomly generated instances. The instances we use are the following:

- Three instances from Palmer (1994) with to 6, 12 and 24 vertices. The vertices correspond to cities in the United States of America and the demands are

inversely proportional to their costs (distances).

- Two instances attributed to Raidl and found in Rothlauf (2009) with 10 and 20 vertices respectively. The distances and demands are uniformly distributed in [0, 100].

- Three instances from Berry et al. (1995). One with 6 vertices and the other two with 35 vertices.

- Three instances generated by us. One with 4 vertices, from Example 3.1 and two random instances generated with 15 and 20 vertices.

- Four instances from the well-known OR-Library (http://people.brunel.ac.uk/ mastjjb/jeb/info.html) called AP (Australian Post) which contain information on the 200 most important cities in Australia and their corresponding postal flows.

We set the CPU time limit to 86,400 seconds (one day). Whenever CPLEX is not able to solve an instance within the time limit, we write TIME in the corresponding entry of the table. If the algorithm runs out of memory we write MEM.

6.1 MIP Formulations and Combinatorial Bounds

In this section we summarize the results obtained when implementing in CPLEX the path-based formulation PBF, the flow-based formulation FBF, and the second two-index formulation $2IF2$. The second path-based formulation $PBF2$ and the first two-index formulation $2IF$ cannot be directly solved by a general purpose solver (such as CPLEX) because they require the development of ad hoc column generation and branch-and-cut methods, respectively. Given that we are already testing other formulations that obtain similar bounds as well as the limited scope of this thesis, they were not considered in these computational experiments. We also show the

results from the combinatorial bounds to assess their quality. The detailed results of the comparison are provided in Table 6.1. The first column contains the name of the instance. The next three columns correspond to the CPU time in seconds needed to obtain an optimal solution and the duality gap relative to the LP bound obtained with the PBF, FBF and 2IF2, respectively. The optimality gap is computed as $LPGap = 100(OPT - LB_F)/OPT$, where OPT is the optimal value and LB_F is the lower bound obtained with PBF, FBF, and $2IF2$, respectively. The fifth column provides the optimal value of the considered instances or the best upper bound obtained with any method (indicated with an $*$ for the cases when the optimal is not obtained). The last column shows the optimality gap associated with the combinatorial bounds and the best known upper bound.

	PBF		FBF		2IF2		optimal value	Comb. bound
Instance	time(s)	LP Gap(%)	time(s)	LP Gap (%)	time(s)	LP Gap (%)		(%)
4 Vertices	0.56	0.00	0.59	2.14	1.01	100.00	3500	2.85
Palmer6	1.96	0.00	1.92	19.12	4.58	100.00	679976	20.25
Palmer12	85.00	10.66	25.00	36.10	TIME	100.00	3631576	37.46
Palmer24	TIME	4.23	101.00	12.54	MEM	100.00	1442336	24.65
Radil10	8.00	22.87	3.28	43.47	TIME	100.00	91567	46.79
Radil20	TIME	57.97	MEM	68.97	MEM	100.00	456571(*)	70.52
Berry6	3.60	7.26	2.14	15.44	6.43	100.00	608	24.17
Berry35	MEM	MEM	MEM	65.44	MEM	100.00	51227(*)	66.98
Berry35u	MEM	MEM	MEM	77.54	MEM	100.00	54412(*)	78.69
Random15	2612.00	42.04	1040.00	56.45	TIME	100.00	24536	56.89
Random20	TIME	47.60	TIME	60.87	TIME	100.00	50855(*)	61.16
10ll	4.39	10.75	3.12	30.24	TIME	100.00	72550	31.00
20ll	TIME	42.69	TIME	54.32	TIME	100.00	117955(*)	54.87
25ll	TIME	50.97	TIME	60.56	TIME	100.00	133605(*)	60.87
40ll	MEM	MEM	MEM	68.30	MEM	100.00	154621(*)	68.45

Table 6.1: Comparison of the three MIP formulations.
(*)Best upper bound known.

