

**PROJETO DE REDES DE SENSORES SEM FIO:
FORMULAÇÕES E ALGORITMOS EXATOS**

VINÍCIUS WELLINGTON COELHO DE MORAIS

**PROJETO DE REDES DE SENSORES SEM FIO:
FORMULAÇÕES E ALGORITMOS EXATOS**

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: GERALDO ROBSON MATEUS

Belo Horizonte

Agosto de 2018

VINÍCIUS WELLINGTON COELHO DE MORAIS

**TOPOLOGICAL DESIGN OF WIRELESS SENSOR
NETWORK: FORMULATIONS AND EXACT
ALGORITHMS**

Thesis presented to the Graduate Program
in Computer Science of the Universidade
Federal de Minas Gerais - Departamento
de Ciência da Computação in partial ful-
fillment of the requirements for the degree
of Doctor in Computer Science.

ADVISOR: GERALDO ROBSON MATEUS

Belo Horizonte

August 2018

© 2018, Vinícius Wellington Coelho de Moraes.
Todos os direitos reservados.

Morais, Vinícius Wellington Coelho de

M828t Topological design of wireless sensor network:
formulations and exact algorithms / Vinícius
Wellington Coelho de Moraes. — Belo Horizonte, 2018
xx, 85 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas
Gerais - Departamento de Ciência da Computação
Orientador: Geraldo Robson Mateus

1. Computação - Teses. 2. wireless sensor networks.
2. cardinality constraints . 3. backbone arborescence.
4. simple directed cycles. I. Orientador I . Título..
I. Título.

CDU 519.6*61(043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO


Topological design of Wireless sensor network: Formulations and exact algorithms

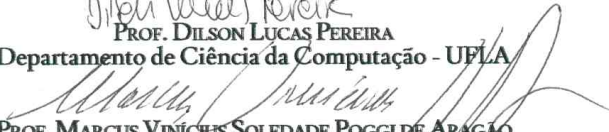
VINÍCIUS WELLINGTON COELHO DE MORAIS

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:


PROF. GERALDO ROBSON MATEUS - Orientador
Departamento de Ciência da Computação - UFMG


PROF. ALEXANDRE SALLES DA CUNHA
Departamento de Ciência da Computação - UFMG


PROF. DILSON LUCAS PEREIRA
Departamento de Ciência da Computação - UFPA


PROF. MARCUS VINÍCIUS SOLEDADE POGGI DE ARAGÃO
Departamento de Informática - PUC RIO


PROF. VINÍCIUS FERNANDES DOS SANTOS
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 31 de agosto de 2018.

For my family, who taught me to persist.

Acknowledgments

First of all, I want to express my gratitude to my advisor Prof. Geraldo Robson Mateus for guiding me through all these years and for clearing my ways into the scientific world. Also due to Prof. Alexandre Salles da Cunha, for his teachings and great helpfulness. Special thanks to my friends and first advisors Alessandro Vivas and Luciana Assis. It has been an honor working with all of you.

Many special thanks go to my mother, my motivation, my strength. To my brother and sisters Alvaro, Vanessa, and Vivian, which have always been the inspiration for my efforts. To Gisele for the encouragement and help. To all my relatives, my cousins, my stepfather, and my grandparents. They directly contributed to my education, character, and personality.

I want to thank my colleagues from DECOM/UFVJM, LaPO/DCC/UFMG, CIRRELT/UdeM and SEEK-AI for having a good and productive time at Diamantina, Belo Horizonte, Montreal and São Paulo. They shared the office with me and contributed directly to my academic and professional growth.

A cordial thanks to the UFMG staff. In particular, many thanks to all from the Departamento de Ciência da Computação for helping me whenever I needed. To the girls on the secretariat, they did everything possible to develop this work.

Finally, I want to thank the Brazilian National Council for Scientific and Technological Development (CNPq), the Foundation for Support of Research of the State of Minas Gerais, Brazil (FAPEMIG), and Coordination for the Improvement of Higher Education Personnel, Brazil (CAPES) for the financial support of my PhD studies.

Abstract

Monitoring events, physical or environmental phenomena in different scenarios like military tactics, security, industrial and health, involves several activities such as sensing, processing, and transmission of data related to temperature, humidity, pressure, movement and so on. Most of the time, collecting and spreading the data requires a huge planning effort, mainly when covering places with difficult access, like mines, forests, volcanoes or even highly radioactive sites. Therefore, these characteristics imply in very costly activities. In consequence of that, the development of efficient approaches to minimize the rising costs is one challenging issue. This scenario is a prominent motivation for the design of Wireless Sensor Networks (WSNs) which establish the main subject of this thesis. Given a limited set of sensors and a single sink, we are particularly interested in the topological design of WSNs. The optimization problem considered here consists in clustering the sensors and defining a communication topology to gather the sensed information throughout the network. Natural connectivity and coverage requirements are satisfied assuming an imposition on the number of clusters. Two variants of the problem are studied: p-arborescence star problem (p-ASP) and p-cycle star problem (p-CSP). p-ASP organizes the network into a fixed number of p clusters and defines the communication topology as a rooted directed tree, i.e., an arborescence, setting direct paths from each sensor to a single sink, the root of arborescence. In p-CSP, instead of a backbone tree, mobile-sink based networks are designed replacing the core arborescence by a directed cycle, representing the route traversed by a mobile sink to visit each of the p predefined cluster-heads. The overall objective of the problems is to maximize the network lifetime by minimizing the clustering and routing costs. We introduce mixed integer programming formulations based on directed cutset constraints, compact multicommodity flow formulations and exact solution approaches. Branch-and-bound, branch-and-cut and column-and-row generation are the algorithms introduced here. To validate our approaches, computational experiments are performed on instance sets extended from the literature involving up to 200 vertices.

Keywords: wireless sensor networks, cardinality constraints, backbone arborescence, simple directed cycles.

List of Figures

1.1	Illustration of a feasible p-ASP solution involving 10 sensors, 4 cluster-heads and the sink.	5
1.2	Illustration of a feasible p-CSP solution involving 14 sensors, 5 cluster-heads and the sink.	6
3.1	Illustration of the <i>pcorecycle</i> backbone for a solution with $p = 2$ that violates inequalities (3.11) and (3.21).	37
4.1	Illustration of decomposition examples for p-ASP and p-CSP solutions in a configuration with the sink and $p = 4$ cluster-heads, together with a set of stars incident to that configuration.	48
4.2	Pseudo-code of Lagrangian heuristic to solve the pricing subproblem implemented in the CRG procedure.	54
4.3	Illustration of two different solutions based on the same configuration. . . .	55
4.4	Pseudo-code of the overall column-and-row (CRG) generation for p-ASP and p-CSP. In the CRG, the PricingSubproblem procedure (Line 8) uses algorithms for solving the pricing subproblems given in subsection 4.2.1. . .	57

List of Tables

2.1	Linear programming lower bounds for p-ASP formulations.	23
2.2	Comparison of the aggregated computational results for BCF-C, BCF-T and BCD when solving STS instances [Bardossy and Raghavan, 2010] adapted to p-ASP.	24
2.3	Detailed computational results: BCF-C, BCF-T, and BCD when solving p-ASP instances with $29 \leq n \leq 100$	26
2.4	Detailed computational results: BCF-T, BCF-C and BCD when solving p-ASP instances with $126 \leq n \leq 199$	27
3.1	Linear programming lower bounds for p-CSP formulations.	42
3.2	Comparison of the aggregated computational results for BBF, BCF and BCD when solving p-CSP instances.	43
4.1	Comparison of formulations \mathcal{P}_F , \mathcal{P}_D and \mathcal{P}_a for p-ASP. Instances on which the procedure finds certified optimal values are indicated with (*) and instances on which the procedure finds an optimal solution but fails in providing an optimality certificate are indicated with (+).	60
4.2	Comparison of formulations \mathcal{F}_u , \mathcal{D}_d , \mathcal{D}_d^+ and \mathcal{P}_c for p-CSP. Instances on which the procedure finds certified optimal values are indicated with (*) and instances on which the procedure finds an optimal solution but fails in providing an optimality certificate are indicated with (+).	62
A.1	Detailed computational results for instances adapted from Bardossy and Raghavan [2010] : BCF-C, BCF-T and BCD when solving p-ASP instances with $ V = 100$ and $p = 10$	70
A.2	Detailed computational results for instances adapted from Bardossy and Raghavan [2010] : BCF-C, BCF-T and BCD when solving p-ASP instances with $ V = 100$ and $p = 20$	71

- A.3 Detailed computational results for instances adapted from Bardossy and Raghavan [2010] : BCF-C, BCF-T and BCD when solving p-ASP instances with $|V| = 100$ and $p \in \{30, 40\}$ 72
- A.4 Detailed computational results for instances adapted from Bardossy and Raghavan [2010] : BCF-C, BCF-T and BCD when solving p-ASP instances with $|V| = 100$ and $p \in \{50, 60\}$ 73
- A.5 Detailed computational results: BBF, BCF and BCD when solving p-CSP instances with $51 \leq |V| \leq 100$ 75
- A.6 Detailed computational results: BBF, BCF and BCD when solving p-CSP instances with $127 \leq |V| \leq 200$ 76

Contents

Acknowledgments	xi
Abstract	xiii
List of Figures	xv
List of Tables	xvii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	3
1.3 Topological WSN problems	4
1.4 Literature review	7
1.5 Outline and main contributions	10
2 The p-arborescence star problem	13
2.1 p -ASP formulations	13
2.1.1 Directed multicommodity flow formulation	15
2.1.2 Directed cutset based formulation	16
2.1.3 Strengthening the formulations	17
2.2 Algorithms	18
2.2.1 Branch-and-cut algorithm based on \mathcal{P}_F	18
2.2.2 Branch-and-cut algorithm based on \mathcal{P}_D	19
2.2.3 Benders-based heuristic	20
2.3 Computational results	21
2.3.1 LP relaxation lower bounds	22
2.3.2 Comparison of Branch-and-cut algorithms	24
2.4 Concluding remarks	29

3	The p-cycle star problem	31
3.1	p -CSP formulations	31
3.1.1	Additional valid inequalities	33
3.1.2	Projecting out the flow variables	35
3.2	Algorithms	38
3.2.1	Primal heuristic	38
3.2.2	Separation routines	38
3.2.3	Exact algorithms	39
3.3	Computational results	40
3.4	Concluding remarks	45
4	Configuration-based approach for topological problems in the design of wireless sensor networks	47
4.1	Configuration-based formulations	47
4.2	Column-and-row generation algorithms	51
4.2.1	Pricing subproblem	51
4.2.2	EDCUTs separation routine	55
4.2.3	CRG miscellanea	55
4.3	Computational results	58
4.4	Concluding remarks	64
5	Epilogue	65
5.1	Final Remarks	65
	Appendix A Supplementary results	69
	Bibliography	77

Chapter 1

Introduction

In this Chapter, we place the thesis in the context of *wireless sensor networks* (WSNs). We start detaching the main problems in the design of WSNs that we are about to cover. In section 1.3, we formally define the topological problems addressed in the thesis. Literature is reviewed in section 1.4. Finally, in the last sections, we close the chapter reviewing the main ingredients of the work.

1.1 Motivation

Monitoring events, physical or environmental phenomena in different scenarios such as military tactics, security, industry and health, involves several activities such as sensing, processing and transmission of data related to temperature, humidity, pressure of gases or even liquids, movement and so on [Arampatzis et al., 2005; Kuorilehto et al., 2005; Alemdar and Ersoy, 2010; Rawat et al., 2014; Carrabs et al., 2015]. Most of the time, collecting and spreading the data requires a huge planning effort, mainly when covering places with difficult access, such as mines, forests, volcanoes or even highly radioactive sites. Therefore, these characteristics lead to very costly activities. In consequence of that, the development of efficient approaches to minimize the arising costs is a challenging issue.

The above scenario is a prominent motivation for the design of wireless sensor networks (WSNs) [Morais et al., 2016c], which establish the main subject of this thesis: consider a set of autonomous, tiny and compact *sensors*, capable of performing activities such as sensing, processing, and transmission, and a set of *sinks* responsible for collecting the sensed data and managing the network. A sink has an unlimited or renewable energy source and it also has the function of processing and aggregating the information received from sensors. It can be *static* or *mobile*, visiting the monitored

site and collecting data [Kim et al., 2003; Basagni et al., 2008]. The sensors are made up of sensing boards, processor, transmission radio and battery. Because of their purposes, size and low cost, they have serious restrictions of energy, low performance and small radius of transmission. These characteristics lead to a limited lifetime of sensors and also the network itself [Anastasi et al., 2009]. In order to overcome such a problem, the sensors must be organized and a set of routes has to be chosen defining the transmission paths among sensors and sinks. Thereby, we aim to maximize the network lifetime by minimizing the routing costs. Besides natural connectivity requirements, in this thesis, we also deal with covering and cardinality constraints in an integrated manner.

The definition of a WSN topology involves ensuring fundamental requirements, being coverage probably the central point on the design of a such network. Coverage refers to the portion of sensing area that is within reach of at least one sensor. The coverage level can be total or partial, but, in general, coverage of the entire sensing area is always preferred. Therefore, one has to choose a set of active sensors such that each demand point is within a range of at least one sensor. Indeed, fulfilling such a need becomes a crucial problem when placing the sensors: a subset of sensors must be deployed in order to guarantee connectivity and coverage of the entire sensing area [Efrat et al., 2005; Castano et al., 2014]

Data transmission is one of the most expensive operations in terms of energy consumption [Akyildiz et al., 2002]. An alternative to maximize the network lifetime is to define hierarchical networks and implement efficient routing approaches in order to avoid premature disconnection due to the energy depletion. In many cases, the transmission topology is arranged in clusters, each one having a sensor assuming the role of *cluster-head*. Through some wireless protocol, the cluster-head is responsible for collecting and aggregating the data from other sensors within the cluster and transmitting it directly or indirectly to a sink. Regular sensors within a cluster must communicate directly with the elected cluster-head, using a single-hop strategy or through other sensors using a multi-hop protocol [Aioffi et al., 2011]. Classical combinatorial optimization problems as p -median [Hakimi, 1964, 1965] and k -center [Agarwal and Sharir, 1998] appear naturally in such a clustering context to efficiently manage the network energy consumption by reducing the transmission range of the sensors. The transmission among cluster-heads and sinks is typically multi-hop. Thus, the connectivity requirement is enforced with a two-level sensor-to-sink scheme, *intra-cluster* and *inter-cluster* routes. In both levels, a direct structure is defined to transmit data from sensors to cluster-heads (intra-cluster routes) and from cluster-heads to sinks (inter-cluster routes).

An important requirement that strongly impacts on the design of WSNs and directly influences on the complexity of meeting data routing as well as other Quality of Service (QoS)¹ parameters, is the *cardinality constraint*. Knowing that aggregation and data transmission are the two most expensive activities in terms of energy consumption, the cluster-heads are expected to consume much more energy compared to regular sensors, that only monitor and transmit data. In general, they deplete their batteries very quickly becoming isolated from the rest of the network. Thus, in order to prolong the network lifetime, clustering protocols like the Minimum Transmission Energy Routing protocol, LEACH [Heinzelman et al., 2000], and LEACH-C [Heinzelman et al., 2002] perform a periodic rotation on a fixed number of preselected cluster-heads. The cardinality of such a cluster-head set is typically determined as a percentage of active sensors within each time period. As we will discuss, restricting the cardinality of a cluster-head set and consequently the number of clusters to be defined is a very hard requirement on the design of WSNs and, in consequence of that, is an important design ingredient explored in our work.

Network lifetime is defined as the time span from deployment to the instant in which the network is considered nonfunctional [Chen and Zhao, 2005]. However, the moment when a network is considered nonfunctional is specific to the application. Connectivity is an important factor tied to network utility and an essential consideration in the definition of network lifetime. The connectivity-related network lifetime definitions include: 1) Time until the first sensor failure [Chen and Zhao, 2005; Kang and Pooven-dran, 2005]; 2) Time until the failure of a certain percentage of sensor nodes [Raicu et al., 2005]; 3) Time until all the sensors die [Heinzelman et al., 2000]; 4) Time until the first cluster-head failure or no communication backbone exists [Chen and Zhao, 2005; Dong, 2005]. The most common lifetime definition is the first one [Hang and Seah, 2009]. Because of that, in this work, we studied a scenario where lifetime is based on that definition.

1.2 Objectives

In this thesis, we are particularly interested in two topological WSN problems with cardinality constraints: *p-arborescence star problem* (p-ASP) and *p-cycle star problem* (p-CSP). p-ASP organizes the network into a fixed number of p clusters and, in the first level, it defines a transmission topology as a rooted directed tree, i.e., an *arborescence* establishing directed paths from each cluster-head to a single sink, the root of ar-

¹QoS usually refers to quality as perceived by the user and/or application.

borescence. Intra-cluster routes, in the second level, follow a single-hop protocol while multi-hop is preferred for inter-cluster routes. On the other hand, in the P-CSP, instead of a backbone arborescence, a mobile-sink based network is designed to replace the core arborescence by a directed cycle, representing the trail traversed by the mobile sink to visit each of the p predefined cluster-heads. This approach prevents the sensor from spending their limited energy in relaying messages through a long path to the sink. In the present study, we investigate the application of a mathematical programming technique to those problems. We decompose the topology structure of the problems in a way to obtain strong mathematical formulations. In the next section, we present a detailed description of the problems addressed in this thesis.

1.3 Topological WSN problems

Let $D = (V, A)$ be a connected bidirectional digraph with a set V of $n + 1$ vertices (n sensors and one sink) and a set A of m arcs. $r \in V$ is a special vertex named *sink*. Assume that a positive integer p ($1 \leq p < n$) is given and define the vertex subset $W = \{r, w_1, \dots, w_p\} \subseteq V$. $|W| = p + 1$ and $W \setminus \{r\}$ is denoted by *cluster-head* set. The p -*arborescence star problem* (p-ASP) consists in clustering the vertices into p clusters (disjoint vertex subsets), $\{K^{w_1}, \dots, K^{w_p}\}$, each one associated with a *cluster-head*, such that: i) $V \setminus \{r\} = \bigcup_{i=1}^p K^{w_i}$; ii) the *sink* and all p *cluster-heads* are connected with p arcs forming a spanning reverse arborescence rooted at r ; iii) every other vertex $j \in V \setminus W$ is connected by an arc to a *cluster-head* $w_i \in W$. We introduce the following notation: a *backbone* is a subdigraph (W, A_W) , where $A_W \subseteq \{(i, j) \in A : i, j \in W\}$; an *intra-cluster star* is a subdigraph $(K^{w_i}, A_K^{w_i})$, centered at *cluster-head* $w_i \in W \setminus \{r\}$, spanning the vertices within a given cluster K^{w_i} , such that $A_K^{w_i} = \{(j, w_i) \in A : j \in K^{w_i}\}$; the objective of p-ASP is to minimize *intra* and *inter-cluster* total costs, given by $\sum_{a \in A_W} c_a + \sum_{i=1}^p \sum_{a \in A_K^{w_i}} c_a$, where $c_a > 0$ is a positive cost on each arc $a \in A$.

Figure 1.1 illustrates a feasible p-ASP solution involving 10 sensors and 4 *cluster-heads*. The arcs of the connected *backbone*, rooted at the *sink*, are drawn in bold lines. *Intra-cluster* arcs are drawn in dashed lines, connecting each vertex (black dots) $v \notin W$ to a *cluster-head* (white) $w_i \in W \setminus \{r\}$. The *backbone* is a reverse arborescence.

Placing p-ASP in the design of ad-hoc WSN, $V \setminus \{r\}$ represents the sensor set and $r \in V$ the fixed *sink*. In this kind of network, the positive cost c_a , $a = (i, j) \in A$, corresponds to the energy required to send a packet from sensor i to sensor j , or from a sensor i to the *sink* (when $j = r$), and p is the number of *cluster-heads* to be placed.

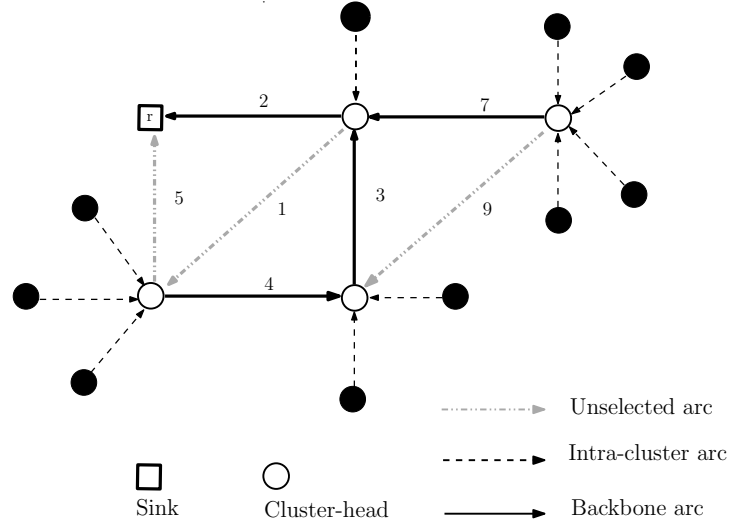


Figure 1.1: Illustration of a feasible p-ASP solution involving 10 sensors, 4 cluster-heads and the sink.

Each *cluster-head* aggregates the packages within a *cluster* and sends them to the *sink* through the *backbone*. In this context, the aim is to maximize the network lifetime by minimizing the routing cost, while ensuring coverage constraints. Note that, in the definitions above, the communication between a *cluster-head* and the other sensors within the same cluster must be *single-hop*, while the communication among *cluster-heads* and *sink* may be *multi-hop* [Aioffi et al., 2011]. The communication model assumes that the sensors are randomly distributed in the network field and have the same transmission range.

The *p-cycle star problem* (p-CSP) is also defined in terms of $D = (V, A)$ and the fixed positive integer $p > 1$ ($p < n$). $r \in V$ represents a higher order vertex, in this problem, a *mobile sink*. As for p-ASP, $W = \{r, w_1, \dots, w_p\} \subseteq V$ is the vertex subset, where $|W| = p + 1$ and $W \setminus \{r\}$ is denoted by *cluster-head set*. Two different sets of nonnegative fixed costs are considered, *cycle cost* $\{c_{(i,j)} > 0 : (i,j) \in A\}$ and *assignment cost* $\{d_{(i,j)} > 0 : (i,j) \in A\}$. p-CSP consists in clustering the vertices into p clusters, $\{K^{w_1}, \dots, K^{w_p}\}$, each one associated with a *cluster-head*, such that: i) $V \setminus \{r\} = \bigcup_{i=1}^p K^{w_i}$; ii) the *sink* and all p *cluster-heads* are connected with $p + 1$ arcs forming a *simple directed cycle* (W, A_W) of D , where $A_W = \{(i,j) \in A : i,j \in W\}$ and $|A_W| = p + 1$; iv) every other vertex $j \in V \setminus W$ is connected by an arc to a *cluster-head* $w_i \in W$. As before, *intra-cluster star* is a subdigraph $(K^{w_i}, A_K^{w_i})$, centered at *cluster-head* $w_i \in W \setminus \{r\}$, spanning the vertices within a given cluster K^{w_i} , such that $A_K^{w_i} = \{(j,w_i) \in A : j \in K^{w_i}\}$. The objective of p-CSP is to minimize *inter* and *intra-cluster* total costs, given by $\sum_{(i,j) \in A_W} c_{i,j} + \sum_{i=1}^p \sum_{(i,j) \in A_K^{w_i}} d_{i,j}$.

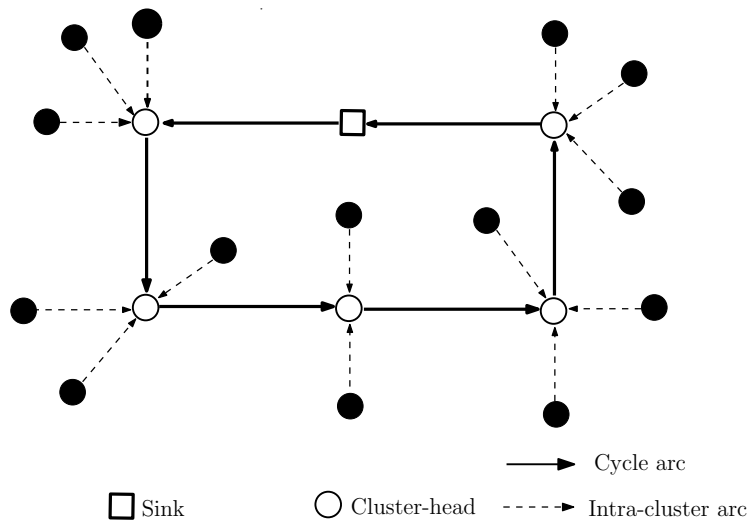


Figure 1.2: Illustration of a feasible p-CSP solution involving 14 sensors, 5 cluster-heads and the sink.

Figure 1.2 illustrates a feasible p-CSP solution involving 14 sensors, 5 *cluster-heads* and a *sink*. The arcs of the *directed cycle* are drawn in bold lines. *Intra-cluster* arcs are drawn in dotted lines, connecting each sensor (black dots) $v \notin W$ to a *cluster-head* (white) $w_i \in W \setminus \{r\}$. The *simple directed cycle* has exactly $p+1$ arcs. Besides, exactly one arc pointing into and other pointing out of a *cluster-head* there exist.

p-CSP appears in the design of ad-hoc wireless sensor networks (WSN) [Morais et al., 2016c], where one common strategy to extend the WSN lifetime is to use a mobile *sink* to gather sensed data through the sensor network. This approach prevents the sensors to spend their limited energy in relaying of messages through paths to the *sink*. However, in that approach, the message delivery latency can increase significantly due to the long route traversed by the *sink* to visit each sensor. In fact, since a WSN is expected to have a large number of sensors it could be impractical to visit each sensor. In order to go around of such problem, clustering schemes with a predetermined number of cluster are considered. Then, p *cluster-heads* have to be placed over the network and *clusters* must be defined connecting sensors to the *cluster-heads*. In regard of modelling WSN applications as p-CSP, $V \setminus \{r\}$ represents the sensor set and $r \in V$ the mobile *sink*. The *assignment cost* corresponds to the cost of sending a packet from sensor $i \in V$ to a *cluster-head* $w_j \in W$, such that $(i, w_j) \in A$, and the *cycle costs* correspond to the cost of the route traversed by the *sink* to visit all *cluster-heads*.

In the problems above, the communication model assumes that the sensors are randomly distributed in the network field and have the same transmission range. The communication between any pair of sensors is bidirectional, i.e., for any pair of sensors i and $j \in V$, i is in the transmission range of j only if j is in the transmission range of

i. This imply that if there exists an arc $(i, j) \in A$ then $(j, i) \in A$.

1.4 Literature review

The class of problems that aims to optimize the energy consumption and extend the WSN lifetime is known as the *maximum lifetime problems* (MLPs) [Cardei et al., 2005; Carrabs et al., 2015]. Typically, problems in that class involve the definition of topologies and the density control of sensors (DCP). DCP being the problem of planning the sequence of activation and deactivation of sensors while ensuring WSN requirements [Aioffi et al., 2011]. Accordingly, MLP variants seek to define sensor sets, not necessarily disjoint, to cover the sensed area and set their appropriate activation schedule times in such a way that the sensors in the cover being used in a given moment are kept in active state while all the other sensors are turned off to save energy. In the literature, enforcing connectivity among active sensors leads to the study of the *connected maximum network lifetime problem* (CMLP) [Raiconi and Gentili, 2011] when coverage of the entire sensing area is required and α -CMLP [Gentili and Raiconi, 2013; Castano et al., 2014; Carrabs et al., 2015] when partial coverage is preferred. In both cases, a communication topology connecting active sensors and sink must be defined. In particular, this is the purpose of topological problems like: p-ASP and p-CSP.

p-ASP belongs to the vast literature of clustering problems, where some of the simplest problems are known to be NP-hard [Agarwal. and Procopiuc, 1998], including euclidean *k-center* [Agarwal and Sharir, 1998] and *p-median* [Hakimi, 1964, 1965] problems. Introduced in [Matos et al., 2012], p-ASP is motivated from an application in wireless sensor networks where a clustering protocol to simulate the clusters formation and data transmission is designed. To solve the problem, a GRASP-based heuristic was also presented in [Matos et al., 2012]. The good performance of their protocol were compared with LEACH and LEACH-C. p-ASP is trivial to solve when $p = 1$: first compute $c_{i,r} + \sum_{a \in \delta_i} c_a$ for each vertex $i \in V \setminus \{r\}$, assuming that non-existing arcs have infinity costs, where $\delta_i = \{a \in A : a = (j, i), j \in V \setminus \{r\}\}$; then, choose as *cluster-head* the vertex that minimizes the total costs. Another polynomially solvable case of the problem applies when $p = n$, in which case the problem reduces to the minimum weight spanning arborescence problem (MSA), which can be solved efficiently [Chu and Liu, 1965; Edmonds, 1967]. In general, p-ASP is NP-hard, since the *p-median* problem can be reduced to it. The *p-median* problem is a simple uncapacitated facility location problem where a fixed number p of facilities must be selected. If one imposes

the *sink* to be among the selected facilities and conveniently define costs $c_a > 0, a \in A$, to *intra-cluster* arcs and $c_a = 0$ to *backbone* arcs, solving the p -median is equivalent of dropping the *backbone* arcs from a solution of p -ASP.

Another closely related problem is the *tree-star problem* (T-SP) [Nguyen and Knippel, 2007; Lucena et al., 2015]. T-SP is defined as the problem of finding a minimum cost spanning tree of an undirected graph $G = (V, E)$, assuming different cost structure for edges connecting *leaves* and internal vertices. T-SP is related to the *minimum connected dominating set problem* (MCDSP) [Gendron et al., 2014], the problem of finding a connected subset $C \subset V$ with smallest cardinality, such that $\Gamma(C) = V$, where $\Gamma(C) = C \cup \{j \in V : \{i, j\} \in E, i \in C\}$. Given a dominating set C , a solution of MCDSP is a subgraph $(C, E(C))$, where $E(C) = \{\{i, j\} \in E : i, j \in C\}$ plays the role of *backbone*. Unlike p -ASP, in T-SP and MCDSP, the *backbone* is modeled in terms of an undirected graph and has no fixed cardinality associated to it; moreover, the sink does not need to be connected to the *backbone*. Exact algorithms based on mixed-integer programming (MIP) formulations were developed for T-SP [Nguyen and Knippel, 2007], in particular a branch-and-cut algorithm that separates generalized subtour elimination constraints (GSECs) [Lucena et al., 2015]. For MCDSP, MIP formulations and exact algorithms were proposed in [Gendron et al., 2014; Simonetti et al., 2011]. Similarly to p -ASP, T-SP and MCDSP are used in the design of ad-hoc WSN applications where the network topology may change dynamically.

A clustering problem resembling p -ASP is the *connected facility location problem* (ConFL) [Gollowitzer and Ljubić, 2011]. In ConFL, it is assumed that the set of vertices is partitioned into potential facility locations I , customers J and interconnection points K , where $I \cup K$ is the set of *Steiner* vertices, while the vertices in J are named *terminals*. ConFL consists of selecting a subset of facilities to open, assigning each customer to an open facility and defining a *Steiner* tree spanning terminals and *Steiner* vertices, using or not interconnection points to enforce connectivity. The objective is to minimize assignment costs, facility opening costs and *Steiner* tree costs. The topological design of ConFL define trees spanning open facilities and customers. Bardossy and Raghavan [2010]; Leitner et al. [2017] addressed the ConFL and discussed how their solution methods are also applied to a family of tree-star problems known in the literature. More precisely, Leitner et al. [2017] described the Steiner tree-star (STS) [Lee et al., 1993], generalized Steiner tree-star [Khuller and Zhu, 2002] and rent-or-buy problems [Swamy and Kumar, 2004] in graphs as special cases of the asymmetric version of the ConFL. In [Leitner et al., 2017], to solve the problems addressed in their paper, a sophisticated branch-and-cut based framework was also implemented. Polyhedral investigations for ConFL versions were carried out in [Gollowitzer and Ljubić,

2011; Leitner et al., 2017; Gollowitzer et al., 2013; Leitner et al., 2015], branch-and-cut algorithms were developed in [Arulselvan et al., 2011]. A branch-and-cut-and-price algorithm for a variant of ConFL with capacities on facilities (CapConFL) appeared in [Leitner and Raidl, 2011]. Exact solution methods based on compact formulations for a ConFL with capacities on the facilities and links of the networks were also investigated in [Gollowitzer et al., 2013]. Heuristics for connected facility location and related problems were described in [Bardossy and Raghavan, 2010] and [Ljubić, 2007].

A location-allocation problem also close to p-ASP is the *capacitated p-cable trench problem with covering problem* (Cp-CTPC) [Calik et al., 2017]. Cp-CTPC belongs to the family of *cable trench problems* (CTP) [Vasko et al., 2002], in which it is assumed that the set of vertices V is partitioned into clients J and potential primary S and secondary I servers (i.e., $V = I \cup S \cup J$). Each client $j \in J$ has a positive demand q_j and each secondary server $i \in I$ has positive capacity Q_i . Each client $j \in J$ must be supplied by a secondary server $i \in I$ respecting capacity and a covering radius r , i.e., the distance $d_{ij} \leq r$. In particular, *cable trench problems* combine the minimum spanning tree problem and the path generation problems into a multi-hierarchical topology where no backbone connecting the multiple primary servers is required. Fixed costs are associated with cable installation in addition to the cable usage costs that depend on the number of paths between the primary vertex and any of the other nodes using a cable. Thus, the cost structure does not depend on the client coverage cost. As a variant of the CTP, the Cp-CTPC consists in select p primary servers, an arbitrary number of secondary servers and define a topology connecting secondary servers to the selected primary ones, respecting covering and capacity constraints. Cp-CTPC also generalizes the most studied *cable trench* variants, the *p-cable-trench problem* (p-CTP) [Marianov et al., 2012]. This is the problem of locating p primary servers such that each client must be connected to one such facility. Aside from introducing Cp-CTPC and placing it in the *cable trench* literature, Calik et al. [2017] also designed an algorithmic framework based on Benders decomposition. Contrary to CTP problems, p-ASP incorporates the design aspect of the well-known p-median problem and the minimum cost arborescence problem. Its objective is to define a topology rooted into a fixed sink (i.e., $|S| = 1$), with a constraint on the number of cluster-heads (open secondary servers), taking into account the coverage cost over the cost structure. As stated before, in p-ASP there are neither opening costs nor capacity constraints associated to *cluster-heads*.

Various related problems appear in the clustering and routing literature. They are also based on cycle topologies, like p-CSP. For instance, a topological design problem similar to p-CSP is the *travelling circus problem* (TCP). TCP consists of defining a tour with p out of $|V|$ vertices of a network such that the length of the tour and the

assignment cost to satisfy the demand of all customers from the vertices on the tour are minimized. Hartmann and Özgür Özlük [1999] extended their results on the p -cycle (*simple directed cycle* consisting of p arcs) polytope [Hartmann and Özgür Özlük, 2001] to solve the TCP. A TCP formulation on complete digraphs is given in [Current and Schilling, 1994]. The undirected version of the p -cycle was investigated by Nguyen and Maurras [2001], which brought polyhedral analysis to the problem. The study of all these problems has been supported by contributions for the *simple cycle problem* on directed [Balas and Oosten, 2000] and undirected [Fischetti et al., 2007] graphs. p -CSP is also related to the *ring star problem* (RSP) [Labbé et al., 2004], the problem of defining a set of stars and connect them through a ring structure. Likewise the TCP and p -CSP, RSP is a location-allocation problem but contrary to its counterpart the simple cycle on its core structure is defined as an undirected subgraph. Moreover, it has no fixed root (the sink) and no fixed cardinality. Among all the referred problems, RSP is the only one that like p -CSP imposes a prefixed vertex, the *root*, to be in the optimal cycle. This condition is justified by the application requirements, that for the p -CSP also forbids the direct assignment of a *non-cluster-head* vertex to the *sink*.

Except for ConFL and CapConFL, the related problems described above are also motivated in the WSN context. Furthermore, p -ASP and p -CSP have some similarities with other location-allocation problems in transportation and telecommunication networks. In particular the connected hub location problems, such as the *hub line* [de Sá et al., 2015], *tree of hubs* and *cycle hub location* problems [Contreras et al., 2010], as well as the *capacitated m -ring-star* [Baldacci et al., 2007], *median cycles* [Labbé et al., 2005] and minimum cost hop-and-root constrained forest [Pereira and da Cunha, 2018] problems. Those problems consist of locating a subset of vertices and connecting them through paths, cycles or trees, considering assignment costs to allocate non-located vertices to the backbone and capacity/budget constraints.

1.5 Outline and main contributions

The thesis consists of five chapters. Below we outline each of the remain chapters and highlight the main contributions.

Chapter 2 is dedicated to the *p -arborescence star problem*. We introduce MIP formulations and exact solution methods for p -ASP. Contrary to the related problems found in the literature, our models specify a *backbone* defined by directed paths from each vertex to the *sink*, traversing a set of p *cluster-heads*. The formulations are based on directed multicommodity flow models and on directed cutset (DCUT) inequalities.

We also develop three branch-and-cut algorithms and a combinatorial Benders-based heuristic, which is used to provide the initial solution to the exact algorithms. Based on compact multicommodity flow formulations, two algorithms (BCF-T and BCF-C) exploit different approaches to handle constraints that impose the flow capacity on the *backbone* arcs. Based on the DCUT formulation, the third algorithm (BCD) manages DCUT inequalities in a cutting-plane fashion. The Benders-based heuristic alternates between two problems: a master problem and a subproblem. The master problem defines a *cluster-head* set W , while the subproblem verifies the feasibility of the *cluster-head* set provided by the master problem, i.e., it evaluates the *backbone* connectivity and the coverage of vertices in $V \setminus W$. Optimal solutions obtained with the proposed branch-and-cut algorithms are reported for instances with at most 200 vertices and 60 *cluster-heads*.

Chapter 3 concerns to the *p-cycle star problem*. First, a compact formulation based on a directed multicommodity flow model is given. This formulation is then strengthened with additional valid inequalities. With the resulting formulations on hands, we derive cutset based inequalities by projecting out the continuous flow variables. All of this is described in Section 3.1. Additionally, a branch-and-bound and two non-standard branch-and-cut algorithms based on the proposed formulations are given in Section 3.2. Comments over the computational results and the complexity to solve the problem appear in Section 3.3. We close the chapter in Section 3.4.

Chapter 4 concerns formulations based on configurations for p-ASP and p-CSP. Essentially, we decompose the topology structure of the refereed topological WSN problems in such a way to obtain strong mathematical formulations. In order to evaluate bounds of the proposed formulations, we also introduce *column-and-row* generation (CRG) procedures. The chapter is organized as follows. In Section 4.1, we introduce the mathematical programming formulations. In Section 4.2, the column-and-row generation algorithms are described. Our computational experiments are presented and discussed in Section 4.3. Finally, the conclusion and future directions are drawn in the last section.

The thesis is fundamentally based on the following publications:

- Morais et al. [2015], *Formulating and solving the coverage constrained p-tree problem*. Presented at the 2015 *CORS/INFORMS* International Meeting. This work concerns to formulation and preliminary results of the BC algorithm for p-ASP.
- Morais et al. [2016b], *The p-cycle star problem: Formulations and cutting-plane methods*. Presented at the 4th International Symposium on Combinatorial Opti-

mization (2016-ISCO). This work concerns to compact formulation and preliminary results of cutting plane algorithms for the p-CSP.

- Morais et al. [2019] *The p-arborescence star problem: Formulations and exact solution approaches*. Work published in Computer and Operations Research. In this paper we address the p-ASP. We introduce and compare formulations based on directed cutset constraints and multicommodity flow variables. Valid inequalities to strengthen the models are also given. To solve these models, we develop efficient cutting-plane methods. We carry out extensive computational experiments and comparisons to illustrate the performance of our methods.
- Morais et al. [2016c], *Optimization problems, models, and heuristics in wireless sensor networks*. Book chapter printed in Handbook of Heuristics. This chapter presents a literature review on optimization problems, models and heuristics in the context of Wireless Sensor Networks.
- Morais and Mateus [2019] *Configuration-based approach for topological problems in the design of wireless sensor networks*. Work published in International Transactions in Operational Research. In this paper we propose configuration-based formulations for p-ASP and p-CSP and design a *column-and-row* generation (CRG) procedure to evaluate the bounds.

In parallel with his research topic, the student has been participating in other works related to his research topic: Formulations for optimization problems. As a result, the following work has been published:

- Morais et al. [2016a], *A Branch-and-cut-and-price algorithm for the Stackelberg Minimum Spanning Tree Game*, published in Electronic Notes in Discrete Mathematics. This work introduces a reformulation and a Branch-and-cut-and-price algorithm for the Stackelberg Minimum Spanning Tree Game (StackMST) .

Chapter 2

The p -arborescence star problem

This Chapter is dedicated to the *p-arborescence star problem*. We introduce MIP formulations and exact solution approaches for the problem. In Section 2.1, we present the different p-ASP formulations, namely the multicommodity flow formulation and the DCUT formulation, as well as valid inequalities to strengthen the models. In Section 2.2, we present branch-and-cut algorithms and a Benders-based heuristic. The computational experiments are discussed in Section 2.3. We close the chapter in Section 2.4.

2.1 p-ASP formulations

To present p-ASP formulations, we introduce some notation that will be used throughout the chapter. Denote by $\delta^+(S) = \{(i, j) \in A : i \in S, j \notin S\}$ the set of arcs pointing out from the vertex subset $S \subset V$ and by $\delta^-(S) = \{(i, j) \in A : i \notin S, j \in S\}$ the set of arcs pointing into the vertex subset S . For simplicity, we use δ_i^+ and δ_i^- instead of $\delta^+(\{i\})$ and $\delta^-(\{i\})$ when referring to the vertex subset $S = \{i\}$. Given a subset $C \subset V$, we define the subdigraph $D_C = (C, A_C)$, where $A_C = \{(i, j) \in A : i, j \in C\}$.

p-ASP formulations use binary variables $\{h_i \in \mathbb{B} : i \in V\}$, where $\mathbb{B} = \{0, 1\}$, to identify a set of feasible *cluster-heads* to be located. Accordingly, $h_i = 1$ if vertex i is selected to be a *cluster-head*, $h_i = 0$ otherwise. For consistency, since the *sink* belongs to the *backbone*, we let $h_r = 1$.

Additionally, binary variables $\{y_{ij} \in \mathbb{B} : (i, j) \in A\}$ are used to identify whether a vertex $i \in V \setminus \{r\}$ is covered ($y_{ij} = 1$) or not ($y_{ij} = 0$) by a *cluster* with *cluster-head* located at $j \in V \setminus \{r\}$, while binary variables $\{z_{ij} \in \mathbb{B} : (i, j) \in A\}$ are used to identify *backbone* arcs, i.e., $z_{ij} = 1$ if (i, j) belongs to the *backbone*, $z_{ij} = 0$ otherwise. Here,

the *backbone* is represented as a reverse arborescence rooted at r , i.e., there are no arcs pointing out from r . Then, the general p -ASP formulation is given by:

$$\min \left\{ \sum_{(i,j) \in A} c_{ij} y_{ij} + \sum_{(i,j) \in A} c_{ij} z_{ij} : (h, y, z) \in \mathcal{P} \cap (\mathbb{B}^{n+1} \times \mathbb{B}^m \times \mathbb{B}^m) \right\} \quad (2.1)$$

where $\mathcal{P} = \mathcal{P}_c \cap \{(2.9)\}$ denotes the intersection of the polyhedron \mathcal{P}_c , given by

$$\sum_{j \in \delta_i^+} y_{ij} = 1 - h_i, \quad i \in V \setminus \{r\} \quad (2.2)$$

$$\sum_{i \in V \setminus \{r\}} h_i = p, \quad (2.3)$$

$$(z_{ij} + z_{ji}) + y_{ij} \leq h_j, \quad (i, j) \in A \quad (2.4)$$

$$\sum_{j \in \delta_i^+} z_{ij} = h_i, \quad i \in V \setminus \{r\} \quad (2.5)$$

$$h_r = 1, \quad (2.6)$$

$$\sum_{i \in V} (z_{ri} + y_{ir} + y_{ri}) = 0, \quad (2.7)$$

$$\sum_{i \in \delta_r^+} z_{ir} \geq 1, \quad (2.8)$$

with the generic connectivity constraint

$$\text{Backbone}(h, z). \quad (2.9)$$

Constraint (2.9) imposes the connectivity of the *backbone* induced by h and z . $\text{Backbone}(h, z)$ defines a spanning arborescence polytope considering only the *cluster-heads* and the *sink*. Constraints (2.2)-(2.4) ensure that all vertices are spanned and enforce the location of p *cluster-heads*. These constraints define the clustering problem.

Constraints (2.2) determine whether or not an *intra-cluster* arc $(i, j) \in A$ appears in a feasible cluster. If $h_i = 0$ holds, vertex i is left out of the *cluster-head* set and no *backbone* arc pointing out of it exists. In this case, there must exist a *cluster-head* $j \in W$, such that $h_j = 1$, and an *intra-cluster* arc pointing outward from i towards j therefore exists, i.e., $y_{ij} = 1$ holds. Otherwise, if $h_i = 1$ applies, no *intra-cluster* arc pointing out of it exists, since $i \in W$. Thus, in that case, $y_{ij} = 0$ must apply for all $j \in \delta_i^+$.

Constraint (2.3) imposes the cardinality of the *cluster-head* set. Alongside with

(2.5) and (2.6), (2.3) also enforces the cardinality of the *backbone*. Constraints (2.4) play the role of *linking* constraints among variables y , z and h . If an arc $(i, j) \in A$ is selected to be in the *backbone*, the same arc cannot be selected as an *intra-cluster* arc. Consequently, in the former case, both of its endpoints must be *cluster-heads*. Constraints (2.5) force that each *cluster-head* has an outgoing *backbone* arc. Constraints (2.6) and (2.7) are the consistency constraints for the *sink*. They state that no vertices in $V \setminus W$ are connected to the *sink* and no *backbone* arc pointing out of it exists. Likewise, we must impose that at least one *backbone* arc must reach the *sink*. This is done by adding constraint (2.8). Finally, the objective function (2.1) minimizes *intra* and *inter-cluster* total cost.

The exact algorithms described in this chapter differ, mainly, in the way they handle the generic constraint (2.9). Based on that, two different formulations are considered to represent $Backbone(h, z)$. In Section 2.1.1, we describe a classical multicommodity flow formulation involving additional continuous variables. In Section 2.1.2, we present a directed cutset (DCUT) based formulation where connectivity is ensured by means of an exponential number of inequalities. Valid inequalities to strengthen both formulations are given in Section 2.1.3.