Table 6.1 shows that in terms of CPU time the best formulation is the FBF, as in terms of the LP relaxation we are observing the theoretical result presented in the previous chapter, where the PBF is able to obtain the tightest LP bounds. We can see that the 2IF2 turned out to be the worst of the three formulations in terms

of CPU time and LP relaxation, even though it has less variables and the number of constraints is polynomially bounded. These results can be partially explained by the fact that 2IF2 uses a large number of constraints with a big M coefficient. It is common in MIP that such type of formulations usually provide very weak lower bounds associated with their LP relaxations.

For many instances we were not able to find the optimal solution (*), either because we run out of memory or because we reached the time limit. When the latter happened, the optimality gap in CPLEX remained at least above 20%. In terms of the combinatorial bound from Section 5.1, we can see that although this bound represents a good improvement to the LP bound of $2IF$, in most instances, the LP relaxation of the PBF and the FBF provides better lower bounds. However, it is important to mention that the combinatorial bound requires a very small amount of memory and can be obtained much more efficiently than the LP relaxation of any formulation.

6.2 Heuristics

In this section we present the computational results from the implementation of the greedy deterministic heuristic and the GRASP metaheuristic. We present the optimal solution obtained with CPLEX and for those instances not solved to optimality yet, we show the best upper bound found so far (indicated with *). We also show the upper bound given with the TSP solution. That is, we evaluate the flow cost objective of the FCHT on the optimal solution of the TSP given by the minimization of the design cost objective.

Table 6.2 compares the optimal value obtained by CPLEX (or the best upper bound known) with the corresponding values obtained by the approximation methods.

In the case of the GRASP algorithm, we used $1,000$ iterations for each instance and we record the best solution obtained, whereas in the other deterministic methods they are executed only one time. The CPU time of both the greedy and the GRASP algorithms did not exceed 3 minutes for any instance, whereas for the smaller instances (i.e. less than 15 vertices) the CPU time was less than 10 seconds. Therefore, the results associated with the running times are omitted from the table.

Instance	optimal	TSP solution	greedy heuristic	GRASP heuristic
4 vertices	3500	45.31	20.40	0.00
Palmer6	679976	6.27	0.00	0.00
Palmer12	3631576	3.00	0.00	0.00
Palmer24	1442336	46.37	3.76	3.17
Radil10	91567	18.61	18.61	0.00
Radil20(*)	456571	0.00	21.32	4.15
Berry6	608	0.00	8.01	0.00
Berry35(*)	34593	-	32.47	0.00
Berry35u(*)	54412	0.00	37.41	9.30
Random15	24536	11.36	14.62	0.00
Random20(*)	50855	0.00	39.45	1.07
10ll	72550	0.00	10.72	0.00
20ll(*)	117955	6.05	6.15	0.00
25ll(*)	133605	1.72	8.94	0.00
40ll(*)	154621	6.47	7.48	0.00

Table 6.2: Optimality Gap (%) for FCHT.

The results presented in Table 6.2 show that the GRASP method dominates the others with the exception of instances Radil20 and Berry35u. An interesting observation is that, even though the TSP solution is far from optimal on several instances, it provides the best known bound for four instances.

6.3 Branch-and-Cut

We next present the results obtained with the branch-and-cut method using various sets of valid inequalities. Table 6.3 is a comparison between the basic formulation

and the formulation strengthened with the mixed dicut inequalities only at the root node of the enumeration tree as well as with two other branch-and-cut methods. The first one is the branch-and-cut of CPLEX, which tries to incorporate general valid inequalities such as flow cover and lifted cover inequalities, MIR inequalities, and Gomory cuts, among others. The second one is a combination of the previous one with the addition of the mixed dicut inequalities.