2.1.1 Directed multicommodity flow formulation

In this section, we introduce our first approach to model p-ASP. $Backbone(h, z)$ is modeled in terms of a classical multicommodity flow formulation that introduces additional continuous flow variables. The following notation is used. Let $Q = V \setminus \{r\}$ be the set of commodities, sharing the same destination r , each one having a single origin $q \in V \setminus \{r\}$. Denote by $\{f_{ij}^q \in \mathbb{R}_+ : (i, j) \in A, q \in Q\}$ the set of continuous flow variables, representing the fraction of flow on each arc (i, j) for each commodity $q \in Q$. Alongside the decision variables $\{h_i \in \mathbb{B} : i \in V\}$ and $\{z_{ij} \in \mathbb{B} : (i, j) \in A\}$ defined above, assume that $\mathcal{P}_f = \mathcal{P}_c \cap \{(2.10), (2.11), (2.12)\}$ denotes the set obtained from \mathcal{P} by replacing constraint (2.9) by the following constraints:

$$\sum_{(i,j) \in \delta_i^+} f_{ij}^q - \sum_{(j,i) \in \delta_i^-} f_{ji}^q = \begin{cases} 0 & \text{if } i \neq q, \\ h_i & \text{if } i = q, \end{cases} \quad i \in V \setminus \{r\}, q \in Q, \quad (2.10)$$

$$f_{ij}^q \leq z_{ij}, \quad (i, j) \in A, q \in Q, \quad (2.11)$$

$$f_{ij}^q \geq 0, \quad (i, j) \in A, q \in Q. \quad (2.12)$$

Constraints (2.10) represent the *flow conservation* equations for each vertex-

commodity pair. These constraints ensure that only vertices selected to be *cluster-heads* are able to send flow to the *sink*. The *forcing constraints* (2.11) imply that the flow for each commodity only traverses an arc that belongs to the *backbone* defined by the z -variables. Finally, (2.12) impose the non-negativity of the flow variables.

Thus, the following compact p-ASP formulation is obtained:

$$\min \left\{ \sum_{(ij) \in A} c_{ij} y_{ij} + \sum_{(ij) \in A} c_{ij} z_{ij} : (h, y, z, f) \in \mathcal{P}_f \cap \hat{O} \right\} \quad (2.13)$$

where $\hat{O} = (\mathbb{B}^{n+1} \times \mathbb{B}^m \times \mathbb{B}^m \times \mathbb{R}^{m \times n})$.

2.1.2 Directed cutset based formulation

In this section, we introduce our second approach to model p-ASP. *Back bone*(h, z) is represented here in terms of exponentially many DCUT inequalities that ensure solution connectivity and additionally prevent subcycles. Thus, assume that $\mathcal{P}_d = \mathcal{P}_c \cap \{(2.14)\}$ denotes the set obtained from \mathcal{P} by replacing constraint (2.9) by the following constraints:

$$\sum_{(i,j) \in \delta^+(S)} z_{ij} \geq h_v, \quad v \in S, S \subset V \setminus \{r\}, 2 \leq |S| \quad (2.14)$$

DCUT inequalities (2.14) enforce that for any vertex subset $S \subset V \setminus \{r\}$, if a vertex $v \in S$ is a *cluster-head*, there must exist *backbone* arcs pointing out from S . This condition applies since $r \notin S$, otherwise, p-ASP solutions would not be connected. Thus, a directed cutset p-ASP formulation is given by:

$$\min \left\{ \sum_{(i,j) \in A} c_{ij} y_{ij} + \sum_{(i,j) \in A} c_{ij} z_{ij} : (h, y, z) \in \mathcal{P}_d \cap (\mathbb{B}^{n+1} \times \mathbb{B}^m \times \mathbb{B}^m) \right\} \quad (2.15)$$

In what follows, given any polyhedron \mathcal{U} obtained from a MIP formulation for p-ASP, by relaxing the integrality constraints, we denote by $v(\mathcal{U})$ the corresponding LP relaxation lower bound. Under the spanning arborescence representation, as the next result indicates, formulations \mathcal{P}_f and \mathcal{P}_d are equivalent.

Proposition 2.1.1. $v(\mathcal{P}_f) = v(\mathcal{P}_d)$

Proof. It is well known, from the *maximum flow* and *minimum cut* theorem [Magnanti and Wolsey, 1995], that the capacity of the directed cut determined by a subset $S \subset V$ bounds the flow sent from a vertex $v \in S$ to the root $r \in V \setminus S$. Thus, $\sum_{(i,j) \in \delta^+(S)} z_{ij} \geq h_v$ is valid for all $v \in S$ if and only if the constraint set (2.10)-(2.12) has a feasible solution. \square

2.1.3 Strengthening the formulations

Note that, in formulation \mathcal{P}_f , the total flow for all commodities is bounded by the total demand, which is p due to constraint (2.3). This suggests a simple strengthening of the aggregate version of *forcing constraints* (2.11) that typically bounds the flow on the arcs to at most the cardinality of the vertex set. Thus, the formulation can be strengthened with the following valid inequalities:

$$\sum_{q \in Q} f_{ij}^q \leq pz_{ij}, \quad (i, j) \in A. \quad (2.16)$$

In the remainder, denote by $\mathcal{P}_F = \mathcal{P}_f \cap \{(2.16)\}$ the formulation obtained when appending (2.16) to \mathcal{P}_f . Clearly, we have $v(\mathcal{P}_F) \geq v(\mathcal{P}_f)$.

We may also reinforce \mathcal{P}_d with the following lifted version of DCUT inequalities (2.14):

$$\sum_{(i,j) \in \delta^+(S)} z_{ij} - \sum_{j \in \delta_v^+ \cap S} y_{vj} \geq h_v, \quad v \in S, S \subset V \setminus \{r\}, 2 \leq |S| \quad (2.17)$$

Inequalities (2.17) generalizes the GSEC constraints for MCDSP (see Gendron et al. [2014] for details), that ensures that any vertex $v \in V$ either belongs to S or has a neighbor vertex in S . To show that (2.17) is a valid inequality for p-ASP formulation, consider a vertex subset $S \subset V \setminus \{r\}$ and a vertex $v \in S$. Note that, to break circuits and also enforce solution connectivity, there must exist a *backbone* arc pointing out from S . This is true since the *sink* is the root of the *backbone*. Thus, we have two cases to consider: i) v is a *cluster-head* vertex in S , in which case (2.17) reduces to (2.14); ii) v is not a *cluster-head* but it may connect to a *cluster-head* $j \in S$, in which case, due to (2.2) and (2.5), inequalities (2.17) hold. After substituting (2.14) by (2.17), let us denote by \mathcal{P}_D the strengthened DCUT based formulation obtained. Since (2.17) dominate (2.14), we have $v(\mathcal{P}_D) \geq v(\mathcal{P}_d)$.

It is worth noting that inequalities (2.17) also enforce the linking constraint among variables y , z and h . Let us take the vertex subset $S = \{v_1, v_2\}$ and observe that $\sum_{(i,j) \in \delta^+(S)} z_{ij} = \sum_{j \in \delta_{v_1}^+} z_{v_1j} + \sum_{j \in \delta_{v_2}^+} z_{v_2j} - z_{v_1v_2} - z_{v_2v_1} \geq h_{v_1} + y_{v_1v_2}$. Now, since $\sum_{j \in \delta_{v_1}^+} z_{v_1j} = h_{v_1}$ and $\sum_{j \in \delta_{v_2}^+} z_{v_2j} = h_{v_2}$ we have $h_{v_1} + h_{v_2} - z_{v_1v_2} - z_{v_2v_1} \geq h_{v_1} + y_{v_1v_2}$ or $(z_{ij} + z_{ji}) + y_{ij} \leq h_j$. This shows that the linking constraints (2.4) are implied by (2.17) when $|S| = 2$. It should be noted that inequalities (2.14) and (2.17) are valid for all subsets $S \subset V \setminus \{r\}$ not containing the *sink* and not only for those of cardinality at most p . But, whenever the solution being separated is integer, due to the cardinality constraint (2.3), we can identify DCUTs in an economical way and append to the model only those with $2 \leq |S| \leq p$.

2.2 Algorithms

The most important details of our branch-and-cut (BC) algorithms are described in this section. We refer to Padberg and Rinaldi [1991] and Wolsey and Nemhauser [1999] for a detailed description of branch-and-cut solution methods. Here, three BC algorithms for p -ASP are given: BCF-T, BCF-C and BCD. BCF-T and BCF-C are based on formulation \mathcal{P}_F , whereas BCD is based on \mathcal{P}_D . The CPLEX optimization package is used to solve linear programs and also to manage the enumeration trees. The BC algorithms are initialized with a valid upper bound provided by the Benders-based heuristic, described in Section 2.2.3.

2.2.1 Branch-and-cut algorithm based on \mathcal{P}_F

In this section, we present the branch-and-cut algorithms based on formulation \mathcal{P}_F . Motivated by the observation that the main drawback when solving \mathcal{P}_F by means of branch-and-bound (BB) algorithms is due to the large number of forcing constraints (2.11) and (2.16), the obvious alternative is to relax these constraints and add them in a dynamic way within a cutting-plane algorithm. By doing so, one may take advantage of the cutting-plane tools available in a general purpose MIP solver. The benefits of such approach are also described in [Gendron and Larose, 2014; Chouman et al., 2016].

The separation problems for (2.11) and (2.16) are trivial. Given an LP solution $(\bar{h}, \bar{y}, \bar{z}, \bar{f})$, it is sufficient to check if $\bar{f}_{ij}^q - \bar{z}_{ij} > 0$ for each pair of commodity $q \in Q$ and arc $(i, j) \in A$ to identify all violated inequalities (2.11). If $\sum_{q \in Q} \bar{f}_{ij}^q > p\bar{z}_{ij}$ for any arc $(i, j) \in A$ a valid inequality (2.16) is identified. When no violated inequalities are found for both families, we have computed the LP relaxation lower bound.

The algorithms BCF-T and BCF-C differ in the way of handling constraints (2.11) and (2.16) in a cutting-plane fashion. These algorithms try to find the best tradeoff between the quality of the lower bounds and the CPU time consumed.

BCF-T separates inequalities (2.11) for fractional LP solutions at every node, but instead of appending to the model all the violated ones, it only adds those for which the violation exceeds a given threshold α , i.e., $\bar{f}_{ij}^q - \bar{z}_{ij} > \alpha$. Aside from the root node of the BC enumeration tree, inequalities (2.16) are only separated when an integer solution is defined, which ensures that a feasible solution is obtained.

BCF-C, on the other hand, is a direct implementation from a state of the art MIP solver (we use CPLEX 12.6), where a cut-pool with all inequalities (2.11) and (2.16) is provided beforehand to the solver. It is up to the solver to decide when and in which node of the enumeration tree the cuts are added.

For the two algorithms, priority was set for branching on h -variables first. Indeed, when branching on an h -variable, we are “*lifting*” the corresponding *flow conservation* equations (2.10). In this way, *forcing constraints* are also lifted and a very small proportion of the cuts are needed.

2.2.2 Branch-and-cut algorithm based on \mathcal{P}_D

BCD is the branch-and-cut algorithm based on \mathcal{P}_D . At the root node of its enumeration tree, lifted DCUT inequalities (2.17) are separated by means of a fractional separation routine. In the remainder of the nodes, the separation routine is only invoked when an integer solution is found and $2 \leq |S| \leq p$, which is much easier than separating fractional solutions. To enforce solution connectivity of integer solutions only DCUT inequalities (2.14) are separated. We also experimented separating fractional solutions throughout the enumeration tree, but this version was outperformed by our approach.

Let us assume that the LP relaxation is computed to optimality, where $\bar{z} \in [0, 1]^m$, $\bar{y} \in [0, 1]^m$ and $\bar{h} \in [0, 1]^{n+1}$ denote its optimal solution. If $(\bar{z}, \bar{y}, \bar{h})$ is integer and its corresponding support digraph is connected and circuit-free, then $(\bar{z}, \bar{y}, \bar{h})$ is an optimal solution to p-ASP. Otherwise, valid inequalities as (2.17) may be violated by $(\bar{z}, \bar{y}, \bar{h})$ and consequently should be appended into the model.

The exact separation for DCUT inequalities (2.14) can be efficiently carried out by means of solving a minimum cut problem defined over the support digraph $\bar{D} = (\bar{V}, \bar{A})$, for $\bar{A} = \{(i, j) \in A : \bar{z}_{ij} > 0\}$ and arc capacities $\{\bar{z}_{ij} : (i, j) \in \bar{A}\}$. Based on the *maximum flow* and *minimum cut* theorem, Dinic’s algorithm is used for defining the corresponding mincut set. Thus, let $S \subset V \setminus \{r\}$ be the vertex subset defining the minimum cut $(S, V \setminus S)$ in \bar{D} . A DCUT inequality is violated whenever the capacity of

its corresponding cut, i.e., $\sum_{(ij) \in (S, V \setminus S)} \bar{z}_{ij}$, is less than \bar{h}_v , for $v \in S$. We adapted this procedure to also separate lifted DCUT. A lifted DCUT inequality (2.17) is violated whenever $\sum_{(ij) \in (S, V \setminus S)} \bar{z}_{ij}$ is less than $\bar{h}_v + \sum_{j \in \delta_v^+ \cap S} \bar{y}_{vj}$, for $v \in S$. The most violated lifted DCUT (2.17) identified is always added to the model. The others are appended only if they are orthogonal enough with the most violated one; otherwise, they are forgotten. An inequality is said to be orthogonal enough if the inner product (evaluated after normalizing the inequality with Euclidean norm) between that inequality and the most violated one does not exceed a given factor ε , that was set to 0.7 in our algorithms. As stated above, whenever $(\bar{z}, \bar{y}, \bar{h})$ is integer, we identify DCUTs in an economical way by means of a *depth-first search* algorithm that identifies connected components over \bar{D} .

Likewise for its counterparts, BCF-T and BCF-C, in BCD, priority was also set for branching on h -variables.

2.2.3 Benders-based heuristic

In this section, we introduce the CB-heuristic used to initialize the upper bound for our exact BC algorithms. At this point, one may note that, given a feasible *cluster-head* set W , the problem becomes easy to solve when fixing h -variables corresponding to that set. If W is connected, a spanning arborescence (ST_w) can be computed by means of a polynomial algorithm [Edmonds, 1967]. Additionally, the problem of defining a cluster (K^i, A_K^i) for all $i \in W$, spanning vertices not in W , is a simple assignment problem (AP_w), which can be solved efficiently by only comparing the cost c_{ji} for each pair $i \in W \setminus \{r\}$ and $j \notin W$. Alongside with the cardinality constraint (2.3), the CB-heuristic exploits that property in a combinatorial Benders-like fashion. Basically, the heuristic alternates between two problems: a master problem and a subproblem. The master problem defines a *cluster-head* set W , while the subproblem attests the feasibility of the *cluster-head* set provided by the master, i.e., it verifies the connectivity of the *backbone* and the coverage of vertices in $V \setminus W$.

Assume the following master problem

$$(M) \quad \min \{ \theta : (h, \theta) \in \mathcal{H} \cap (\mathbb{B}^{n+1} \times \mathbb{R}) \} \quad (2.18)$$

where \mathcal{H} is defined by (2.3), (2.6) and $\{\theta \geq 0\}$. The subproblem consists in checking whether or not the corresponding solution h^w , obtained when solving (M), leads to a feasible solution for ST_w and AP_w , where $W = \{i \in V : h_i^w = 1\}$. In case of infeasibility,

we add a *feasibility cut* (2.19) to (M) :

$$\sum_{i \in V: h_i^w = 1} h_i \leq p. \quad (2.19)$$

Whenever h^w is feasible, we add an *optimality cut* (2.20) to (M) :

$$\theta^s \left(\sum_{i|h_i^w=0} h_i \right) + \theta \geq v^s, \quad (2.20)$$

where v^s is the optimal value for the current subproblem and $\theta^s = v^s - v^l$, where v^l is a lower bound on the optimal value of the problem v^* . Since the optimal value (θ) of the master problem (M) is a lower bound on v^* , inequalities (2.20) become active only when $h=h^w$ and then $\theta = v^s$, i.e., h^w is optimal. Otherwise, since the master problem minimizes θ , at least one h value is modified and the constraint becomes redundant. Inequality (2.20) acts exactly as the classical Benders optimality cut. These cuts have been introduced in the context of stochastic integer programs [Laporte and Louveaux, 1993].

The initial lower bound v^l is computed by solving $v(\mathcal{P}_c)$. Given its corresponding LP optimal value for h -variables, the CB-heuristic selects the p vertices with higher LP value as the first *cluster-head* candidate set W . Based on the feasibility of ST_w and AP_w , *feasibility* or *optimality* cuts are generated. The CB-heuristic is then given by rounds of solving the master problem and identifying *feasibility* (2.19) and *optimality* (2.20) cuts.

A drawback of the CB-heuristic is given by the fact that the bounds provided are weak and since there is nothing more in the master problem than the cardinality constraint on h -variables, one may enumerate all possible combinations of values for h and, consequently, generate a huge number of *feasibility* cuts. In fact, this is an exact procedure, which we use as heuristic given its limitations: we simply run the procedure until a given time limit is reached.

2.3 Computational results

In this section, we assess the quality of the LP lower bounds given by our formulations as well as the efficiency of the exact solution algorithms BCF-T, BCF-C and BCD. All these algorithms are initialized with a feasible solution and a cut-off value provided by

the CB-heuristic performed for 5 minutes. In our experiments the threshold parameter α for BCF-T was set to 0.4.

Computational experiments have been performed on instances with $|V| \in [30, 200]$ and different ranges for the digraph density. Some instances considered in the current study were adapted from STS instances introduced in [Bardossy and Raghavan, 2010]. These are Euclidean instances with vertices randomly located on a 100×100 square grid. To adapt a STS instance to p-ASP, we set the value of p equal to the cardinality of *cluster-heads* set, pick vertex 0 as the *sink* and for each edge $e = \{i, j\}$ of a STS instance, with cost c_e , we define two arcs with cost $c_{i,j} = c_{j,i} = c_e$. We consider STS instances with 100 vertices, $|V|= 100$, and $p \in \{10, 20, 30, 40, 50, 60\}$, where $p < n$. Additional instances are also introduced here, as we described next.

The proposed p-ASP instances are randomly generated as follows. Given a source instance defined as a complete digraph $H = (V, A_H)$ with costs $\{c_{i,j} > 0 : (i, j) \in A\}$, we fix $p \in \{4, 5, 8, 10, 15, 20, 30\}$ and $n \in \{29, 50, 75, 98, 100, 126, 151, 179, 199\}$, such that $|V| = n + 1$. We randomly choose with uniform probability a subset $W \subset V$ containing p cluster-heads, such that $r \in W$, $|W| = p + 1$ and $D_W = (W, A_W)$ is a connected subdigraph. Then, we assign each vertex $i \notin W$ to a cluster-head $j \in W \setminus \{r\}$ by enlarging A_W with the new complementary arc $a = (i, j)$. At that point, we have a random solution for p-ASP. Let \hat{A} denote the set of arcs in that solution. We then initialize: $A \leftarrow \hat{A}$. Additional arcs are then randomly picked from $A_H \setminus \hat{A}$ and added to A , until a desired graph density $d \in \{30\%, 40\%, 50\%, 60\%, 70\%, 100\%\}$ is obtained. Each instance name clearly indicates the corresponding values for n, p and d . To illustrate, for instance $n200p10d70$, $n = 199$, $p = 10$ and $d = 70\%$. The proposed instance set is made available for download at http://www.dcc.ufmg.br/~vwcmorais/p_asp_instances/.

The computational results reported were obtained with an Intel[®] XEON E5645 Core[™] i7-980 hexa-core machine, running at 2.4GHz, with 24 GB of shared RAM memory. The algorithms were implemented in C++ and compiled with g++. IBM LOG CPLEX concert libraries (version Optimization Studio 12.6) were used in our implementation. All algorithms were executed with only one core; no multi-threading was allowed. All the remaining CPLEX parameters are kept at default values for cut generation, pre-processing procedures and management of the enumeration tree.

2.3.1 LP relaxation lower bounds

In our first experiment, we evaluate the LP relaxation lower bound provided by each formulation presented in Section 2.1. In order to compute the bounds, all CPLEX

parameters for cut generation and pre-processing procedures were turned off and α set to 0. The experiments are conducted with an instance subset with $29 \leq n \leq 100$. The results are reported in Table 2.1. The first two columns of the table provide the instance name and the corresponding optimal objective function value (OPT). The next three columns show, respectively, the lower bound value $v(\mathcal{P}_f)$, the CPU time spent to compute that bound followed by the relative gap $((\text{OPT} - v(\mathcal{P}_f))/\text{OPT})$. In the other columns, we present the LP relaxation lower bound for the other formulations, \mathcal{P}_F , \mathcal{P}_d and \mathcal{P}_D , as well as the gap and CPU time to compute the bound. To compute the bound for \mathcal{P}_d and for its strengthened version \mathcal{P}_D , the separation routines of BCD are used. It can be seen that the bounds given by \mathcal{P}_F are slightly better than those provided by \mathcal{P}_f . In general, the evaluation of $v(\mathcal{P}_F)$ is more time consuming than that of $v(\mathcal{P}_f)$. As stated in Proposition 2.1.1, \mathcal{P}_f and \mathcal{P}_d are equivalent, i.e., the same bounds are provided for both formulations. However, due to the large number of violated forcing inequalities (2.11) added to the model, along with the additional set of flow variables, computing $v(\mathcal{P}_f)$ is more expensive than computing $v(\mathcal{P}_d)$. In general, among all formulations considered here, \mathcal{P}_D is the strongest one. This shows that the lifted DCUTs (2.17) significantly improve the bound.

Instance	OPT	Linear programming lower bounds											
		$v(\mathcal{P}_f)$	$t(s)$	$g(\%)$	$v(\mathcal{P}_F)$	$t(s)$	$g(\%)$	$v(\mathcal{P}_d)$	$t(s)$	$g(\%)$	$v(\mathcal{P}_D)$	$t(s)$	$g(\%)$
n30p4d60	555	540.22	1.28	2.66	540.33	5.94	2.64	540.22	0.04	2.66	551.17	0.03	0.69
n30p5d100	387	379.29	4.58	1.99	379.33	12.86	1.98	379.29	0.10	1.99	387.00	0.05	0.00
n30p8d50	472	463.50	0.86	1.80	463.50	5.07	1.80	463.50	0.07	1.80	472.00	0.04	0.00
n30p8d60	404	395.00	2.40	2.23	395.00	4.87	2.23	395.00	0.04	2.23	400.75	0.05	0.80
n51p5d100	640	628.99	39.87	1.72	629.37	438.12	1.66	628.99	0.88	1.72	633.59	1.16	1.00
n51p8d30	854	831.12	2.73	2.68	831.34	43.79	2.65	831.12	0.08	2.68	836.01	0.07	2.11
n51p8d40	763	750.32	4.90	1.66	750.32	68.93	1.66	750.32	0.48	1.66	753.96	0.19	1.18
n51p10d100	499	477.68	65.03	4.27	477.91	394.04	4.23	477.68	0.90	4.27	486.66	1.43	2.47
n76p5d100	934	912.41	793.78	2.31	915.39	9111.92	1.99	912.41	11.52	2.31	923.69	10.59	1.10
n76p10d50	895	876.43	263.22	2.07	877.26	1875.80	1.98	876.43	9.16	2.07	883.35	5.24	1.30
n76p10d60	833	815.24	683.65	2.13	815.31	6457.66	2.12	815.24	7.36	2.13	824.30	5.46	1.04
n76p10d70	787	770.38	231.89	2.11	770.57	3563.08	2.09	770.38	3.78	2.11	776.19	2.02	1.37
n99p5d50	3719	3428.99	209.68	7.80	3432.84	1764.11	7.69	3428.99	5.29	7.80	3449.33	4.98	7.25
n99p5d60	3582	3279.34	308.63	8.45	3284.39	1086.90	8.31	3279.34	4.21	8.45	3297.79	3.88	7.93
n99p5d100	2678	2631.88	3468.94	1.72	2637.99	17955.20	1.49	2631.88	45.18	1.72	2665.56	93.19	0.46
n99p10d50	2617	2472.23	928.03	5.53	2475.47	13354.00	5.41	2472.23	18.27	5.53	2512.30	15.36	4.00
n101p5d50	1817	1651.88	295.15	9.09	1652.74	15908.70	9.04	1651.88	6.14	9.09	1658.21	1.78	8.74
n101p5d60	1511	1429.75	341.88	5.38	1431.54	729.85	5.26	1429.75	12.88	5.38	1439.15	6.75	4.76
n101p10d40	1336	1243.66	525.31	6.91	1244.88	7167.84	6.82	1243.66	8.72	6.91	1256.47	20.38	5.95
n101p10d50	1176	1112.67	951.77	5.39	1112.93	11401.90	5.36	1112.67	7.88	5.39	1122.38	3.91	4.56

Table 2.1: Linear programming lower bounds for p-ASP formulations.

2.3.2 Comparison of Branch-and-cut algorithms

In the second set experiments, we assess the efficiency of the exact solution algorithms described here. Detailed computational results for BCF-C, BCF-T and BCD are presented in Tables 2.2, 2.3 and 2.4.

Table 2.2 contains results for instances adapted from Bardossy and Raghavan [2010] aggregated by p (i.e., the cardinality of *facilities* set). In this table, we report the value of p , the number of arcs ($|A|$) and the number of instances ($\#NS$) on each instance subset, and for each algorithm we show: the number of instances solved to optimality ($\#NC$) in the set, the average CPU total time (T), the average number of nodes in the branch-and-bound tree ($\#N$), and the average relative gap ($g(\%)$) ($UB - LB)/UB$), computed at the end of the execution of each algorithm, where UB is upper bound and LB is the average lower bound. Averages are taken over all instances in the instance set. Specifically, these instances turned out easier to solve with our algorithms. The overall best performance is obtained by BCD (followed by BCF-T), that is enable to solve to optimality all the instances considered. In most of the instances, the optimal solution is found at the root node in the branch-and-bound tree (often within a few seconds). Detailed computational results with that instance set for BCF-C, BCF-T and BCD are given in supplementary tables available in the Appendix A.

Instance set			BCF-C			BCF-T			BCD					
p	$ A $	$\#NS$	$\#NC$	T	$\#N$	$g(\%)$	$\#NC$	T	$\#N$	$g(\%)$	$\#NC$	T	$\#N$	$g(\%)$
10	1890	80	80	37.5	0.3	0.00	80	24.2	0.1	0.00	80	5.1	0.0	0.00
20	3580	80	80	2600.0	5.8	0.00	80	1616.5	5.2	0.00	80	145.1	2.9	0.00
30	5070	36	36	6930.1	5.2	0.00	34	6400.4	4.2	0.01	36	44.6	0.7	0.00
40	6360	21	21	7565.9	1.6	0.00	21	5230.5	1.5	0.00	21	41.8	0.3	0.00
50	7450	15	13	8182.9	6.1	0.04	15	5928.1	5.9	0.00	15	67.7	0.6	0.00
60	8340	67	66	6371.3	1.4	0.01	67	4325.8	1.5	0.00	67	303.9	0.1	0.00

Table 2.2: Comparison of the aggregated computational results for BCF-C, BCF-T and BCD when solving STS instances [Bardossy and Raghavan, 2010] adapted to p-ASP.

Tables 2.3 and 2.4 contain results for the proposed p-ASP instances. The column headings are: **Name**, the instance name; **CB**, the initial upper bound given by CB-heuristic; \mathbf{v}^+ , the best upper bound found among all algorithms; $\mathbf{h}(\%)$, the relative gap $((CB - \mathbf{v}^+)/CB)$ between \mathbf{v}^+ and the initial primal solution; **BUB**, the best upper bound computed over the course of a given algorithm; **BLB**, the best lower bound; $\mathbf{g}(\%)$, the corresponding duality gap $((BUB - BLB)/BUB)$; $\mathbf{\#N}$, the total number of nodes investigated in the BB enumeration tree; t_r , the CPU time (in seconds) at the

root node; t_a , the overall CPU total time. Whenever an instance was not solved to proven optimality within a time limit of 5 hours, an indication “-” is given.

Instance				BCF-C					BCF-T					BCD				
Name	CB	v^+	$h(\%)$	BUB	BLB	$g(\%)$	#nodes	t_r/t_a	BUB	BLB	$g(\%)$	#nodes	t_r/t_a	BUB	BLB	$g(\%)$	#nodes	t_r/t_a
n30p4d50	626	626	0.0	626	626.0	0.0	2	0.2 / 0.4	626	626.0	0.0	2	2.1 / 2.3	626	626.0	0.0	1	0.0 / 0.1
n30p4d60	578	555	4.0	555	555.0	0.0	7	0.1 / 0.3	555	555.0	0.0	3	1.5 / 1.7	555	555.0	0.0	1	0.0 / 0.1
n30p4d70	486	472	2.9	472	472.0	0.0	3	0.1 / 0.3	472	472.0	0.0	3	0.5 / 0.7	472	472.0	0.0	1	0.0 / 0.0
n30p5d100	395	387	2.0	387	387.0	0.0	15	0.6 / 2.7	387	387.0	0.0	12	12.3 / 12.8	387	387.0	0.0	1	0.1 / 0.1
n30p8d50	532	472	11.3	472	472.0	0.0	42	0.3 / 1.2	472	472.0	0.0	16	2.4 / 3.2	472	472.0	0.0	1	0.0 / 0.0
n30p8d60	478	404	15.5	404	404.0	0.0	50	0.7 / 28.5	404	404.0	0.0	43	3.8 / 5.6	404	404.0	0.0	3	0.1 / 0.1
n30p8d70	429	372	13.3	372	372.0	0.0	1172	2.5 / 229.7	372	372.0	0.0	280	6.8 / 16.8	372	372.0	0.0	15	0.1 / 0.3
n51p4d40	1309	1106	15.5	1106	1106.0	0.0	23	0.8 / 6.9	1106	1106.0	0.0	22	2.8 / 6.6	1106	1106.0	0.0	29	0.1 / 0.8
n51p4d60	971	895	7.8	895	895.0	0.0	24	1.7 / 6.6	895	895.0	0.0	29	7.3 / 10.2	895	895.0	0.0	33	0.2 / 0.9
n51p4d70	929	754	18.8	754	754.0	0.0	2	0.9 / 1.5	754	754.0	0.0	2	5.2 / 5.6	754	754.0	0.0	1	0.1 / 0.2
n51p5d100	664	640	3.6	640	640.0	0.0	20	2.8 / 7.4	640	640.0	0.0	16	0.9 / 7.2	640	640.0	0.0	12	1.2 / 2.5
n51p8d30	1051	854	18.7	854	854.0	0.0	31	1.6 / 8.0	854	854.0	0.0	23	7.0 / 9.4	854	854.0	0.0	15	0.1 / 0.5
n51p8d40	770	763	0.9	763	763.0	0.0	22	1.8 / 12.4	763	763.0	0.0	12	15.1 / 16.7	763	763.0	0.0	8	0.2 / 0.7
n51p8d50	748	699	6.6	699	699.0	0.0	23	2.3 / 11.3	699	699.0	0.0	23	8.8 / 10.0	699	699.0	0.0	30	0.1 / 0.6
n51p10d100	537	499	7.1	499	499.0	0.0	948	4.0 / 3956.7	499	499.0	0.0	660	116.8 / 389.9	499	499.0	0.0	181	1.4 / 7.1
n76p5d60	1207	1176	2.6	1176	1176.0	0.0	40	12.4 / 140.3	1176	1176.0	0.0	40	44.1 / 72.6	1176	1176.0	0.0	61	3.3 / 12.9
n76p5d70	1297	1111	14.3	1111	1111.0	0.0	27	12.7 / 107.1	1111	1111.0	0.0	22	67.2 / 89.0	1111	1111.0	0.0	55	2.1 / 7.7
n76p5d100	1012	934	7.7	934	934.0	0.0	67	13.5 / 190.3	934	934.0	0.0	24	133.5 / 171.7	934	934.0	0.0	46	10.6 / 24.6
n76p10d50	945	895	5.3	895	895.0	0.0	40	14.0 / 498.2	895	895.0	0.0	28	121.6 / 154.9	895	895.0	0.0	24	5.2 / 7.6
n76p10d60	873	833	4.6	833	833.0	0.0	57	14.1 / 326.1	833	833.0	0.0	49	176.9 / 218.0	833	833.0	0.0	27	5.5 / 9.1
n76p10d70	893	787	11.9	787	787.0	0.0	50	12.4 / 157.3	787	787.0	0.0	47	120.0 / 131.9	787	787.0	0.0	55	2.0 / 7.6
n99p5d50	4678	3719	20.5	3719	3719.0	0.0	163	14.5 / 421.4	3719	3719.0	0.0	164	339.7 / 565.5	3719	3719.0	0.0	232	5.0 / 42.8
n99p5d60	4354	3582	17.7	3582	3582.0	0.0	208	27.3 / 862.0	3582	3582.0	0.0	208	567.6 / 1301.0	3582	3582.0	0.0	498	3.9 / 83.6
n99p5d100	3021	2678	11.4	2678	2678.0	0.0	96	19.0 / 355.2	2678	2678.0	0.0	33	1049.4 / 1098.2	2678	2678.0	0.0	14	93.2 / 111.4
n99p10d50	2779	2617	5.8	2809	2589.7	7.8	166	14.3 / -	2617	2617.0	0.0	407	978.2 / 7305.6	2617	2617.0	0.0	381	15.4 / 84.5
n99p10d60	2562	2517	1.8	2517	2517.0	0.0	508	16.0 / 15610.7	2517	2517.0	0.0	552	422.6 / 1219.5	2517	2517.0	0.0	336	25.7 / 107.7
n99p10d70	2402	2220	7.6	2220	2220.0	0.0	143	20.0 / 11616.3	2220	2220.0	0.0	190	605.2 / 1745.9	2220	2220.0	0.0	226	43.5 / 110.5
n101p5d50	2123	1817	14.4	1817	1817.0	0.0	307	16.2 / 1050.4	1817	1817.0	0.0	338	3.2 / 424.8	1817	1817.0	0.0	499	1.8 / 67.5
n101p5d60	1795	1511	15.8	1511	1511.0	0.0	47	19.6 / 134.1	1511	1511.0	0.0	43	440.2 / 555.3	1511	1511.0	0.0	142	6.8 / 42.7
n101p5d100	1348	1177	12.7	1177	1177.0	0.0	54	17.8 / 163.1	1177	1177.0	0.0	61	13.5 / 280.3	1177	1177.0	0.0	19	35.3 / 50.0
n101p10d40	1495	1336	10.6	1341	1329.0	0.9	450	14.3 / -	1336	1336.0	0.0	668	523.6 / 4772.6	1336	1336.0	0.0	798	20.4 / 176.4
n101p10d50	1241	1176	5.2	1176	1176.0	0.0	624	14.2 / 5538.5	1176	1176.0	0.0	468	879.7 / 5233.5	1176	1176.0	0.0	615	3.9 / 88.3
n101p10d60	1328	1100	17.2	1134	1088.7	4.0	403	15.7 / -	1119	1088.6	2.7	504	409.4 / -	1100	1100.0	0.0	1220	5.3 / 152.2
n101p10d70	1045	1005	3.8	1012	1004.0	0.8	262	20.1 / -	1005	1005.0	0.0	356	1446.4 / 3327.8	1005	1005.0	0.0	47	70.0 / 111.5

Table 2.3: Detailed computational results: BCF-C, BCF-T, and BCD when solving p -ASP instances with $29 \leq n \leq 100$.

Instance				BCF-C					BCF-T					BCD				
Name	CB	v ⁺	h(%)	BUB	BLB	g(%)	#nodes	t _r /t _a	BUB	BLB	g(%)	#nodes	t _r /t _a	BUB	BLB	g(%)	#nodes	t _r /t _a
n127p10d50	260015	236637	9.0	236637	236637.0	0.0	21	18.8/88.8	236637	236637.0	0.0	20	299.1/416.5	236637	236637.0	0.0	27	5.9/13.2
n127p10d60	252912	228766	9.5	228766	227699.0	0.5	184	22.0/-	228766	228766.0	0.0	445	1791.5/2552.7	228766	228766.0	0.0	663	5.7/90.9
n127p10d70	232963	203322	12.7	203322	203322.0	0.0	74	24.8/5282.3	203322	203322.0	0.0	77	3369.2/3707.9	203322	203322.0	0.0	165	13.2/64.3
n127p10d100	224833	176740	21.4	176740	176740.0	0.0	35	23.1/229.4	176740	176740.0	0.0	34	465.2/660.1	176740	176740.0	0.0	25	11.5/24.4
n127p15d40	223244	212200	4.9	223244	206102.0	7.7	150	20.3/-	212200	212200.0	0.0	187	851.9/1572.2	212200	212200.0	0.0	218	2.2/25.9
n127p15d50	224738	201391	10.4	201391	201391.0	0.0	277	19.5/17594.8	201391	201391.0	0.0	279	1708.1/4704.5	201391	201391.0	0.0	332	7.3/48.0
n127p15d60	182392	179339	1.7	179339	179339.0	0.0	156	19.1/3563.3	179339	179339.0	0.0	133	468.3/1166.2	179339	179339.0	0.0	60	20.5/33.5
n127p15d70	179511	170419	5.1	178731	169475.0	5.2	183	20.1/-	170419	170419.0	0.0	702	3178.9/8406.0	170419	170419.0	0.0	433	35.9/137.2
n127p20d40	199012	192822	3.1	192822	192822.0	0.0	31	26.6/125.9	192822	192822.0	0.0	35	1292.6/1439.9	192822	192822.0	0.0	50	5.6/13.0
n127p20d50	186082	173547	6.7	175881	172180.0	2.1	96	22.1/-	173547	173547.0	0.0	481	1963.6/6763.7	173547	173547.0	0.0	212	23.8/68.2
n127p20d70	159377	148547	6.8	159377	145462.0	8.7	122	20.1/-	148547	148547.0	0.0	199	3961.3/9348.4	148547	148547.0	0.0	48	32.4/61.6
n152p15d40	256089	219134	14.4	294133	207936.0	29.3	241	26.2/-	224267	216447.0	3.5	4080	8696.5/-	219134	219134.0	0.0	4637	236.4/4259.3
n152p15d60	193541	163976	15.3	193541	145044.0	25.1	118	27.2/-	213697	155242.0	27.4	489	8656.0/-	163976	163976.0	0.0	3028	1062.5/6017.1
n152p15d70	172625	143867	16.7	172625	128549.0	25.5	142	28.4/-	152613	137557.0	9.9	2580	6781.4/-	143867	143867.0	0.0	70	2976.0/3772.9
n152p20d30	300977	232151	22.9	300977	207399.0	31.1	123	19.7/-	235753	225793.0	4.2	2392	6954.2/-	232151	232151.0	0.0	8263	147.6/4381.3
n152p20d40	234796	181375	22.8	234796	158540.0	32.5	329	35.4/-	268510	167851.0	37.5	290	8511.2/-	181375	181375.0	0.0	6518	444.5/6624.0
n152p20d70	127293	112631	11.5	127293	95622.4	24.9	121	27.8/-	116029	102506.0	11.7	1863	7947.5/-	112631	112631.0	0.0	103	6672.7/7661.5
n152p30d30	208477	177131	15.0	208477	160867.0	22.8	101	20.8/-	183805	168071.0	8.6	605	3226.9/-	177131	177131.0	0.0	1019	266.0/671.6
n180p5d60	282910	205784	27.3	210671	188056.0	10.7	166	65.3/-	205784	194049.0	5.7	847	44.3/-	205784	205784.0	0.0	5877	81.9/5438.5
n180p5d70	229681	173597	24.4	179197	164793.0	8.0	186	66.8/-	179197	164544.0	8.2	167	36.5/-	173597	173597.0	0.0	3847	72.0/5408.5
n180p10d50	168912	116841	30.8	131484	106170.0	19.3	136	41.5/-	120392	109651.0	8.9	901	11865.2/-	116841	116841.0	0.0	9042	449.7/14364.0
n180p10d70	100395	90532	9.8	100395	77901.1	22.4	155	49.0/-	100395	-	-	-	-/-	90532	90532.0	0.0	4538	701.4/14241.6
n180p15d60	94059	81152	13.7	94059	66532.8	29.3	76	38.8/-	94059	-	-	-	-/-	81152	75892.4	6.5	5289	749.8/-
n200p10d50	190426	146314	23.2	190426	121059.0	36.4	118	62.4/-	190426	-	-	-	-/-	146314	136542.0	6.7	10043	354.5/-
n200p10d70	155915	110072	29.4	155915	90293.3	42.1	103	82.3/-	155915	-	-	-	-/-	110072	101070.0	8.2	4439	932.8/-
n200p15d50	176278	102864	41.6	176278	85274.9	51.6	95	50.1/-	176278	-	-	-	-/-	102864	95837.3	6.8	6701	605.5/-
n200p20d40	139706	101934	27.0	139706	84555.7	39.5	114	40.4/-	139706	84555.7	39.5	2	17858.3/-	101934	95886.4	5.9	12201	650.2/-

Table 2.4: Detailed computational results: BCF-T, BCF-C and BCD when solving p-ASP instances with $126 \leq n \leq 199$.

Table 2.3 reports the results for instances with $29 \leq n \leq 100$, i.e, the *small* instances. Comparing only the algorithms based on the multicommodity flow formulations, it is important to note that, in general, BCF-C is faster at the root node than BCF-T. The reason for this comes from the way the MIP solver is handling the cuts. BCF-C stops prematurely the separation for valid inequalities (2.11) and (2.16). BCF-T, on the other hand, attempts to find a better compromise between quality of the lower bounds and CPU time consumed when handling violated forcing inequalities (2.11) for fractional solutions. Consequently, BCF-C is expected to enumerate more BC nodes than BCF-T. Indeed, BCF-T starts the enumeration with a smaller optimality gap in most instances. As result, we can see from Table 2.3 that 30 out of 34 instances were solved by BCF-C and 33 by BCF-T. Still, for the *small* instances, our computational results suggest that BCD is likely to attain better results than BCF-C and BCF-T. Overall, thanks to the quality of the LP relaxation lower bound provided by \mathcal{P}_D and the CPU time needed to compute that bound, BCD outperforms its counterparts in terms of CPU time spent and number of instances solved (the 34 instances were solved by this algorithm to proven optimality). Finally, note that the upper bounds indicated in CB are usually close to v^+ .

Table 2.4 reports the results for instances with $126 \leq n \leq 199$, i.e, the *large* instances. It can be observed from Table 2.4 that computing the LP relaxation $v(\mathcal{P}_F)$ is a very difficult task with large values of n . Unlike the smallest instances in that set, BCF-T was not able to compute the LP relaxation for several instances with n bigger than 180. The other algorithms performed well. In general, BCF-C was the first algorithm in terms of CPU time at the root node. However, in the course of the search, BCF-C solves models involving a huge number of *forcing constraints*. At this point, the picture turns in favour of BCF-T. BCF-T solved 11 out of 27 instances to optimality, while BCF-C closed only 6 of the instances evaluated. Additionally, for instances not solved to optimality, BCF-C manages to improve the initial upper bound, but fails closing the optimality gap. BCF-T, on the other hand, slightly improves the initial primal bound provided by the heuristic for the largest instances, in some cases that value remains the same as in CB. Overall, out of the three exact solution algorithms investigated, BCD still remains the best algorithm: BCD solved to optimality 22 instances and for the instances left unsolved provided smaller optimality gaps.

To conclude, it is interesting to remark that during our experiments, we found out that some sparse instances with the same value of n and d were more difficult to solve than others. Take for example the instances $n127p10d60$ and $n127p15d60$. This can be explained by the fact that when locating a very small number of *cluster-heads* it becomes very difficult to maintain the connectivity of *backbone tree* and to ensure

the coverage of non-*cluster-head* vertices. Thus, it can be pointed out that the main challenge of p-ASP is to define the optimal location of *cluster-heads*.

2.4 Concluding remarks

In this chapter, we investigated exact solution approaches for solving p-ASP. The main aspects of p-ASP are motivated by the design of a hierarchical ad-hoc wireless sensor network when the vertices (sensors) are organized in groups (clusters), selecting for each group a cluster-head as the leader and defining a topological structure to disseminate information to a predefined sink node. We placed the problem in the literature and differentiated its application from the classic problems of determining tree-based network topologies, like tree-star and cable-trench problems. We considered two formulation approaches for the problem. First, a compact formulation involving additional continuous variables over a multicommodity directed flow model to enforce the connectivity, while in the second approach, connectivity is ensured by means of an exponential number of circuit breaking constraints. In addition to the formulations, we also introduced a Benders-based heuristic and three exact solution algorithms: BCF-T, BCF-C and BCD. BCF-T and BCF-C are based on the compact flow-based formulation while BCD is based on circuit breaking constraints. The algorithms differ mainly in the way they handle constraints that ensure the topology connectivity. BCF-C and BCF-T rely on standard implementation of a cutting-plane algorithm. BCD separates lifted directed cutset inequalities only for fractional solutions at the root node and integer solutions in the remainder of the enumeration tree.

An extensive computational study was performed on two set of instances: a set of randomly generated instances proposed in this paper and a subset of instances adapted from Steiner tree-star (STS) benchmarks. Assuredly, the instances adapted from STS turned out to be easily solvable (very often solved within a few seconds using our algorithms). In each of the instance sets, our computational experiments have demonstrated that algorithm BCD outperformed BCF-C and BCF-T. Additionally, the results showed the superiority of the directed cutset based formulation over the multicommodity flow formulation. They also validated the idea of simplifying the separation routines in favour of evaluating more BB nodes.

Future works include a branch-and-price-and-cut algorithm based on set covering reformulation. We also intend to proceed with the characterization of p-ASP valid inequalities, such as capacity constraints that benefit from the cardinality constraint, to separate them into our algorithms. Another direction is to extend our formulations

and algorithms to other related problems such as the *median p -cycle problem*, where a cycle connecting p *cluster-heads* would play the role of *backbone* instead of a spanning arborescence.

Chapter 3

The p -cycle star problem

This Chapter is dedicated to the *p-cycle star problem*. We present mixed integer formulations for the problem. First, a compact formulation based on a directed multi-commodity flow model is given. This formulation is then strengthened with additional valid inequalities. With the resulting formulations on hands, we derive cutset based inequalities by projecting out the continuous flow variables. All of this is described in Section 3.1. Additionally, a branch-and-bound and two non-standard branch-and-cut algorithms based on the proposed formulations are given in Section 3.2. Comments over the computational results appear in Section 3.3. We close the chapter in Section 3.4.