	No cuts			Mixed dicuts				CPLEX cuts			CPLEX cuts + Mixed dicuts			
Instance	LP Gap	time	BB nodes	LP Gap	time	BB nodes	mixed cuts	LP Gap	time	BB nodes	LP Gap	time	BB nodes	mixed cuts
4 vertices	2.14	0.05	0	0.00	0.02	0	2	0.00	0.03	0	0.00	0.02	0	2
Palmer6	19.12	0.03	16	8.90	0.05	7	21	0.68	0.08	0	0.01	0.08	0	8
Palmer12	36.10	15.48	8148	24.70	93.35	9961	281	12.62	52.48	1574	12.00	51.81	1385	39
Palmer24	12.54	125.00	14985	8.15	1855.00	25358	3163	12.25	191.00	16152	8.15	778.00	11013	2274
Radil10	43.47	0.35	410	39.81	1.41	379	196	24.80	1.55	161	24.56	1.59	145	74
Radil20(*)	68.97	TIME	TIME	67.26	TIME	TIME	TIME	60.78	TIME	TIME	60.59	TIME	TIME	TIME
Berry6	15.44	0.22	14	12.44	0.07	17	16	7.15	0.12	8	7.15	0.07	7	7
Berry35(*)	65.44	MEM	MEM	65.12	MEM	MEM	MEM	55.41	MEM	MEM	55.36	MEM	MEM	MEM
Berry35u(*)	77.54	MEM	MEM	77.40	MEM	MEM	MEM	72.02	MEM	MEM	71.57	MEM	MEM	MEM
random15	56.45	588.00	179703	54.72	4704.00	180315	2179	43.61	956.00	27248	43.55	1583.00	36667	252
random20(*)	60.87	TIME	TIME	59.45	TIME	TIME	TIME	50.15	TIME	TIME	50.15	TIME	TIME	TIME
10ll	30.24	3.91	1345	20.87	12.86	534	339	11.02	0.89	69	11.12	1.08	82	53
20ll(*)	54.32	TIME	TIME	49.65	TIME	TIME	TIME	44.34	TIME	TIME	44.23	TIME	TIME	TIME
25ll(*)	60.56	TIME	TIME	56.95	TIME	TIME	TIME	52.84	TIME	TIME	52.82	TIME	TIME	TIME
40ll(*)	68.30	MEM	MEM	65.98	MEM	MEM	MEM	62.03	MEM	MEM	61.32	MEM	MEM	MEM

Table 6.3: Branch-and-cut

Table 6.3 shows that the implementation of the branch-and-cut method using the valid inequalities from proposition 5.3, provides improved LP relaxation bounds at the root node of the enumeration tree. However, a somehow unexpected result is that this improvement seems not to have a positive impact on the CPU time. The results show that the LP value improves for all three approaches comparing with no cuts added. Moreover, the best LP gap in all instances is the one when we apply CPLEX's cuts and the mixed dicuts inequalities at the same time, however the CPU time increases for most instances.

We also implemented the mixed dicuts with outflow from proposition 5.4 and the lower bound in most instances increased compared with the mixed dicuts alone. However, the improvement was small (from 0.1% to 0.9%) and there was no significant change in the time spent to solve the instances. Whereas for inequalities (5.3) and (5.4), we executed the FBF exchanging them with (4.14), showing that for the instances compiled the results did not improve significantly. However, inequalities (5.3) might always be used instead of (4.14) since the number of them does not change.

Chapter 7

Conclusions and Further Research

In this thesis we introduced the *minimum flow cost Hamiltonian tour problem*. It is a combinatorial optimization problem that, to the best of our knowledge, has not been addressed in the literature before. One of the main contributions of this work was to study the polyhedron associated with the set of feasible solutions of the FCHT. We proved that the FCHT belongs to the class of *NP*-hard problems and proposed five different MIP formulations. These formulations were theoretically compared with respect to the quality of their LP relaxation bounds as well as computationally using commercial solvers such as CPLEX 12.5. The results showed that, in terms of the LP bounds, the two path-based formulations are the most promising ones while the FBF seemed to be more efficient when proving optimality with a general purpose solver.