3.1 p-CSP formulations

To present p-CSP formulations, we introduce some notation that will be used throughout the chapter. Denote by $\delta^+(S) = \{(i, j) \in A : i \in S, j \notin S\}$ the set of arcs pointing out from the vertex subset $S \subset V$ and by $\delta^-(S) = \{(i, j) \in A : i \notin S, j \in S\}$ the set of arcs pointing into the vertex subset S . For simplicity, we use δ_i^+ and δ_i^- instead of $\delta^+(\{i\})$ and $\delta^-(\{i\})$ when referring to the vertex subset $S = \{i\}$. Given a p-CSP formulation \mathcal{U} , denote by $v(\mathcal{U})$ its linear programming (LP) relaxation lower bound and by \hat{x} the corresponding LP optimal value for any variable x used in \mathcal{U} . Additionally, assume that $\text{proj}_x(\mathcal{U})$ is the projection of \mathcal{U} onto the space of x variables.

The formulations presented in this chapter are based on the fact that a directed cycle may be decomposed into a set of paths from every vertex of the cycle to a predefined one. Accordingly, p-CSP asks for paths from each cluster-head to the sink. Based on that, a multicommodity flow formulation and directed cutset inequalities are used to model the directed cycle backbone, named here *pcorecycle*.

Let $Q = \{q \in V \setminus \{r\}\}$ be the set of commodities, sharing the same destination r (the *sink*), each one having a single origin $q \in V \setminus \{r\}$. Along the sets defined above, p-CSP formulations use the following variables: $\{h_i \in \mathbb{B} = \{0, 1\} : i \in V \setminus \{r\}\}$, to identify the selected cluster-head set W and $h_r = 1$; $\{z_{ij} \in \mathbb{B} : (i, j) \in A\}$, to identify the arcs in the *pcorecycle*; $\{y_{ij} \in \mathbb{B} : (i, j) \in A\}$ to identify the *intra-cluster* arcs. Accordingly, if $(i, j) \in A$ is in the *pcorecycle* ($z_{ij} = 1$) then both of its endpoints must be chosen as cluster-heads ($h_i = 1$ and $h_j = 1$) or *sink* ($h_r = 1, i = r$ or $j = r$). Likewise, whenever a vertex $i \in V \setminus \{r\}$ is covered by a cluster-head $j \in W$ we have that $y_{ij} = 1, h_i = 0$ and $h_j = 1$. Additionally, the formulations also involve continuous flow variables $\{f_{ij}^q \in R_+ : (i, j) \in A, q \in Q\}$, representing the fraction of flow on each arc (i, j) for each commodity $q \in Q$.

Alongside with variables defined above, define \mathcal{F} as a compact set given by $\{f_{ij}^q \geq 0, (i, j) \in A, q \in Q\}$, $\{0 \leq h_i \leq 1, i \in V\}$, $\{z_{ij} \geq 0, (i, j) \in A\}$, $\{y_{ij} \geq 0, (i, j) \in A\}$ and:

$$h_i + \sum_{j \in \delta_i^+} y_{ij} = 1, \quad \forall i \in V, \quad (3.1)$$

$$\sum_{i \in V \setminus \{r\}} h_i = p, \quad (3.2)$$

$$h_r = 1, \quad (3.3)$$

$$\sum_{j \in V \setminus \{r\}} (y_{rj} + y_{jr}) = 0, \quad (3.4)$$

$$z_{ij} + z_{ji} + y_{ij} \leq h_j, \quad \forall (i, j) \in A, \quad (3.5)$$

$$\sum_{j \in \delta^+(i)} z_{ij} = h_i \quad \forall i \in V, \quad (3.6)$$

$$\sum_{j \in \delta^-(i)} z_{ji} = h_i, \quad \forall i \in V, \quad (3.7)$$

$$\sum_{(i,j) \in \delta_i^+} f_{ij}^q - \sum_{(j,i) \in \delta_i^-} f_{ji}^q = \begin{cases} 0 & \text{if } i \neq q, \\ h_i & \text{if } i = q, \end{cases} \quad i \in V \setminus \{r\}, q \in Q, \quad (3.8)$$

$$f_{ij}^q \leq z_{ij}, \quad \forall (i, j) \in A, \forall q \in Q. \quad (3.9)$$

A compact formulation for p-CSP is:

$$\min \left\{ \sum_{(i,j) \in A} c_{ij} z_{ij} + \sum_{(i,j) \in A} d_{ij} y_{ij} : (h, y, z, f) \in \mathcal{F} \cap \hat{O} \right\}, \quad (3.10)$$

where $\hat{O} = (\mathbb{B}^{n+1} \times \mathbb{B}^{|A|} \times \mathbb{B}^{|A|} \times \mathbb{R}^{|A||Q|})$.

Constraints (3.1) determine whether or not an *intra-cluster* arc $(i, j) \in A$ appears in a feasible cluster. If $h_i = 0$, vertex i is left out of the cluster-head set. In this case, there must exist a cluster-head j , such that $h_j = 1$, and an *intra-cluster* arc pointing outward from i towards j therefore exists, i.e., $y_{ij} = 1$ holds. Whenever $h_i = 1$, no *intra-cluster* arc pointing out of it exists. In this case, $y_{ij} = 0$ must apply for all $j \in \delta_i^+$. Constraints (3.1) also guarantee that only one cluster must be centered at a cluster-head $i \in W$.

Equality (3.2), imposes the cardinality of the cluster-head set. Constraint (3.3) guarantees that the *sink* must be in *pcorecycle*. By the WSN application requirements, the mobile *sink* only visits cluster-heads, a connection *sensor-sink* is not allowed. This is enforced by constraint (3.4).

Constraints (3.5) play the role of *linking* constraints among variables y , z and h . If arc $(i, j) \in A$ is in a *pcorecycle*, the same cannot be selected as an *intra-cluster* arc. Consequently, in the former case, both of its endpoints must be cluster-heads. It is worth noting that inequalities (3.5) are not valid when $p = 1$, i.e. cycles of size 2 are not accepted. Let us take vertex $i \in V$ as single cluster-head, thus $h_i = 1$ and $h_r = 1$. Note that $(z_{ir} + z_{ri}) + y_{ir} \leq h_r$ or $(z_{ri} + z_{ir}) + y_{ri} \leq h_i$ does not hold. Therefore, the strong linking constraints are valid for p-CSP only because $p > 1$.

Constraints (3.6) enforce that exactly one arc of *pcorecycle* must be pointing out from a selected cluster-head. By its turn, constraints (3.7) guarantee that exactly one arc of *pcorecycle* must be pointing into a selected cluster-head. Together, (3.6) and (3.7) are referred to as the *flow conservation* constraints since they also impose the equality between in and out degree on any vertex of the *pcorecycle*.

The *three-index flow conservation* equations (3.8) ensure that every cluster-head must send one unit of flow to the *sink*. The forcing constraints (3.9) impose the capacity on the flow traversing an arc member of *pcorecycle*. Together, (3.6)-(3.9) and $\{f_{ij}^q \geq 0, (i, j) \in A, q \in Q\}$ define a directed path from each cluster-head to the *sink*. Note that we still need to keep equalities (3.7) in the model, even whether constraints (3.6) and (3.8) are present, that because along with (3.6), (3.7) also enforces that the in and out degrees on each cluster-head are equal. Finally, the objective function (3.10) minimizes *intra* and *inter-cluster* total cost.

3.1.1 Additional valid inequalities

p-CSP formulations can be strengthened with the addition of valid inequalities. Note that, in formulation \mathcal{F} , the total flow for all commodities is bounded by the number

of cluster-heads, which is p due to constraint (3.2). Thus, the formulation can be strengthened with the following valid inequalities:

$$\sum_{q \in Q} f_{ij}^q \leq pz_{ij}, \quad \forall (i, j) \in A. \quad (3.11)$$

Constraints (3.11) suggest a simple strengthening of the aggregate version of forcing constraints (3.9) that typically bounds the flow on the arcs to at most the cardinality of $V \setminus \{r\}$.

Note that due (3.2), (3.3), (3.6) and (3.7) equation (3.12) can be derived. This constraint is valid since the defined topology of a *pcorecycle* must have exactly $p+1$ arcs.

$$\sum_{i \in V} \sum_{j \in \delta_i^+} z_{ij} = p + 1 \quad (3.12)$$

One may also reinforce the formulations with additional logic based inequalities. For instance, inequalities (3.13) impose that there are no directed paths with two arcs within the same *cluster*. Inequality (3.14) enforces that if an arc belongs to *pcorecycle* both of its endpoints must be cluster-heads. Finally, inequality (3.15) states that if a vertex is not an endpoint of arcs in *pcorecycle* at least one of its *close neighbors* must be a cluster-head.

$$y_{ik} + y_{kj} \leq 1, \quad i, j, k \in V \setminus \{r\}, \quad (3.13)$$

$$2z_{ij} \leq h_i + h_j, \quad (i, j) \in A, \quad (3.14)$$

$$-\sum_{j \in \delta^+(i)} z_{ij} - \sum_{j: (i,j) \in \delta_i^+} h_j \leq -1, \quad i \in V. \quad (3.15)$$

Equality (3.12) is redundant for the LP relaxation of p-CSP since it is obtained from a linear combination of constraints (3.2), (3.3), (3.6) and (3.7). Note that due to the strong *linking* constraints (3.5), inequalities (3.14) are also redundant. Although these inequalities are redundant, we observed that their addition generally improves the computing times.

In the remainder, denote by \mathcal{F}_u the intersection of \mathcal{F} with (3.11) - (3.15). Clearly, we have $v(\mathcal{F}_u) \geq v(\mathcal{F})$.

3.1.2 Projecting out the flow variables

The main drawback when solving \mathcal{F}_u by means of branch-and-bound (BB) algorithms is due to the large number of flow variables and constraints (3.9) and (3.11). It is well known that by projecting out the flow variables of a multicommodity flow formulation, one can derive cutset based inequalities from the extreme rays of the corresponding projection cone [Magnanti and Wolsey, 1995]. These are the projection inequalities (or *Benders feasibility cuts*). As formulation \mathcal{F}_u provides good bounds, but computed with a high computational cost, we follow the method described in [Maculan, 1987] to project out the flow variables and to introduce Benders like inequalities keeping the quality of the LP bounds.

Let \mathcal{P} denote the polytope defined over the variable set $\{h, z, f\}$ given by constraints (3.8), (3.9), $\{f_{ij}^q \geq 0, (i, j) \in A, q \in Q\}$, $\{h_i \geq 0, i \in V\}$, $\{z_{ij} \geq 0, (i, j) \in A\}$ and (3.11). To project out the flow variables, suppose that the (h, z) -variables are fixed and consider the following dual of the minimization problem defined over \mathcal{P} :

$$(DP) \quad \max \sum_{q \in Q} \hat{h}_q u_q^q - \sum_{q \in Q} \sum_{(ij) \in A} w_{ij}^q \hat{z}_{ij} - p \sum_{(ij) \in A} \pi_{ij} \hat{z}_{ij} \quad (3.16)$$

$$u_i^q - u_j^q - w_{ij}^q - \pi_{ij} \leq 0, \quad (i, j) \in A, i \neq r, q \in Q, \quad (3.17)$$

$$-u_j^q - w_{rj}^q - \pi_{rj} \leq 0, \quad j \in V_r, q \in Q, \quad (3.18)$$

where $\{u_i^q : i, q \in Q\}$, $\{w_{ij}^q \geq 0 : q \in Q, (i, j) \in A\}$ and $\{\pi_{ij} \geq 0 : (i, j) \in A\}$ are respectively the dual variables associated with constraints (3.8), (3.9) and (3.11). DP can be either an unbounded or bounded problem. In the former case, there is an unboundedness direction (u, w, π) such that we may define the following projection inequality:

$$\sum_{q \in Q} \hat{h}_q u_q^q - \sum_{q \in Q} \sum_{(ij) \in A} w_{ij}^q \hat{z}_{ij} - p \sum_{(ij) \in A} \pi_{ij} \hat{z}_{ij} \leq 0. \quad (3.19)$$

We evaluated unboundedness cases for DP. In both cases, we identify an extreme ray of the polyhedral convex cone ending up with valid cut-set based inequalities.

Case 1: $\pi_{ij} = 0, \forall (i, j) \in A$. In this case, DP can be decomposed by commodities defining a DP_q problem for each $q \in Q$. If DP_q is unbounded for a given $q \in Q$ its

corresponding primal problem is infeasible. In this matter, it is not possible to send flow from q to r in the support digraph (\bar{V}, \bar{A}) defined by (\hat{h}, \hat{z}) , where $\bar{V} = \{i \in V : \hat{h}_i > 0\}$ and $\bar{A} = \{(i, j) \in A : \hat{z}_{ij} > 0\}$. Let $(S, \bar{S}) \in A$ be the cut defined over that digraph, such that $S \subset V \setminus \{r\}$, $\bar{S} = V \setminus S$ and $v \in S$. Assume that $u_i^q = 1$ if $i \in S$ and $u_i^q = 0$ otherwise, likewise $w_{ij}^q = \begin{cases} 0, & \text{if } i, j \in S \text{ or } i, j \in \bar{S} \\ 1 & \text{if } i \in S, j \in \bar{S} \text{ or } i \in \bar{S}, j \in S. \end{cases}$ Now, replacing u_i^q and w_{ij}^q in (3.19) by their corresponding value and aggregating over (S, \bar{S}) we have:

$$\sum_{(i,j) \in \delta^+(S)} z_{ij} \geq h_v, \quad v \in S, S \subset V \setminus \{r\}, 2 \leq |S| \leq p, \quad (3.20)$$

which is exactly the well defined direct cutset inequality (DCUT) [Magnanti and Wolsey, 1995]. This case is a well know result extended from [Maculan, 1987].

Case 2: $\exists(i, j) \in A : \pi_{ij} > 0$. In this case, whenever the system defining \mathcal{P} is unbounded when $h = \hat{h}$ and $z = \hat{z}$, we can generate an extreme ray of the referred projection cone as follows: (i) take a partition (S, \bar{S}) of V in the support digraph given by the solution (\hat{h}, \hat{z}) , such that $\bar{S} = V \setminus S$ and $r \in \bar{S}$; (ii) set $w_{ij}^q = 0$, $(i, j) \in A$, and (iii) $u_i^q = 1/p$ if $i \in S$ and $u_i^q = 0$ otherwise; (iv) likewise $\pi_{ij} = \begin{cases} 0, & \text{if } i, j \in S \text{ or } i, j \in \bar{S} \\ 1/p & \text{if } i \in S, j \in \bar{S} \text{ or } i \in \bar{S}, j \in S. \end{cases}$ Setting the values of u_i^q , π_{ij} and w_{ij}^q in (3.19) and aggregating it over the arcs $\{(i, j) \in A, i \in S, j \in \bar{S}\}$ we define a valid inequality:

$$\sum_{(i,j) \in \delta^+(S)} z_{ij} \geq 1/p \sum_{i \in S} h_i, \quad S \subset V \setminus \{r\}, p < |S|. \quad (3.21)$$

Inequalities (3.21) are violated by LP solutions with $p < |S|$ having a directed path spanning the vertices in S . Such a path is composed of more than $p-1$ arcs. Figure 3.1 illustrates the *pcorecycle* of a solution violated by inequalities (3.11) and (3.21).

Accordingly, we can now define an alternative formulation to \mathcal{F} . The so-called DCUT formulation \mathcal{D} is given by the intersection of (3.1)-(3.7), $\{1 \geq h_i \geq 0, i \in V\}$, $\{z_{ij} \geq 0, (i, j) \in A\}$, $\{y_{ij} \geq 0, (i, j) \in A\}$ and (3.20). We know that: $\text{proj}_{(h,z,y)}(\mathcal{F}) = \mathcal{D}$ [Magnanti and Wolsey, 1995]. Then, the p -CSP direct cutset based formulation is written as

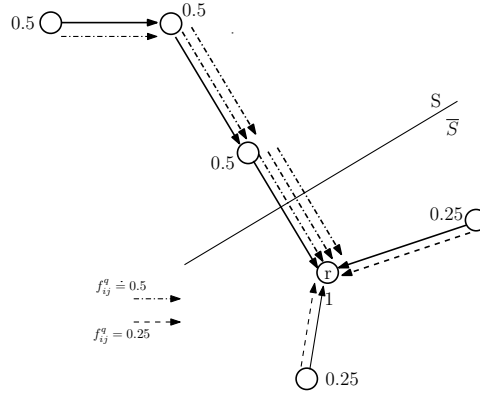


Figure 3.1: Illustration of the *pcorecycle* backbone for a solution with $p = 2$ that violates inequalities (3.11) and (3.21).

$$\min \left\{ \sum_{(i,j) \in A} c_{i,j} z_{ij} + \sum_{(ij) \in A} d_{ij} y_{ij} : (h, y, z) \in \mathcal{D} \cap \hat{H} \right\}, \quad (3.22)$$

where $\hat{H} = (\mathbb{B}^{n+1} \times \mathbb{B}^{|A|} \times \mathbb{B}^{|A|})$.

One may also reinforce \mathcal{D} with the following lifted version of DCUT inequalities (3.20):

$$\sum_{(i,j) \in \delta^+(S)} z_{ij} - \sum_{j \in \delta_v^+ \cap S} y_{vj} \geq h_v, \quad v \in S, S \subset V \setminus \{r\}, 2 \leq |S| \leq p \quad (3.23)$$

Inequalities (3.20) and (3.23) are the so-called breaking subcycle constraints. Given any vertex subset $S \subset V \setminus \{r\}$ and a cluster-head vertex $v \in S$, to break subcircuits and also enforce solution connectivity there must exist an arc of *pcorecycle* pointing out from S . This is true since the *sink* must be in *pcorecycle*. For cases where v is not a cluster-head, it may be connected to a cluster-head $j \in S$. In this case, the lifted inequalities (3.23) hold.

Finally, denote by \mathcal{D}_d the intersection of \mathcal{D} with (3.12)-(3.15) and (3.21). As well, assume that \mathcal{D}_d^+ denotes the intersection of \mathcal{D}_d with (3.23). As result we have: $v(\mathcal{D}_d^+) \geq v(\mathcal{D}_d) \geq v(\mathcal{D})$.

3.2 Algorithms

In this section, we present the most important details of our exact solution procedures for p -CSP. Precisely, a branch-and-bound (BB) and two branch-and-cut (BC) algorithms. The algorithms were implemented by means of the CPLEX optimization package. For a detailed description of BB and BC methods, we refer to Padberg and Rinaldi [1991] and Wolsey and Nemhauser [1999].

Next, we describe a primal heuristic and the separation routines used by our exact solution procedures.

3.2.1 Primal heuristic

The exact algorithms are initialized with a valid upper bound computed as follows. Let $C = V \setminus \{r\}$ and $W = \{r\}$ be respectively the initial candidate and cluster-head sets. Initially $pcorecycle$ is given by $r \longleftrightarrow r$, then, at each iteration, a candidate vertex $k \in C$ with smallest score l_k is removed from C and added in W until the desired cardinality $|W| = p + 1$ is reached, increasing $pcorecycle$ by an arc. With a feasible cluster-head set on hand we assign every other vertex in $V \setminus W$ to the cheapest cluster-head in $W \setminus \{r\}$. The score function l_k is: $l_k = (c_{ik} + c_{kj} - c_{ij}) + \sum_{j \in \delta_k^- \cap C} d_{kj}$, for every $k \in C$. The first term of that equation computes the insertion cost of a vertex $k \in C$ between every pair of adjacent vertices $\{i, j\}$ in the current $pcorecycle$, while the second term corresponds to an estimation of the *intra-cluster* cost if k is chosen as a cluster-head.

3.2.2 Separation routines

Let us assume that the LP relaxation is computed to optimality, and let $\hat{z} \in [0, 1]^m$, $\hat{y} \in [0, 1]^m$, $\hat{h} \in [0, 1]^{n+1}$ and $\hat{f} \in [0, 1]^{nm}$ denote the optimal solution. If $(\hat{h}, \hat{y}, \hat{z})$ is integer and its corresponding support digraph is connected and subcycle-free, then $(\hat{h}, \hat{y}, \hat{z})$ is an optimal solution to p -CSP. Otherwise, valid inequalities may be violated by $(\hat{h}, \hat{y}, \hat{z})$ and consequently should be appended into the model.

The separation of (3.9), (3.11), (3.13), (3.14) and (3.15) is trivial. Given an LP solution $(\hat{h}, \hat{y}, \hat{z})$, it is sufficient to check whether $\hat{f}_{ij}^q - \hat{z}_{ij} > 0$ for each pair of commodity $q \in Q$ and arc $(i, j) \in A$ to identify all violated inequalities (3.9). For any arc $(i, j) \in A$, if $\sum_{q \in Q} \hat{f}_{ij}^q > p\hat{z}_{ij}$ a violated inequality (3.11) is identified. Whereas if $2\hat{z}_{ij} \geq \hat{h}_i + \hat{h}_j$ then a violated inequality (3.14) is identified. Given any triple of vertices $i, k, j \in Q$, check whether $\hat{y}_{ik} + \hat{y}_{kj} \geq 1$ to find a violated inequality (3.13). Finally, we

check whether $\sum_{j \in \delta^+(i)} \hat{z}_{ij} + \sum_{j: (i,j) \in \delta_i^+ \cup \delta_i^-} \hat{h}_j \geq 1$ for each $i \in V$ to identify all violated inequalities (3.15).

The exact separation for DCUT inequalities (3.20) can be efficiently carried out by means of solving a minimum cut problem defined over the support digraph $\bar{D} = (\bar{V}, \bar{A})$, for $\bar{A} = \{(i, j) \in A : \hat{z}_{ij} > 0\}$ and arc capacities $\{\hat{z}_{ij} : (i, j) \in \bar{A}\}$. Based on the *maximum flow* and *minimum cut* theorem, Dinic's algorithm is used to define the corresponding mincut set. Thus, let $S \subset V \setminus \{r\}$ be the vertex subset defining the minimum cut $(S, V \setminus S)$ in \bar{D} . A DCUT inequality is violated whenever the capacity of its corresponding cut, i.e., $\sum_{(ij) \in (S, V \setminus S)} \hat{z}_{ij}$, is less than \hat{h}_v , for $v \in S$. We adapted this procedure to also separate valid inequalities (3.21) and lifted DCUT (3.23). An inequality (3.21) is violated whenever $\sum_{(ij) \in \delta^+(S)} \hat{z}_{ij} \leq 1/p \sum_{i \in S} \hat{h}_i$ for $S \subset V \setminus \{r\}$ and $p < |S|$. Likewise, a lifted DCUT inequality (3.23) is violated whenever $\sum_{(ij) \in (S, V \setminus S)} \hat{z}_{ij}$ is less than $\hat{h}_v + \sum_{j \in V_v \cap S} \hat{y}_{vj}$, for a $v \in S$. When no violated inequalities are found for all families, one have computed the LP relaxation lower bound.

3.2.3 Exact algorithms

BBF is the branch-and-bound algorithm based on \mathcal{F} . Since \mathcal{F} is a compact formulation, besides of embedding the corresponding p-CSP model into CPLEX, no significant implementation work was performed by us. For this reason, no additional details are provided for that algorithm.

The branch-and-cut algorithm BCF based on \mathcal{F}_u is motivated by the observation that the main drawback when solving the compact formulation \mathcal{F}_u is due to the huge number of capacity constraints (3.9), (3.11), (3.13) - (3.15). The obvious alternative to overcome that problem is to relax those constraints and add them in a dynamic way within a cutting-plane fashion. Accordingly, in BCF a cut-pool with all these inequalities are provided beforehand to the MIP solver. It is up CPLEX to manage of the cut-pool. Basically, it must check the violations of inequalities in that pool and decide when and in which node of the enumeration tree the cuts are added.

BCD is the branch-and-cut algorithm based on \mathcal{D}_d^+ implemented with calls to CPLEX `concert` libraries. At the root node of its enumeration tree, inequalities (3.20), (3.21) and (3.23) are separated by means of a fractional separation routine. In the remainder of the nodes, the separation routines for those inequalities are only invoked when an integer solution is defined. Like BCF, BCD also keeps a cut-pool with inequalities (3.13)-(3.15) handled by the solver.

Whenever a new LP relaxation $(\hat{z}, \hat{y}, \hat{h})$ of a given BCF or BCD node is evaluated, we compute a valid upper bound with an adapted version of the heuristic described

above. In order to provide a new primal solution, LP information is aggregated to the score equation. The score l_k is computed as $l_k = (c_{ik}\hat{z}_{ik} + c_{kj}\hat{z}_{kj} - c_{ij}\hat{z}_{ij})\hat{h}_k + \sum_{j \in V_k \cap C} d_{kj}\hat{y}_{kj}$, for every candidate vertex $k \in C$. A p -CSP solution is iteratively constructed in order to update the best upper bound.

The exact algorithms implement a best-bound enumeration search, embedded with callbacks for CPLEX `concert` libraries to implement separation routines, management of the cut-pool, search tree, heuristic frequency, and branching strategies. To all exact algorithms, we set priority to branch first on variable h and then on z .

3.3 Computational results

Computational experiments were conducted on an Intel[®] XEON E5645 Core[™] i7-980 hexa-core machine with 2.4GHz of clock and 24 GB of shared RAM, running under Linux operating system. BBF, BCF and BCD were implemented in C++ and compiled with g++. IBM ILOG CPLEX `concert` library (version Optimization Studio 12.6.2) was used in our implementation. All algorithms were executed with only one core; no multi-threading was allowed. Finally, in our experiments, we set a maximum time limit of 3 hours of CPU time.

The experiments were performed on an instance set based on TSP test problems from the TSPLIB library. We consider a desired integer $p \in \{5, 10, 15, 20\}$, a number of $n+1 \in \{51, 70, 99, 100, 127, 144, 152, 200\}$ vertices and a constant $\alpha \in \{3, 5, 8\}$. Let b_{ij} be the distance for each pair of vertices $i, j \in V$ in the TSPLIB file. We set *cycle cost* $c_{ij} = \lceil \alpha b_{ij} \rceil$ and assignment cost $d_{ij} = \lceil (10 - \alpha)b_{ij} \rceil$ for each arc $(i, j) \in A$. Each instance name ($cc_v_alpha_p$) clearly indicates the TSP file name (cc), the cardinality of the vertex set $v = |V|$ and the corresponding values for α and p . To illustrate, for instance $eil_51_3_5$, $v = |V| = 51$, $\alpha = 3$ and $p = 5$. The instance set is made available for download at http://www.dcc.ufmg.br/~vwcmorais/p_csp_instances/.

In Table 3.1 we present the lower bounds provided by the formulations presented in Section 3.1. The results were obtained from experiments conducted with an instance subset with $51 \leq |V| \leq 100$. In the table, whenever an instance was not solved within the time limit, an indication “-” is provided. The first column entries in the table indicate the instance name. The next columns indicate the lower bound value for a given formulation and the CPU time (in seconds) spent to compute that bound. Finally, the last column provides the corresponding optimal objective function value (**Opt**). As we can see, since \mathcal{F} and \mathcal{D} are equivalent those formulations provide the same bound. However, due to the large number of violated forcing inequalities (3.9) along with the

additional set of flow variables, computing $v(\mathcal{F})$ is more expensive than computing $v(\mathcal{D})$. In spite of improving the bounds when appending (3.11) to \mathcal{F} , we realized that the computational time to compute that new bound increases considerably. Note that for some instances it was not possible to compute $v(\mathcal{F}_u)$ within the time limit. Because of that, BBF is based on \mathcal{F} resigning the quality of the bound to gain in processing time. In our experiments, we tried to separate (3.21) together with the compact formulation \mathcal{F}_u but no violated cut was found. We conjecture that $\mathcal{F} \cap (3.11)$ and $\mathcal{D} \cap (3.21)$ provide the same LP bounds, but a proof and an exact separation routine for (3.21) remain to be given. The results also indicate that (3.23) dominates (3.21) since $v(\mathcal{D} \cap \{(3.12) - (3.15)\} \cap (3.23)) \geq v(\mathcal{D}_d)$ for all instances evaluated. To conclude, note that \mathcal{D}_d^+ is the strongest among all formulations considered here.

Linear programming lower bounds													
Instance	$v(\mathcal{F})$	$t(s)$	$v(\mathcal{D})$	$t(s)$	$v(\mathcal{F}_u)$	$t(s)$	$v(\mathcal{D}_d)$	$t(s)$	$v(\mathcal{D} \cap \{(3.12) - (3.15)\} \cap (3.23))$	$t(s)$	$v(\mathcal{D}_d^+)$	$t(s)$	Opt
eil_51_3_5	4213.7	47.5	4213.7	0.2	4215.9	3451.6	4214.0	0.7	4225.4	0.7	4225.4	0.7	4253.0
eil_51_5_5	3310.5	50.2	3310.5	0.2	3320.7	3091.2	3310.5	0.6	3338.7	0.7	3338.7	0.7	3421.0
eil_51_8_5	1717.2	60.1	1717.2	0.2	1737.3	3591.6	1731.5	0.6	1915.3	1.7	1916.1	2.1	1990.0
eil_51_3_10	2982.8	106.5	2982.8	0.2	2983.8	3538.8	2982.8	0.5	3013.8	0.7	3013.8	0.7	3060.0
eil_51_5_10	2574.9	100.6	2574.9	0.2	2576.2	5320.8	2575.0	0.5	2606.6	0.6	2606.6	0.6	2719.0
eil_51_8_10	1661.2	259.0	1661.2	0.3	1672.3	5752.0	1665.0	0.6	1846.0	1.7	1852.2	1.6	1872.0
eil_51_3_15	2524.0	129.8	2524.0	0.2	2525.3	7654.5	2524.3	0.5	2565.3	0.6	2567.4	0.7	2608.0
eil_51_5_15	2304.6	236.3	2304.6	0.2	2305.7	6445.9	2304.6	0.5	2379.8	0.9	2387.3	0.8	2461.0
eil_51_8_15	1748.8	206.3	1748.8	0.2	1749.4	3591.7	1748.8	0.5	1886.9	0.9	1889.1	1.0	1898.0
eil_51_3_20	2223.1	190.8	2223.1	0.3	2223.3	2325.1	2223.1	0.5	2263.9	0.8	2264.0	1.0	2287.0
eil_51_5_20	2183.1	227.5	2183.1	0.2	2183.1	1460.1	2183.1	0.5	2270.4	1.1	2271.2	1.1	2292.0
eil_51_8_20	1893.7	180.6	1893.7	0.2	1894.5	1466.7	1893.7	0.4	1990.7	0.8	2001.7	0.7	2011.0
st_70_3_5	8257.5	343.1	8257.5	0.5	8268.5	3560.4	8258.3	1.2	8278.4	1.3	8278.4	1.3	8306.0
st_70_5_5	6426.6	1193.4	6426.6	1.1	6484.1	5096.3	6453.8	2.5	6564.7	3.7	6582.6	3.0	6612.0
st_70_8_5	3213.7	2868.5	3213.7	2.8	3410.6	6930.3	3338.2	4.9	3682.1	8.9	3709.3	7.2	3902.0
st_70_3_10	5633.7	1660.1	5633.7	0.8	5637.4	4053.5	5634.0	1.9	5655.0	2.4	5655.0	2.3	5655.0
st_70_3_15	4490.7	1830.2	4490.7	0.8	4506.4	5036.8	4500.1	2.1	4542.8	2.6	4547.0	2.2	4547.0
st_70_3_20	3828.3	2803.2	3828.3	1.2	-	-	3834.3	2.3	3901.4	3.4	3905.2	2.9	3912.0
st_70_5_20	3592.8	3136.3	3592.8	1.7	-	-	3607.0	3.2	3808.9	9.4	3816.9	8.0	3820.0
st_70_8_20	2695.8	2453.3	2695.8	1.1	-	-	2724.4	2.2	3286.7	12.4	3287.0	12.1	3287.0
eil_76_3_5	6224.7	152.5	6224.7	0.5	6224.7	3590.9	6224.7	1.4	6224.7	1.4	6232.9	1.4	6259.0
eil_76_5_5	4804.1	269.4	4804.1	1.0	4816.4	5959.1	4807.2	2.0	4827.5	2.1	4827.6	2.1	4891.0
eil_76_8_5	2326.0	865.1	2326.0	0.9	2412.7	8121.5	2374.6	3.8	2586.9	7.4	2637.4	11.7	2722.0
eil_76_3_10	4486.7	2267.1	4486.7	3.2	4503.9	8288.4	4498.3	3.3	4556.8	6.0	4557.0	5.7	4557.0
eil_76_8_10	1977.8	1757.4	1977.8	3.6	-	-	2006.8	4.3	2361.5	16.2	2370.5	19.5	2468.0
eil_76_8_15	1963.5	3173.9	1963.5	1.9	-	-	1991.1	4.4	2315.0	29.0	2339.2	28.5	2384.0
eil_76_8_20	2035.7	1390.0	2035.7	2.3	-	-	2048.3	4.4	2332.0	23.6	2339.1	30.0	2345.0
rat_99_3_5	18100.0	2762.2	18100.0	3.3	-	-	18127.9	6.4	18314.0	9.6	18314.0	10.2	18314.0
rat_99_3_10	12536.0	3589.8	12536.0	3.0	-	-	12548.7	4.9	12649.0	10.9	12649.0	10.4	12649.0
kroD_100_3_5	349514.5	144.9	349514.5	1.4	-	-	349514.5	3.6	349559.0	3.7	349559.0	3.6	349559.0
kroD_100_3_10	215398.0	554.0	215398.0	1.4	-	-	215398.0	3.5	215398.0	3.4	215398.0	3.5	215398.0

Table 3.1: Linear programming lower bounds for p-CSP formulations.

In Table 3.2 we compare the aggregated results for BBF, BCF and BCD. Instances are grouped in terms of their number of vertices and values of α . The three first columns of the table indicate: the cardinality of vertex set ($|V|$), the value of α and the number of instances in the group (**#inst**). For each algorithm we provide: the number of instances on a given group solved within the imposed CPU time limit (**#solved**), the average number of nodes in the branch-and-bound enumeration tree (applies to all instances in the group) (**#A.nodes**), the average of duality gap (**A-gap**) and the fastest time to solve an instance among all instances with the same number of vertices and α (**faster**). Detailed computational results can be found in the Appendix A.

$ V $	α	#inst	BBF				BCF				BCD			
			#solved	#A.nodes	A-gap	faster	#solved	#A.nodes	A-gap	faster	#solved	#A.nodes	A-gap	faster
51	3	4	2	123.0	0.000	886.5	4	558.3	0.000	7.7	4	327.8	0.000	2.6
	5	4	1	115.3	0.028	1689.5	3	2513.8	0.007	13	4	540.3	0.000	4.1
	8	4	3	330.3	0.016	3152.2	4	3290.0	0.000	97.1	4	49.3	0.000	1.4
70	3	4	2	9.5	0.255	4178.6	4	163.5	0.000	21.6	4	13.8	0.000	2.0
	5	4	0	6.3	0.422	-	3	1051.3	0.009	37.6	4	12.3	0.000	4.1
	8	4	0	8.8	0.352	-	1	9762.0	0.164	3247.6	4	747.0	0.000	13.9
76	3	4	1	3.0	0.356	2676.9	3	1332.0	0.000	26	4	1689.3	0.000	2.1
	5	4	0	1.5	0.416	-	2	2094.0	0.039	27.4	4	2645.0	0.000	7.7
	8	4	0	6.0	0.321	-	1	7628.5	0.151	975.9	4	2062.0	0.000	37.2
99	3	4	0	-	-	-	4	295.3	0.000	241.2	4	14.0	0.000	9.8
	5	4	0	-	-	-	2	2102.0	0.020	805.4	3	7377.0	0.002	71.5
	8	4	0	-	-	-	0	7018.3	0.246	-	1	4228.0	0.021	1527.0
100	3	4	1	3.0	0.000	12021.7	3	380.5	0.006	52.4	4	4.8	0.000	3.7
	5	4	0	-	-	-	2	1669.8	0.055	69.4	4	9.0	0.000	7.0
	8	4	0	-	-	-	0	8284.3	0.249	-	2	9706.5	0.013	9555.7
127	3	4	0	-	-	-	3	174.0	0.004	203.1	4	23.3	0.000	9.7
	5	4	0	-	-	-	2	1003.3	0.033	163	3	4312.8	0.000	18.5
	8	4	0	-	-	-	1	4593.3	0.195	7445.2	1	3647.5	0.045	2621.5
144	3	4	0	-	-	-	2	143.8	0.009	562.9	4	0.0	0.000	10.9
	5	4	0	-	-	-	2	907.3	0.027	3114.2	4	21.0	0.000	45.0
	8	4	0	-	-	-	0	7530.8	0.320	-	0	1292.8	0.042	-
152	3	4	0	-	-	-	1	636.3	0.057	1105.1	3	3286.8	0.000	19.2
	5	4	0	-	-	-	1	740.8	0.164	1911.2	1	1971.8	0.017	62.1
	8	4	0	-	-	-	0	4664.8	0.375	-	1	1166.3	0.106	4592.2
200	3	8	0	-	-	-	5	62.0	0.089	1249.1	8	15.6	0.000	45.2
	5	8	0	-	-	-	1	337.0	0.258	5730.2	6	93.0	0.002	115.3
	8	8	0	-	-	-	0	839.9	0.284	-	0	251.4	0.082	-

Table 3.2: Comparison of the aggregated computational results for BBF, BCF and BCD when solving p-CSP instances.

The results suggest that the most difficult instances are those with a higher value of α , i.e, instances where *cycle cost* is more costly than assignment cost. One may also expect that the instances become more difficult to solve with the increase of p , but we remark that this is not always true. This is due to the fact that when locating a very small number of cluster-heads it becomes very difficult to maintain the connectivity. This indicates that the main challenge of p-CSP is to define the *pcorecycle*.

In general, it can be observed that BCD outperformed BCF and BBF. As shown in Table 3.1, computing the LP relaxation $v(\mathcal{F})$ is a very difficult task with the increase of n and p . Consequently, BBF performs very poorly. Unlike the small instances, BBF was not able to compute the LP relaxation for no more than one instance with $|V|$ bigger than 99. The other algorithms perform quite well. BCD and BCF always improve the initial primal bound provided by the heuristic, while for BBF that value remains the same for $|V|$ bigger than 51. Measured by the number of BB nodes evaluated, BCF was the first algorithm while BCD the second. BBF, in its turn, spent too much time solving each BB node, consequently, fewer nodes are opened by that algorithm. To count, BCD solves 89 out of 120 instances evaluated to proven optimality, while its counterpart BCF and BBF solve respectively 54 and 10 instances.

It is worth mentioning that BCD and BCF outperform BBF due to the way those algorithms handle constraints (3.9), (3.11), (3.13) - (3.15). BBF manages to solve a compact formulation involving a huge number of flow variables and *forcing constraints* at every node of its enumeration tree. On the other hand, BCF handles such constraints in a cutting-plane fashion. In its turn, BCD is a non-standard cutting-plane method, in the sense that fractional points are only separated at the root node of its enumeration tree. These different approaches result in a huge variation of solution quality and CPU time. Consequently, BCF and BCD procedures manage to compute the LP bounds on each BB node in a very efficient way translating into more effective exact algorithms.

On the WSN application side, the clustering scheme modeled as p -CSP has a drawback. In particular, any given cluster-head may be overloaded since the models do not restrict the assignments of vertices (sensors) to cluster-heads. One possible alternative to overcome such a problem is to impose an upper bound on the total assignment cost. The problem with this approach is how to define such tight upper bound in order to keep feasibility. To avoid measuring such value one may impose a penalty $\{\theta \in R_+\}$ on the cluster with higher assignment cost. This could be done by adding valid inequalities imposing an upper bound on the assignment cost to at most the penalty. Ultimately, we observed that thanks to that penalty, the bounds provided by the resulting formulation can be very poor. Thus, an avenue of research would be to investigate optimality cuts to be handled into a cutting-plane framework.

3.4 Concluding remarks

In this Chapter we have presented formulations and exact solution procedures for the *p-cycle star problem* (p-CSP). We first introduced a compact multicommodity directed flow formulation strengthened with additional inequalities. After that, we derived cutset based formulations by projecting out the continuous flow variables. In this scheme, we identified two classes of valid inequalities that are separated into branch-and-cut algorithms. Computational results obtained with a branch-and-bound algorithm based on the compact flow formulation were compared with branch-and-cut algorithms based on both flow and cutset based formulations. In light of our computational study, we can conclude that the algorithm based on cutset models achieves better results than the algorithms based on the multicommodity flow formulations. We also can point out that the main challenge of p-CSP is to define the optimal location of cluster-heads and enforce connectivity of the core simple cycle.

As far as future developments go, it will be interesting to test a Lagrangian relaxation approach based on the cutset based formulation. To compute the bounds one could make use of a relax-and-cut (RC) method. RC would dualize two set of constraints: equalities (3.1) in a static way and inequalities (3.23) in a dynamic fashion. In order to provide the tightest lower bound, subgradient method would be used to approximately solve the Lagrangian dual problem. In this direction, two different BC algorithms may be designed: delayed relax-and-cut (DRC) and non-delayed relax-and-cut (NDRC). The main difference between DRC and NDRC would be basically the point where the relaxed inequalities are separated and dualized: after computing the LP or at every subgradient iteration.

Chapter 4

Configuration-based approach for topological problems in the design of wireless sensor networks

In this Chapter, we investigate the application of a mathematical programming technique based on configurations to p-ASP and p-CSP. In the most essential respects, we decompose the topology structure of the problems in such a way to obtain strong mathematical formulations. Unfortunately, the configuration-based models developed here are not tractable with a standard branch-and-bound approach. The evaluation of bounds provided by those formulations can however be tackled with *column-and-row* generation (CRG) techniques, also introduced here. Based on the formal description of the referred topological WSN problems, in Section 1.3, we introduce the configuration-based reformulations in Section 4.1. In Section 4.2, the CRG algorithms are described. Our computational experiments are presented and discussed in Section 4.3. Finally, the conclusion and future directions are drawn in the last section.

4.1 Configuration-based formulations

Under the cardinality constrained topology representation of p-ASP and p-CSP, each feasible solution for the problems can be decomposed into p -stars and a backbone connecting p cluster-heads and the sink. Thus, a common thread of the problems is to select the core cluster-head set that together with the sink defines a *configuration*. Formally, a configuration is a vertex subset W of cardinality $p + 1$, where W is given by the selected p cluster-heads and the predefined sink. Note that for both problems we

seek for exactly one *connected configuration*. Figure 4.1 illustrates a decomposition example of a configuration with the sink and $p = 4$ cluster-heads.

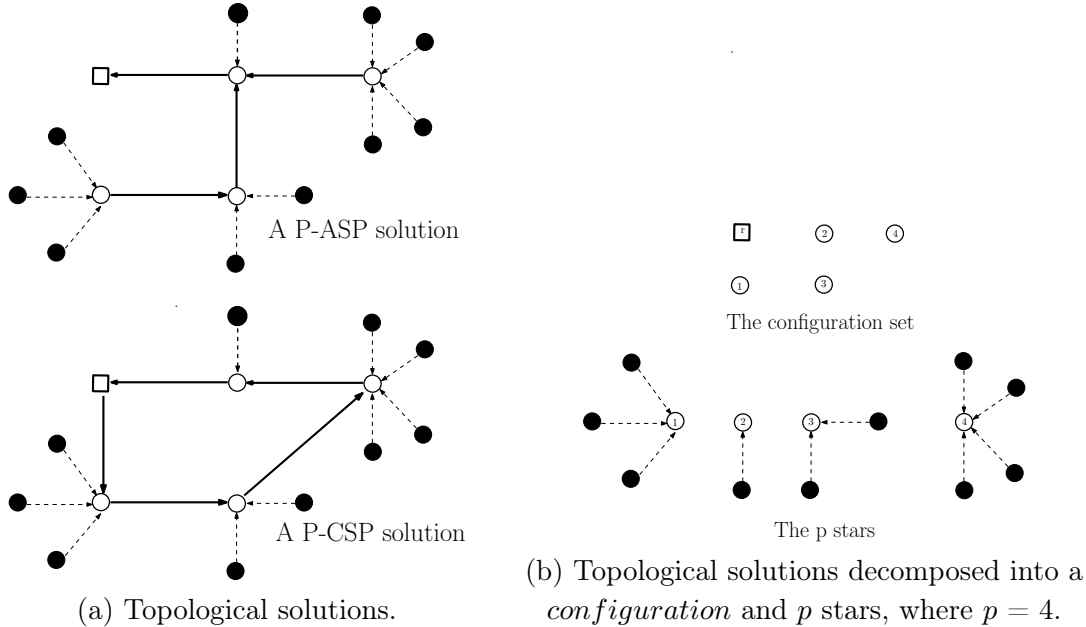


Figure 4.1: Illustration of decomposition examples for p-ASP and p-CSP solutions in a configuration with the sink and $p = 4$ cluster-heads, together with a set of stars incident to that configuration.

Let Λ be the set of feasible configurations, i.e., the set of vertex subsets in D , satisfying cardinality constraint on p . Denote by $W(\lambda)$ the vertex subset defining a configuration $\lambda \in \Lambda$, such that $|W(\lambda)| = p+1$, and $N = V \setminus \{r\}$. For each configuration $\lambda \in \Lambda$, define a constant $a_{j\lambda} = 1$ to indicate whether a vertex $j \in N$ is a cluster-head in λ , $j \in W(\lambda)$, and $a_{j\lambda} = 0$ otherwise. To keep the consistency, since the *sink* belongs to the configuration, we let $a_{r\lambda} = 1$. Every other vertex $i \in N \setminus W(\lambda)$ is assigned to a cluster-head $j \in W(\lambda) \setminus \{r\}$, defining a constant $b_{ij\lambda} = 1$ if $(i, j) \in A$ is an intra-cluster arc incident to configuration λ and $b_{ij\lambda} = 0$ otherwise. Finally, to each configuration $\lambda \in \Lambda$ we could assign a weight $d_\lambda = \sum_{(i,j) \in A} b_{ij\lambda} d_{ij}$, i.e., the total cost of the stars incident to cluster-heads of λ . Given any linear integer programming formulation \mathcal{U} for p-ASP or p-CSP, denote by $v(\mathcal{U})$ its linear programming (LP) relaxation lower bound.