Another important contribution of this thesis was the development of approximate and exact methodologies in order to solve the FCHT or to find good feasible solutions. In particular, we provided combinatorial bounds, two heuristics, a deterministic greedy heuristic and a GRASP meta-heuristic. The combinatorial bounds as well as the heuristic methods were able to efficiently obtain reasonable lower and upper bounds, respectively, within a few minutes for the worst case of the consid-

ered instances. A branch-and-cut algorithm based on the FBF was also presented. We proved the validity of various extended mixed dicut inequalities as well as other classes of inequalities for the FCHT. We also introduced their separation problems and algorithms to solve them. The results obtained in the computational experiments showed that the lower bounds at the root node of the enumeration tree were considerably improved when using the mixed dicut inequalities. Unfortunately, they were not very efficient in reducing the overall CPU time required to prove optimality.

The FCHT turned out to be a very difficult optimization problem. Instances with up to 24 vertices were optimally solved in one day of CPU time. However, formulations and solution methods failed in proving the optimality of the solutions when dealing with some 20 vertices and larger instances. This is particularly interesting when comparing the difficulty with the closely related TSP problem, for which instances with hundreds of vertices can be optimally solved with CPLEX using standard MIP formulations.

There are several aspects of this research topic that are worth further research. It is of theoretical interest to find special cases for which the problem is polynomially solvable. Moreover, we strongly believe that the performance of the branch-and-cut algorithm can be improved by using other families of valid inequalities and lifting procedures. Further exploiting the cycle structure of the network might be useful for finding strong valid inequalities.

Adapting well-known decomposition techniques, such as column generation or benders decomposition, might seem an interesting way to use the quality of the LP bounds associated with the path-based formulations. These methods can exploit the structure of the network in order to efficiently solve larger instances. In terms of approximation techniques, it is clear that more sophisticated meta-heuristic procedures may be developed to obtain and guarantee high quality solutions for the FCHT.

Chapter 8

Bibliography

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs, New Jersey.

Ahuja, R. K. and Murty, V. V. S. (1987). Exact and heuristic algorithms for the optimum communication spanning tree problem. *Transportation Science*, 21(3):163–170.

Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J. (2006). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Asef-Vaziri, A. and Goetschalckx, M. (2008). Dual track and segmented single track bidirectional loop guidepath layout for agv systems. *European Journal of Operational Research*, 186(3):972–989.

Asef-Vaziri, A., Laporte, G., and Ortiz, R. (2007). Exact and heuristic procedures for the material handling circular flow path design problem. *European Journal of Operational Research*, 176(2):707 – 726.

Bazaraa, M. S., Jarvis, J. J., and Sherali, H. D. (2009). *Linear programming and network flows*. Wiley, Hoboken, New Jersey.

Bellman, R. (1958). On a Routing Problem. *Quarterly of Applied Mathematics*, 16:87–90.

Berry, L., Murtagh, B., and McMahon, G. (1995). Applications of a genetic-based algorithm for optimal design of tree-structured communication networks. *In Proceedings of the Regional Teletraffic Engineering Conference of the International Teletrafficc Congress*, pages 361–370.

Chvátal, V. (1973). Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4):305 – 337.

Contreras, I., Fernández, E., and Marín, A. (2010). Lagrangean bounds for the optimum communication spanning tree problem. *TOP*, 18(1):140–157.

Cook, S. A. (1971). *The complexity of theorem-proving procedures*. STOC '71. Association for Computing Machinery, New York, NY, USA.

Cook, W. J. (2011). *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press.

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., and Schrijver, A. (1998). *Combinatorial optimization*. John Wiley & Sons, inc., Hoboken, New Jersey.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410.

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

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Edmonds, J. (1965). Maximum matching and a polyhedron with $0, 1$ vertices. *Journal of Research of the National Bureau of Standards*, 69 B:125–130.

Festa, P. and Resende, M. (2011). Grasp: basic components and enhancements. *Telecommunication Systems*, 46(3):253–271.

Fischetti, M., Monaci, M., and Salvagnin, D. (2012). Three ideas for the quadratic assignment problem. *Operations Research*, 60(4):954–964.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communication of the Association for Computing Machinery*, 5(6):345–346.