Let $x_\lambda \in \mathbb{B} = \{0, 1\}$, for each λ belonging to the configuration set, and $\{z_{ij} \in \mathbb{B} : (i, j) \in A\}$ denote respectively the decision variables used to indicate the configuration being used and the arcs defining a backbone. If λ is used $x_\lambda = 1$, otherwise $x_\lambda = 0$. Likewise, $z_{ij} = 1$ if (i, j) belongs to the backbone, in this case i and j are cluster-heads in a configuration, otherwise $z_{ij} = 0$. Denote by \mathcal{C} the polytope given by the intersection of:

$$\sum_{\lambda \in \Lambda} x_\lambda \geq 1, \quad (4.1)$$

$$\sum_{\lambda \in \Lambda} a_{i\lambda} x_\lambda - \sum_{(i,j) \in \delta_i^+} z_{ij} = 0, \quad i \in N, \quad (4.2)$$

$$\sum_{\lambda \in \Lambda} (b_{ij\lambda} - a_{j\lambda}) x_\lambda + z_{ij} + z_{ji} \leq 0, \quad (i,j) \in A, \quad (4.3)$$

$$\sum_{\lambda \in \Lambda} (a_{v\lambda} + \sum_{j \in S \cap \delta^+ v} b_{vj\lambda}) x_\lambda - \sum_{(i,j) \in \delta^+(S)} z_{ij} \leq 0, \quad S \subseteq N, S \neq \emptyset, v \in S \quad (4.4)$$

$$z_{ij} \geq 0, \quad (i,j) \in A, \quad (4.5)$$

$$x_\lambda \geq 0, \quad \lambda \in \Lambda, \quad (4.6)$$

where $\delta^+(S) = \{(i,j) \in A : i \in S, j \notin S\}$ is the set of arcs pointing out from the vertex subset $S \subset V$ and by $\delta^-(S) = \{(i,j) \in A : i \notin S, j \in S\}$ the set of arcs pointing into the vertex subset S . For simplicity, we use δ_i^+ and δ_i^- instead of $\delta^+(\{i\})$ and $\delta^-(\{i\})$ when referring to the vertex subset $S = \{i\}$.

A constraint (4.1) states that at least one configuration must be selected. In order to prevent slowing down the convergence when computing the LP relaxation of our formulations, since the dual multiplier assigned to (4.1) would lead to a free variable, this constraint is stated in inequality form. Indeed, that can be done since our models lead to minimization problems. Thus, this would be equivalent as to keep a set partitioning formulation [Feillet, 2010]. Constraints (4.2) force each cluster-head to have an outgoing backbone arc. Constraints (4.3) play the role of *linking* constraints among variables z and x . If an arc $(i,j) \in A$ is selected to be in the backbone, the same arc cannot be in an intra-cluster star. Consequently, in the former case, both of its endpoints must be cluster-heads. Inequalities (4.4), named here *extended directed cutset* (EDCUTs) constraints, play the role of sub-cycle breaking inequalities. EDCUTs enforce that for any vertex subset $S \subseteq N$, if a vertex $v \in S$ is a cluster-head or is connected to a cluster-head in S , there must exist backbone arcs pointing out from S . Finally, (4.5) and (4.6) impose the non-negativity on the variables. Note that \mathcal{C} is represented in terms of exponentially many variables (columns) and constraints (the EDCUTs).

A feasible p-ASP solution is defined by a pair $(W(\lambda), A^\lambda)$, $W(\lambda)$, as before, being a vertex subset defining a connected configuration λ and $A^\lambda = \{(i,j) \in A : b_{ij\lambda} = 1\}$ the set of arcs defining the p -stars pointing into $W(\lambda) \setminus \{r\}$. Under the spanning arborescence representation, no backbone arc pointing out of the sink exists. Likewise, at least one backbone arc must reach the sink. These are imposed respectively by

constraints (4.7) and (4.8).

$$\sum_{(r,i) \in \delta_r^+} z_{ri} = 0, \quad (4.7)$$

$$\sum_{(i,r) \in \delta_r^-} z_{ir} \geq 1. \quad (4.8)$$

The p-ASP can be described with the following configuration-based formulation:

$$\min \left\{ \sum_{\lambda \in \Lambda} d_\lambda x_\lambda + \sum_{(i,j) \in A} c_{ij} z_{ij} : (x, z) \in \mathcal{P}_a \cap (\mathbb{B}^{|\Lambda|} \times \mathbb{B}^{|A|}) \right\}, \quad (4.9)$$

where \mathcal{P}_a is the intersection of \mathcal{C} , (4.7) and (4.8).

In order to extend the configuration decomposition idea to p-CSP, one may impose the equality between indegree and outdegree to every cluster-head. Accordingly, exactly one backbone arc of the *simple directed cycle* must be pointing out from a selected cluster-head (or sink) and exactly one backbone arc must be pointing into that cluster-head. Out-degree and in-degree of a configuration vertex is enforced respectively by:

$$\sum_{\lambda \in \Lambda} a_{i\lambda} x_\lambda - \sum_{(i,j) \in \delta_i^+} z_{ij} = 0, \quad i \in V, \quad (4.10)$$

$$\sum_{\lambda \in \Lambda} a_{i\lambda} x_\lambda - \sum_{(j,i) \in \delta_i^-} z_{ji} = 0, \quad i \in V. \quad (4.11)$$

Together, (4.10) and (4.11) are referred as the *degree conservation* constraints. Thus, a configuration-based formulation for p-CSP is given by:

$$\min \left\{ \sum_{\lambda \in \Lambda} d_\lambda x_\lambda + \sum_{(i,j) \in A} c_{ij} z_{ij} : (x, z) \in \mathcal{P}_c \cap (\mathbb{B}^{|\Lambda|} \times \mathbb{B}^{|A|}) \right\}, \quad (4.12)$$

where \mathcal{P}_c is the intersection of \mathcal{C} , (4.10) and (4.11).

4.2 Column-and-row generation algorithms

As previously described, formulations (4.9) and (4.12) involve exponentially many EDCUTs and columns, one column for each pair $(W(\lambda), A^\lambda)$ associated to a configuration $\lambda \in \Lambda$. Therefore, in order to compute the LP relaxation lower bound $v(\mathcal{P}_a)$ and $v(\mathcal{P}_c)$ we devise a *column-and-row* generation (CRG) algorithm, where columns and EDCUTs are separated on-the-fly by dynamic column generation and cutting planes routines. Our procedure, first proceeds with the column generation routine and then, when no column with attractive reduced cost is found, it separates EDCUTs. Let us assume that $\alpha \leq 0$, $\{\pi_i : i \in V\}$, $\{\beta_{ij} \geq 0 : (i, j) \in A\}$, $\{\mu_S^v \geq 0 : v \in S, S \subset N, S \neq \emptyset\}$ and $\{\theta_i : i \in V\}$ are the dual variables respectively assigned to (4.1), (4.2) (or 4.10 when solving p-CSP), (4.3), (4.4) and (4.11). Consider that a subset of columns $\bar{\Lambda} \subset \Lambda$ and EDCUTs is given, then when relaxing the integrability constraints of a configuration-based formulation we define a restricted linear programming master program (RLMP). Denote by $RLMP_a$ and $RLMP_c$ the corresponding master programs associated respectively to formulations (4.9) and (4.12).

Assuming that an optimal basic feasible solution to a given RLMP does exist, the *pricing subproblem* associated to that master problem consists of finding a configuration $\lambda \in \Lambda$ that meets the following expression:

$$\alpha > \sum_{(i,j) \in A} \xi_{ij} b_{ij\lambda} + \sum_{i \in N} \nu_i a_{i\lambda}, \quad (4.13)$$

where $\xi_{ij} = d_{ij} - \beta_{ij} - \sum_{S \subset V: i, j \in S} \mu_S^i$, $(i, j) \in A$. Whenever solving $RLMP_a$, $\nu_i = \sum_{j \in \delta_i^+} \beta_{ji} - \pi_i - \sum_{S \subset V: i \in S} \mu_S^i$, but when solving $RLMP_c$ $\nu_i = \sum_{j \in \delta_i^+} \beta_{ji} - \pi_i - \theta_i - \sum_{S \subset V: i \in S} \mu_S^i$, for all $i \in N$. Configurations that meet that expression are named: *configurations with negative reduced cost*.

4.2.1 Pricing subproblem

The pricing subproblem behind expression (4.13) is the p -median problem with weights on arcs and vertices. The goal is to find a vertex subset which plays the role of medians on D with cardinality equals to p , such that those p median vertices plus the *sink* constitute a configuration. From now on, we will also refer to a cluster-head as a median vertex. Then, the *pricing subproblem* can be formulated as the following linear integer program (IP):

$$\min \left\{ \sum_{(ij) \in A} \xi_{ij} y_{ij} + \sum_{i \in N} \nu_i h_i : (h, y) \in \mathcal{P}_s \cap (\mathbb{B}^{n+1} \times \mathbb{B}^{|A|}) \right\} \quad (4.14)$$

where $\{h_i \in \mathbb{B} = \{0, 1\} : i \in V\}$ and $\{y_{ij} \in \mathbb{B} : (i, j) \in A\}$ are respectively the binary variables used to identify the median vertices and the arcs connecting non-median vertices to the median ones - defining the p stars - and \mathcal{P}_s is the intersection of the well-know p -median constraint set:

$$\sum_{j \in \delta_i^+} y_{ij} = 1 - h_i, \quad i \in N \quad (4.15)$$

$$\sum_{i \in N} h_i = p, \quad (4.16)$$

$$y_{ji} \leq h_i, \quad (j, i) \in A \quad (4.17)$$

and

$$h_r = 1, \quad (4.18)$$

$$\sum_{i \in V} (y_{ir} + y_{ri}) = 0. \quad (4.19)$$

Where (4.18) and (4.19) are only used to keep the consistency with *sink*.

There is a rich literature concerning this well-known problem. Advanced solution techniques based on Lagrangian heuristic [Cornuejols and Nemhauser, 1977; F.Senne and Lorena, 2000], metaheuristics [Resende and Werneck, 2004] and exact algorithm [Avella et al., 2006] have been proposed in the last decades to solve this location-allocation problem. Different variants of Lagrangian relaxation based on \mathcal{P}_s are proposed in the literature. Recently, the best results have been obtained by dualizing constraints (4.15). By doing so, the *Lagrangian dual problem* is written as:

$$\max_{\gamma} \min_{y, h} \sum_{(i, j) \in A} q_{ij} y_{ij} + \sum_{j \in N} \nu_j h_j + \sum_{i \in N} \gamma_i, \quad (4.20)$$

subject to (4.16)-(4.19), $\{h_j \in \mathbb{B} = \{0, 1\} : j \in V\}$ and $\{y_{ij} \in \mathbb{B} : (i, j) \in A\}$, where lagrangian multipliers $\gamma \in \mathbb{R}^{|N|}$ are assigned to (4.15) defining the *lagrangian costs* $\{q_{ij} = \xi_{ij} - \gamma_i : i, j \in N\}$. The cost of selecting a given $j \in N$ as a median vertex may

be defined as $\tau_j = \gamma_j + \nu_j + \sum_{i \in N} \min\{0, q_{ij}\}$, named here as *auxiliary cost*. Thus, the *Lagrangian subproblem* can be defined as:

$$L(\gamma) = \min \left\{ \sum_{j \in N} \tau_j h_j + \sum_{i \in N} \gamma_i : (h, \gamma) \in (4.16) \cap (4.18) \cap (\mathbb{B}^{n+1} \times \mathbb{R}^{|N|}) \right\} \quad (4.21)$$

For a given fixed $\gamma \in \mathbb{R}^{|N|}$, $L(\gamma)$ is computed by the simple enumeration of the p vertices with the smallest value of τ_j plus $\sum_{i \in N} \gamma_i$. In order to provide the tightest lower bound on (4.14), lagrangian multipliers γ are typically computed with the subgradient method.

4.2.1.1 Algorithms for solving the pricing subproblem

In order to price out non-basic configurations to enter the RLMP, we use heuristic and exact methods adapted from p -median's solution methods found in the literature. The main idea is to feed the CRG routine with a diverse set of configurations generated by heuristics and invoke the exact procedure in just a few iterations to ensure that the optimality requirement is met. Based on that, we first resort to a Lagrangian heuristic based on (4.21). Whenever that method fails we call a branch-and-cut algorithm, implemented on MIP solver (CPLEX) to solve (4.14). Independently of the iteration on our pricing scheme, whenever a configuration with negative reduced cost is found it is appended to the corresponding RLMP and a new iteration takes place. The algorithms for solving the pricing subproblem are:

- **Lagrangian heuristic:** The Lagrangian heuristic for solving the pricing subproblem is based on the Lagrangian relaxation (4.20). In this sense, the Lagrangian subproblem (4.21) corresponds to a simple enumeration of p vertices with smallest auxiliary cost. This relaxation was proposed by Narula et al. [1977] and has been successfully used in the literature providing good quality lower bounds. To evaluate the lower bound of (4.14) the subgradient method [Held and Karp, 1970] is used to compute the lagrangian multipliers γ .

Figure 4.2 shows the pseudo-code of the Lagrangian heuristic implemented in this work. At each subgradient iteration k , multipliers γ^k are found and the auxiliary cost τ_j^k computed in Lines 2-3. The lower bound $LB^k = L(\gamma^k)$ is evaluated with the following steps: In Line 4, sort $\{\tau_j^k : j \in N\}$ in non-decreasing order; In Line 5, select p vertices with the smallest value of τ_j^k as medians and set $h_j = 1$, every other non median vertex $i \in N$ ($h_i = 0$) is connected (setted out $y_{ij} = 1$) to a

median j whenever $q_{ij}^k < 0$ and $h_j = 1$ (those arcs added to the solution have associated negative *lagrangian costs* already computed on τ_j^k); In Line 6, sum up $\sum_{j \in N} \tau_j h_j + \sum_{i \in N} \gamma_i$ to compute the bound. This scheme also gives rise to a feasible configuration λ in Lines 7-8. With the *sink* and p cluster-heads $j \in N$ with the smallest cost τ_j^k , select in Line 7, connect all the remaining $|N| - p$ vertices to the closest cluster-heads defining the value of $b_{ij\lambda}$ for $(i, j) \in A$ in Line 8. The upper bound UB^k assigned To λ is also computed at each subgradient iteration k . This is a straightforward way to find primal solutions based on the Lagrangian relaxation (4.21).

```

begin LagrangianHeuristic()
1. for each subgradient iteration  $k$  do
2.   Get multipliers  $\gamma^k$ ;
3.   Compute auxiliary cost  $\tau_j^k$ ;
4.   Sort  $\{\tau_j^k : j \in N\}$  in non-decreasing order;
5.   Select  $p$  vertices with the smallest value of  $\tau_j^k$  as cluster-heads (set  $h_j = 1$  and  $a_{j\lambda} = 1$ );
6.   Compute  $LB^k$  by summing up  $\sum_{j \in N} \tau_j h_j + \sum_{i \in N} \gamma_i$ ;
7.   Define a backbone with the sink and  $p$  cluster-heads in  $N$  (those with  $a_{j\lambda} = 1$ );
8.   Connect vertices  $\{i \in N : h_i = 0\}$  to the closest cluster-heads  $\{j \in N : a_{j\lambda} = 1\}$  to define  $b_{ij\lambda}$  for  $(i, j) \in A$ ;
9. end-for
end
    
```

Figure 4.2: Pseudo-code of Lagrangian heuristic to solve the pricing subproblem implemented in the CRG procedure.

- **Branch-and-cut algorithm:** BCS algorithm based on (4.14) relies on a branch-and-cut method to solve the pricing subproblem to optimality. BCS is implemented with calls to CPLEX concert library (CPLEX, version 12.6). In order to avoid spending too much time at solving the pricing subproblems by the exact procedure, BCS is always aborted right after finding the first configuration with negative reduced cost, no matter which node, the root or its descendants, is being investigated. The BCS algorithm is embedded with a pre-processing phase where we resort to reduction tests derived from the p -median literature to fix in advance a subset of variables and thus reduce the size of the problem to be solved. The reduction tests used here are adapted from [Avella et al., 2006].

We refer to [Barbaros C. Tansel, 1983; Tansel et al., 1983; Mladenović et al., 2007] for surveys on methods to solve p -median problem and its variants that may also be incorporated to the framework described in this section.

4.2.2 EDCUTs separation routine

To separate EDCUTs (4.4) we adapt the separation routine for cutset inequality (DCUT), described in Chapter 2. Given an LP solution (\bar{x}, \bar{z}) , the separation can be efficiently carried out by means of solving a minimum cut problem defined over the support digraph $\bar{D} = (\bar{V}, \bar{A})$, for $\bar{V} = \{i \in V : \sum_{\lambda \in \bar{\Lambda}} a_{i\lambda} \bar{x}_\lambda > 0\}$, $\bar{A} = \{(i, j) \in A : \bar{z}_{ij} > 0\}$ and arc capacities $\{\bar{z}_{ij} : (i, j) \in \bar{A}\}$. Based on the *maximum flow* and *minimum cut* theorem, Dinic's algorithm is used for defining the corresponding mincut set. Thus, let $S \subset N$ be the vertex subset defining the minimum cut $(S, V \setminus S)$ in \bar{D} . An EDCUT inequality is violated whenever the capacity of its corresponding cut, i.e., $\sum_{(ij) \in \delta^+(S)} \bar{z}_{ij}$, is less than $\sum_{\lambda \in \Lambda} (a_{v\lambda} + \sum_{j \in V_v \cap S} b_{vj\lambda}) \bar{x}_\lambda$, for $S \subset N, S \neq \emptyset, v \in S$.

4.2.3 CRG miscellanea

The main drawback of our CRG approach comes from the degeneracy in the associated RLMPs. Take for instance the p-ASP, when solving $RLMP_a$ to compute $v(\mathcal{P}_a)$. Figure 4.3 exemplifies an anomaly that has damaged considerably the algorithm convergence. Observe in (a) that for a given connected feasible configuration, weighted with the dual information, different p-median solutions can be defined as shown in (b).

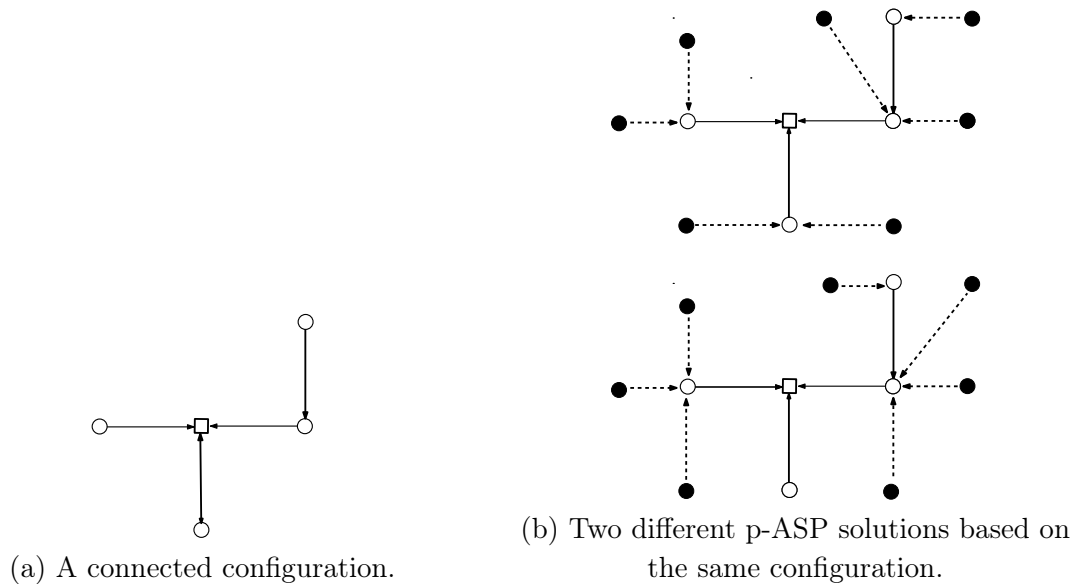


Figure 4.3: Illustration of two different solutions based on the same configuration.

A possible remedy to overcome such a problem is to use a rule to avoid considering columns with different assignments associated to the same configuration. By doing so, consider two different assignments $\{b_{ij\lambda}^1 : (i, j) \in A\}$ and $\{b_{ij\lambda}^2 : (i, j) \in A\}$ with the

respectively weights d_λ^1 and d_λ^2 based on the configuration λ . The rule remarks that the second *priced out* column is only appended to the corresponding RLMP if $d_\lambda^2 < d_\lambda^1$.

In order to reduce the total time spent to evaluate the bounds with CRG algorithm, due to degeneracy and convergence difficulties, we resort to a stabilization method. Among many stabilization methods found in the literature, e.g. [du Merle et al., 1999; Desrosiers and Lübbecke, 2005; Rousseau et al., 2007; Amor et al., 2009], we use the dual stabilization procedure described in Pessoa et al. [2010]. The procedure interacts over a model penalized with dual variables, where artificial variables bounded by a single parameter ε are appended to RLMP introducing positive and negative terms. The penalties change in the course of the algorithm until the dual multiplier converge to an optimal dual solution. In practice, to avoid the need of estimating a proper value of ε , the parameter is decreased at every round of the column-and-row generation algorithm. The complete algorithm runs four rounds with $\varepsilon = (0.1, 0.01, 0.001, 0.0)$. When $\varepsilon = 0.0$, the stabilized model is equivalent to desired restricted master problem. The columns generated on each round of the algorithm must be kept in the model.

4.2.3.1 CRG algorithms and the primal bounds

Figure 4.4 shows the pseudo-code of the overall CRG algorithm described in this work. It starts from the initial configuration λ , generated with the constructive heuristics described at the end of this subsection, and an empty EDCUT set. The loop in Lines 4-18 is performed until no configurations with negative reduced cost nor EDCUTs are found. The column generation steps are implemented in Lines 5-11. Based on the current $\bar{\Lambda}$ and EDCUT sets, the RLMP is defined in Line 5. After solving the corresponding RLMP (we used CPLEX, version 12.6 for that) in Line 6, the dual and primal LP solutions are computed. The best lower bound (LB) is updated in Line 7. A new configuration λ' is obtained when performing a pricing procedure in Line 8. Algorithms for solving the pricing subproblems are given in subsection 4.2.1. The feasibility of the primal solution based on λ' is tested to update the best upper bounds (UB) in Line 9. When no column with attractive reduced cost is found, with the primal solution (\bar{x}, \bar{z}) on hand, EDCUTs are separated in Line 13. The current configuration and EDCUT sets are augmented in Lines 11 and 15, respectively. The best known LB and UB are returned in Line 19.

CRG-A and CRG-C are the implementation of the *column-and-row* generation algorithms described above, based on \mathcal{P}_a and \mathcal{P}_c , to evaluate respectively $v(\mathcal{P}_a)$ and $v(\mathcal{P}_c)$. In the course of these algorithms, no matter which iteration is running, when-

```

begin CRG()
1. Generate a configuration  $\lambda$ ;
2.  $\bar{\Lambda} \leftarrow \{\lambda\}$ ;
3. EDCUT  $\leftarrow \emptyset$ ;
4. while configurations with negative reduced cost or EDCUTs are found do
5.   Define RLMP with  $\bar{\Lambda}$  and EDCUT;
6.   Solve the RLMP model and compute  $(\bar{\alpha}, \bar{\xi}, \bar{\nu}, \bar{x}, \bar{z})$ ;
7.   Update the best LB;
8.    $\lambda' \leftarrow \text{PricingSubproblem}(\bar{\alpha}, \bar{\xi}, \bar{\nu})$ ;
9.   Update the best UB;
10.  if  $\lambda'$  is a configuration with negative reduced cost then
11.     $\bar{\Lambda} \leftarrow \bar{\Lambda} \cup \{\lambda'\}$ ;
12.  else
13.    CUTs  $\leftarrow \text{SeparateEDCUTs}(\bar{x}, \bar{z})$ ;
14.    if CUTs is not empty then
15.      EDCUT  $\leftarrow \text{EDCUT} \cup \text{CUTs}$ ;
16.    end-if
17.  end-if
18. end-while;
19. return LB and UB;
end

```

Figure 4.4: Pseudo-code of the overall column-and-row (CRG) generation for p-ASP and p-CSP. In the CRG, the PricingSubproblem procedure (Line 8) uses algorithms for solving the pricing subproblems given in subsection 4.2.1.

ever a configuration with negative reduced cost is priced out, say $\bar{\lambda} \in \Lambda$, we test the connectivity of its corresponding cluster-heads and *sink* components in order to compute a feasible solution to the problem being solved and then update the best primal bound.