Frangioni, A. and Gendron, B. (2007). 0-1 reformulations of the multicommodity capacitated network design problem. *Discrete Applied Mathematics*, 157(6):1229 – 1241.

Frieze, A. M. and Yadegar, J. (1983). On the quadratic assignment problem. *Discrete Applied Mathematics*, 5(1):89 – 98.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. Freeman, New York, NY, USA.

Garfinkel, R. S. and Gilbert, K. C. (1978). The bottleneck traveling salesman problem: Algorithms and probabilistic analysis. *Journal of Association for Computing Machinery*, 25(3):435–448.

Gendreau, M. and Potvin, J.-Y. (2010). In *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research and Management Science*. Springer, New York, USA.

Gendron, B., Crainic, T., and Frangioni, A. (1999). Multicommodity capacitated network design. In Sansó, B. and Soriano, P., editors, *Telecommunications Network Planning*, pages 1–19. Springer.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275 – 278.

Gutin, G. and Punnen, A. P. (2002). *The Traveling Salesman Problem and Its Variations*. Kluwer, New York, USA.

Hu, T. (1974). Optimum communication spanning trees. *SIAM Journal on Computing*, 3(3):188–195.

Johnson, D. S., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1978). The complexity of the network design problem. *Networks*, 8(4):279–285.

Koopmans, T. C. and Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica*, 25(1):53–76.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416.

Laporte, G., Mercure, H., and Nobert, Y. (1987). Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. *Discrete Applied Mathematics*, 18(2):185–197.

Laporte, G. and Rodríguez-Martín, I. (2007). Locating a cycle in a transportation or a telecommunications network. *Networks*, 50(1):92–108.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B. (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, New York, USA.

Loiola, E. M., Maia-de Abreu, N. M., Boaventura-Netto, P. O., Hahn, P., and Querido, T. (2007). A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657 – 690.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal Association for Computing Machinery*, 7(4):326–329.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and combinatorial optimization*. Wiley, Hoboken, New Jersey.

Nyberg, A. and Westerlund, T. (2012). A new exact discrete linear reformulation of the quadratic assignment problem. *European Journal of Operational Research*, 220(2):314 – 319.

Öncan, T., Altinel, I. K., and Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637 – 654.

Ortega, F. and Wolsey, L. A. (2003). A branch-and-cut algorithm for the single commodity uncapacitated fixed charge network flow problem. *Networks*, 41:143 – 158.

Palmer, C. (1994). An approach to a problem in network desing using genetic algorithms. *PhD thesis, Polytechnic University, Troy, New York*.

Qiu, L., Hsu, W., Huang, S., and Wang, H. (2002). Scheduling and routing algorithms for AGVs: a survey. *International Journal of Production Research*, 40(3):745–760.

Rardin, R. L. and Wolsey, L. A. (1993). Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network flow problems. *European Journal of Operational Research*, 71(1):95 – 109.

Rosen, K. H. (2007). *Discrete Mathematics and its Applications*. McGraw-Hill, New York, 6th edition.

Rothlauf, F. (2009). On optimal solutions for the optimal communication spanning tree problem. *Operations Research*, 57(2):413–425.

Schrijver, A. (2003). *Combinatorial Optimization*. Springer-Verlag, Berlin Heidelberg.

Solomon, M. M. and Desrosiers, J. (1988). Time window constrained routing and scheduling problems. *Transportation Science*, 22(1):1–13.

Tanash, M., Contreras, I., and Vidyarthi, N. (2013). The cycle hubs location problem. *Work in progress*.

Warshall, S. (1962). A theorem on boolean matrices. *J. Association for Computing Machinery*, 9(1):11–12.

Wolsey, L. A. (1998). *Integer programming*. Wiley, Hoboken, New Jersey.

Xu, J., Chiu, S. Y., and Glover, F. (1998). Optimizing a ring-based private line telecommunication network using tabu search. *Management Science*, 45(3):330–345.