In CRG-A, to obtain a p-ASP solution, a *depth-first search* algorithm is used to test the connectivity of a configuration. If $(W(\lambda), A^\lambda)$ is connected we have a valid p-ASP upper bound. Note that, when $\bar{\lambda}$ is a non-connected configuration, instead of achieving a primal solution we may end up with a violated EDCUT inequality. Such a cut, when found, is appended to the model in advance in order to speed up the method.

CRG-C is initialized with a valid upper bound computed as following. Let $C = V \setminus \{r\}$ and $W = \{r\}$ be respectively the initial candidate and cluster-head set. Initially, the core cycle is given by $r \longleftrightarrow r$, then, at each iteration, a candidate vertex $k \in C$ with smallest score l_k is removed from C and added in W until a desire cardinality $|W| = p + 1$ is reached, increasing the cycle by one arc. With a feasible cluster-head set on hand, we assign every other vertex in $V \setminus W$ to the cheapest cluster-head in $W \setminus \{r\}$. The score function l_k is: $l_k = (c_{ik} + c_{kj} - c_{ij}) + \sum_{j \in V_k \cap C} d_{kj}$, for every $k \in C$. The first term of that equation computes the insertion cost of a vertex $k \in C$ between every pair of adjacent vertices $\{i, j\}$ in the current cycle, while the second term corresponds to an estimation on the intra-cluster cost if k is chosen as a cluster-head. In order to update

the best upper bound, whenever a new configuration is defined, a p-CSP solution is iteratively constructed by simply testing the connectivity of the core cycle using the same steps as before.

In the next section, we will present the results and discussion regarding our configuration-based formulations and the procedure to evaluate their corresponding bounds.

4.3 Computational results

In this section, we detail the computational experiments with the column-and-row generation algorithms implemented for p-ASP and p-CSP. We also present the LP relaxations provided by our configuration-based formulations and compare against formulations based on directed multicommodity flow models and on directed cutset (DCUT) inequalities. The experiments were performed on a subset of instances with LP relaxations available, given in Chapters 2 and 3. For p-ASP, we consider instances with $p \in \{4, 5, 8, 10\}$, $n + 1 \in \{30, 51, 76, 99, 101\}$ and graph density $\in \{30\%, 40\%, 50\%, 60\%, 70\%, 100\%\}$. For p-CSP, we consider instances with $p \in \{5, 10, 15, 20\}$, $n + 1 \in \{51, 70, 76, 99, 100\}$. All instances considered here are available at <http://www.dcc.ufmg.br/~vwcmorais/instances/>. Details of how the instances were generated can be obtained on the cited chapters.

For the results that follow, CRG-A and CRG-C are coded in C++ and run under the Linux operational system. The compiler used is GNU g++ with optimization flag -O3 turned on. An Intel® XEON E5645 Core™ i7-980 hexa-core machine, running at 2.4GHz, with 24 GB of shared RAM is used in the experiments. No multi-threading is allowed and the algorithms relied on the LP and MIP solvers of IBM ILOG CPLEX concert library (version Optimization Studio 12.6) with all of their preprocessing, cut generation and primal heuristic modules being turned off when performing with the BCS and solving the RLMPs.

In the first experiment, the quality of the bounds provided by the p-ASP configuration-based formulation \mathcal{P}_a is compared with formulations \mathcal{P}_F and \mathcal{P}_D in Chapter 2. \mathcal{P}_F is a compact multicommodity flow formulation involving continuous variables, flow conservation and forcing constraints to ensure connectivity among cluster-heads and sink. \mathcal{P}_D is a directed cutset (DCUT) formulation that enforces the connectivity by means of an exponential number of breaking sub-cycles constraints. The stopping criterion of CRG-A was set to 5 hours of CPU time, following the experiments settings in Chapter 2. The results obtained by evaluating the bound $v(\mathcal{P}_a)$ with CRG-A are

detailed in Table 4.1. The first column in that table identifies the test instances. The lower bounds provided by formulations \mathcal{P}_F and \mathcal{P}_D and CPU time (quoted in seconds) to compute their bounds are presented in columns 2 to 5, respectively. The next five columns display results obtained by CRG-A, we report: the number of cuts (EDCUT), the number of columns, the value of $v(\mathcal{P}_a)$, the best rounded up upper bounds (UB) achieved over the course of the algorithm and the time taken to compute the bounds. Finally, the last column provides the corresponding known optimal objective function value (OPT) provided in the reference cited above. It can be observed that \mathcal{P}_a strictly dominates the other two formulations, since the lower bound $v(\mathcal{P}_a)$ is stronger than those provided by \mathcal{P}_D and \mathcal{P}_F . \mathcal{P}_D is slightly better than \mathcal{P}_F . The results confirm that $\mathcal{P}_a \subset \mathcal{P}_D \subset \mathcal{P}_F$. However, computing $v(\mathcal{P}_a)$ is more expensive than $v(\mathcal{P}_D)$. The evaluation of $v(\mathcal{P}_F)$ is more time consuming than both of its counterpart formulations. One can also observe that CRG-A provides upper bounds that are very close to the optimal solution, that shows the effectiveness of our heuristic to generate primal solutions. This was observed in all the instances tested. CRG-A manages to provide the optimality certificate on 12 out of 20 instances.

Instance					CRG-A algorithm					OPT	
	$v(\mathcal{P}_F)$	$t(s)$	$v(\mathcal{P}_D)$	$t(s)$	$\#cuts$	$\#cols$	$v(\mathcal{P}_a)$	UB	$t(s)$		
n30p4d60	540.33	5.94	551.17	0.03	72	119	555.00	555	3.31	555	(*)
n30p5d100	379.33	12.86	387.00	0.05	176	298	387.00	387	6.27	387	(*)
n30p8d50	463.50	5.07	472.00	0.04	213	504	472.00	472	10.72	472	(*)
n30p8d60	395.00	4.87	400.75	0.05	236	252	404.00	404	6.98	404	(*)
Avg.	444.54	7.19	452.73	0.04	174.25	293.25	454.50	454.50	6.82	454.50	
n51p5d100	629.37	438.12	633.59	1.16	374	1043	637.37	640	131.69	640	(+)
n51p8d30	831.34	43.79	836.01	0.07	308	783	851.00	854	152.19	854	(+)
n51p8d40	750.32	68.93	753.96	0.19	319	880	762.14	763	55.87	763	(+)
n51p10d100	477.91	394.04	486.66	1.43	415	952	490.99	504	142.03	499	(+)
Avg.	672.24	236.22	677.56	0.71	354.00	914.5	685.45	690.25	120.44	689.00	
n76p5d100	915.39	9111.92	923.69	10.59	1046	4547	928.86	935	3357.42	934	
n76p10d50	877.26	1875.80	883.35	5.24	1024	2551	895.00	895	1593.03	895	(+)
n76p10d60	815.31	6457.66	824.30	5.46	1381	3163	833.00	833	1612.26	833	(+)
n76p10d70	770.57	3563.08	776.19	2.02	696	2377	786.65	787	1065.15	787	(+)
Avg.	844.63	5252.12	851.88	5.83	1036.75	3159.50	860.88	862.50	1906.96	862.25	
n99p5d50	3432.84	1764.11	3449.33	4.98	476	1707	3661.01	3719	517.92	3719	(+)
n99p5d60	3284.39	1086.90	3297.79	3.88	590	2058	3582.00	3582	746.19	3582	(+)
n99p5d100	2637.99	17955.20	2665.56	93.19	903	4446	2670.37	2685	1874.65	2678	
n99p10d50	2475.47	13354.00	2512.30	15.36	2754	4220	2586.28	2623	5428.33	2617	
Avg.	2957.67	8540.05	2981.25	29.35	1180.75	3107.75	3124.92	3152.25	2141.77	3149.00	
n101p5d50	1652.74	15908.70	1658.21	1.78	227	5110	1799.96	1817	967.60	1817	(+)
n101p5d60	1431.54	729.85	1439.15	6.75	218	1214	1511.00	1511	527.42	1511	(+)
n101p10d40	1244.88	7167.84	1256.47	20.38	1965	2097	1291.04	1347	584.41	1336	
n101p10d50	1112.93	11401.90	1122.38	3.91	1197	2153	1169.27	1183	424.18	1176	
Avg.	1360.52	8802.07	1369.05	8.21	901.75	2643.50	1442.82	1464.50	639.73	1460.00	

Table 4.1: Comparison of formulations \mathcal{P}_F , \mathcal{P}_D and \mathcal{P}_a for p-ASP. Instances on which the procedure finds certified optimal values are indicated with (*) and instances on which the procedure finds an optimal solution but fails in providing an optimality certificate are indicated with (+).

In the second experiment, we assess the quality of the bounds provided by the p-CSP configuration based formulation \mathcal{P}_c and compare with formulations \mathcal{F}_u , \mathcal{D}_d and \mathcal{D}_d^+ in Chapter 3. \mathcal{F}_u is a compact multicommodity flow formulation strengthened with additional capacity and linking inequalities. \mathcal{D}_d is a DCUT formulation strengthened with capacity constraints; On the other hand, \mathcal{D}_d^+ is a formulation that comes from the intersection of \mathcal{D}_d with a lifted version of DCUT inequalities. The stopping criterion of the algorithm CRG-C was set to 3 hours of CPU time, following the experiments settings in Chapter 3.

Instance							CRG-C algorithm					OPT
	$v(\mathcal{F}_u)$	$t(s)$	$v(\mathcal{D}_d)$	$t(s)$	$v(\mathcal{D}_d^+)$	$t(s)$	#cuts	#cols	$v(\mathcal{P}_c)$	UB	$t(s)$	
eil51_3_5	4215.90	3451.60	4214.00	0.70	4225.40	0.70	159	108	4248.98	6202.00	8.24	4253
eil51_5_5	3320.70	3091.20	3310.50	0.60	3338.70	0.70	114	857	3378.41	4604.00	165.15	3421
eil51_8_5	1737.30	3591.60	1731.50	0.60	1916.10	2.10	732	5322	1917.07	2182.00	220.78	1990
eil51_3_10	2983.80	3538.80	2982.80	0.50	3013.80	0.70	450	147	3033.86	4945.00	18.39	3060
eil51_5_10	2576.20	5320.80	2575.00	0.50	2606.60	0.60	408	1240	2653.68	3782.00	430.30	2719
eil51_8_10	1672.30	5752.00	1665.00	0.60	1852.20	1.60	89	1360	1858.01	2026.00	209.92	1872
eil51_3_15	2525.30	7654.50	2524.30	0.50	2567.40	0.70	1497	280	2586.98	3887.00	118.22	2608
eil51_5_15	2305.70	6445.90	2304.60	0.50	2387.30	0.80	1059	1519	2423.06	3158.00	392.42	2461
eil51_8_15	1749.40	3591.70	1748.80	0.50	1889.10	1.00	148	1493	1898.00	2061.00	166.94	1898 (+)
eil51_3_20	2223.30	2325.10	2223.10	0.50	2264.00	1.00	978	256	2269.45	3263.00	100.99	2287
eil51_5_20	2183.10	1460.10	2183.10	0.50	2271.20	1.10	1784	1426	2273.73	2843.00	449.92	2292
eil51_8_20	1894.50	1466.70	1893.70	0.40	2001.70	0.70	1446	4768	2011.00	2223.00	889.88	2011 (+)
Avg.	2448.96	3974.17	2446.37	0.53	2527.79	0.98	738.67	1564.67	2546.02	3431.33	264.26	2572.67
st70_3_5	8268.50	3560.40	8258.30	1.20	8278.40	1.30	96	125	8305.00	13702.00	23.32	8306
st70_5_5	6484.10	5096.30	6453.80	2.50	6582.60	3.00	603	2709	6586.11	10183.00	1021.49	6612
st70_8_5	3410.60	6930.30	3338.20	4.90	3709.30	7.20	2118	5036	3726.40	4893.00	1225.92	3902
st70_3_10	5637.40	4053.50	5634.00	1.90	5655.00	2.30	1232	253	5655.00	11456.00	63.83	5655 (+)
st70_3_15	4506.40	5036.80	4500.10	2.10	4547.00	2.20	4254	401	4547.00	4556.00	108.88	4547 (+)
st70_3_20	-	-	3834.30	2.30	3905.20	2.90	6453	358	3908.80	7821.00	370.20	3912
st70_5_20	-	-	3607.00	3.20	3816.90	8.00	4795	2464	3818.40	4031.00	1223.50	3820
st70_8_20	-	-	2724.40	2.20	3287.00	12.10	6808	3152	3287.00	4031.00	2145.90	3287 (+)
Avg.	-	-	4793.76	2.54	4972.68	4.88	3294.88	1812.25	4979.21	7584.13	772.88	5005.13
eil76_3_5	6224.70	3590.90	6224.70	1.40	6232.90	1.40	0	11	6259.00	10579.00	4.84	6259 (+)
eil76_5_5	4816.40	5959.10	4807.20	2.00	4827.60	2.10	7688	1028	4872.81	7688.00	598.73	4891
eil76_8_5	2412.70	8121.50	2374.60	3.80	2637.40	11.70	756	5504	2641.38	3475.00	1349.19	2722
eil76_3_10	4503.90	8288.40	4498.30	3.30	4557.00	5.70	4053	385	4557.00	8417.00	671.32	4557 (+)
eil76_8_10	-	-	2006.80	4.30	2370.50	19.50	4116	3423	2387.35	3122.00	1420.64	2468
eil76_8_15	-	-	1991.10	4.40	2339.20	28.50	4854	3245	2339.89	2905.00	2265.70	2384
eil76_8_20	-	-	2048.30	4.40	2339.10	30.00	28	4181	2339.18	2837.00	1987.24	2345
Avg.	-	-	3421.57	3.37	3614.81	14.13	3070.71	2539.57	3628.09	5574.71	1185.38	3660.86
rat99_3_5	-	-	18127.90	6.40	18314.00	10.20	969	578	18314.00	35012.00	581.75	18314 (+)
rat99_3_10	-	-	12548.70	4.90	12649.00	10.40	2341	474	12649.00	28731.00	301.57	12649 (+)
Avg.	-	-	15338.30	5.65	15481.50	10.30	1655.00	526.00	15481.50	31871.50	441.66	15481.50
kroD100_3_5	-	-	349514.50	3.60	349559.00	3.60	12480	2	349559.00	349559.00	14.84	349559 (*)
kroD100_3_10	-	-	215398.00	3.50	215398.00	3.50	15853	2	215398.00	593700.00	83.48	215398 (+)
Avg.	-	-	282456.25	3.55	282478.50	3.55	14166.50	2.00	282478.50	471629.50	49.16	282478.50

Table 4.2: Comparison of formulations \mathcal{F}_u , \mathcal{D}_d , \mathcal{D}_d^+ and \mathcal{P}_c for p-CSP. Instances on which the procedure finds certified optimal values are indicated with (*) and instances on which the procedure finds an optimal solution but fails in providing an optimality certificate are indicated with (+).

The results obtained by evaluating the bound $v(\mathcal{P}_c)$ with CRG-C are detailed in Table 4.2. In more detail, the first column in that table identifies the test instances. The lower bounds provided by formulations \mathcal{F}_u , \mathcal{D}_d and \mathcal{D}_d^+ and the CPU times to compute their bounds are presented in columns 2 to 7, respectively. Whenever the bounds are not evaluated within the time limit for a given instance, an indication “-” is given. The next five columns display results attained with CRG-A, we report: the number of cuts, the number of columns, the value of $v(\mathcal{P}_c)$, the best rounded up (UB) achieved over the course of algorithm and the time taken to compute the bounds. Finally, the last column provides the corresponding known optimal objective function value (OPT). Analyzing the bounds provided by the formulations, it can be observed that the bounds provided by \mathcal{P}_c are frequently much closer to the optimal solution than its counterparts. One important observation regarding the lower bounds discussed here is the following: EDCUTs inequalities are the counterpart of the lifted DCUT inequalities of \mathcal{D}_d^+ (see Chapter 3), because of that \mathcal{P}_c and \mathcal{D}_d^+ dominate \mathcal{D}_d . Such constraints significantly improved on bounds. In fact, among all models considered here, \mathcal{D}_d is the weakest. On the other hand, computing $v(\mathcal{P}_c)$ is more expensive than computing the bounds of all DCUT based formulation. As it could be observed, for most of the comparisons carried out, computing $v(\mathcal{P}_c)$ is only faster than evaluating the bound of the compact formulation \mathcal{F}_u .

The results show that the proposed configuration-based formulations provide very sharp lower bounds. Very often, our column-and-row generation algorithms manage to find the optimal solution, but fail in providing an optimality certificate. This suggests that our approach is promising, but there is still room for improvements. For instance, note that the upper bound, and consequently the optimality gaps, found for CRG-C are significantly worse than the ones for CRG-A. The reason for that is how the primal solutions are defined for p-ASP and p-CSP. Whenever a new configuration is found, the connectivity of its corresponding backbone is tested. In p-ASP the connectivity is guaranteed by defining an arborescence while for p-CSP a directed cycle. To define an arborescence efficient algorithms are used and basically, only the connectivity of paths is verified. To build a directed cycle our procedures resort to constructive heuristics. It is important to highlight that the time to compute the lower bound for some instances are still high. The main issues to deal with are the considerable amount of the time spent at solving subproblems and separating the EDCUT inequalities. Convergence problems due to degeneracy is also an issue of our algorithms, several columns are priced out but few are effectively appended to the models due to dominance rules. An additional observation is that, as far as CPU times are concerned and in spite of the difficulties on setting dual stabilization parameters properly, without dual stabilization

the large instances could not be solved in reasonable time. This was observed mainly when solving p-CSP instances.

4.4 Concluding remarks

In this chapter, we dealt with routing problems that integrate coverage and cardinality constraints to optimize the wireless sensor networks lifetime. Problems based on arborescence and cycle topologies are defined in the design of that kind of networks. We also proposed configurations-based formulations and column-and-row generation algorithms to evaluate the corresponding bounds.

Computational experiments showed that configuration-based formulations dominate compact multicommodity flow and directed cutset based formulations in the literature. We also observe that the proposed column-and-row generation algorithms manage to find near-optimal solutions. We remark that the proposed configuration-based formulation could also be used to model related location-allocation problems, such as: tree-star problems, facility location problem, hub location problems, and median cycles problems. The results could be improved by tighter mathematical formulations and also improving heuristics used to speed up the algorithms.

We intend to investigate stabilization methods, other than the one already explored, and improve the algorithms to solve the column generation subproblem. We also plan to investigate whether our algorithms could benefit from using local search procedures.

Chapter 5

Epilogue

The final remarks and future work are presented in this chapter. Section 5.1 concludes the thesis with a summary of the accomplished work and addresses the possible future works.

5.1 Final Remarks

In this Thesis, we studied how to apply optimization techniques to the topological design of wireless sensor network. The optimization problems considered here consist of clustering the sensors and defining a communication topology to gather the sensed information throughout the network. Natural connectivity and coverage requirements are satisfied assuming an imposition on the number of clusters. The first problem investigated, named p -arborescence star problem (p-ASP), organizes the network into a fixed number of p clusters and defines a communication topology as a rooted directed tree, i.e., an *arborescence*, setting directed paths from each cluster-head to the sink, the root of an arborescence. We also investigated solutions to the p -cycle star problem (p-CSP), the cycle analogue to the p-ASP. In p-CSP, instead of a backbone tree, mobile-sink based networks are designed replacing the core arborescence by a directed cycle, representing the route traversed by a mobile sink to visit each of the p predefined cluster-heads. The overall objective of the problems is to maximize the network lifetime by minimizing the clustering and transmission (routing) costs.

Two mathematical formulations were proposed for p-ASP. First, a compact formulation involving additional continuous variables over a multicommodity directed flow model to enforce the connectivity, while in the second approach, connectivity is ensured by means of an exponential number of circuit breaking constraints. In addition to the formulations, we also introduced a Benders-based heuristic and three exact solution

algorithms: BCF-T, BCF-C, and BCD. BCF-T and BCF-C were based on the compact flow-based formulation while BCD was based on circuit breaking constraints. The algorithms differ mainly in the way they handle constraints that ensure the topology connectivity. BCF-C and BCF-T rely on the standard implementation of a cutting-plane algorithm. BCD separates lifted directed cutset inequalities only for fractional solutions at the root node and integer solutions in the remainder of the enumeration tree. Through our computational experiments, it was demonstrated that BCD outperformed BCF-C and BCF-T. Additionally, the results showed the superiority of the directed cutset based formulation over the multicommodity flow formulation. They also validated the idea of simplifying the separation routines in favor of evaluating more BB nodes.

For p-CSP, we first introduced a compact multicommodity directed flow formulation strengthened with additional inequalities. With such formulation on hands, we derived cutset based formulations by projecting out the continuous flow variables. In this scheme, we identified two classes of valid inequalities that were separated in a branch-and-cut algorithm. Computational results obtained with a branch-and-bound algorithm based on the compact flow formulation were compared with branch-and-cut algorithms based on both flow and cutset based formulations. In light of our computational study, we concluded that the algorithm based on cutset models achieve better results than the algorithms based on the multicommodity flow formulations. We also pointed out that the main challenge of p-CSP is to define the optimal location of cluster-heads and enforce connectivity of the core simple cycle.

Under the cardinality constrained topology representation, p-ASP and p-CSP topologies can be decomposed into p -stars and a backbone connecting p cluster-heads and the sink. A common thread of the problems is to select the core cluster-head set and the sink. This structure is named *configuration*. The configurations were explored in such a way to obtain strong mathematical formulations. Thus, we proposed configuration-based linear Integer Programming (IP) formulations and devise column-and-row generation (CRG) algorithms to evaluate the bounds. Through our computational experiments, it was shown that configuration-based formulations dominate any the other p-ASP and p-CSP formulations. The proposed CRG was also indicated as an alternative to solve related location-allocation problems in transportation and telecommunication networks.

In general, the end user is interested in ensuring a fair utilization of the WSN resources such that no cluster-head suffers an early energy depletion. Through this work, we showed that the moment in which a network is considered nonfunctional is specific to the application and heavily depends on the definition of network lifetime

being implemented by a protocol. Based on the connectivity requirement, we placed the problems cited above to design WSN topologies with two network lifetime definitions in mind: time until the first sensor failure and time until the first cluster-head failure or no communication backbone exists. The problems presented were based on protocols that aim to enhance the network lifetime by performing a periodic rotation on a fixed number of preselected cluster-heads. Therefore, exploring the cardinality constraints of this cluster-head set was the central focus of the thesis.

Future works include a Lagrangian relaxation approach based on the cutset based formulation for p-ASP and p-CSP. To compute the bounds one could make use of a relax-and-cut (RC) method. RC would dualize two set of constraints: assignment equalities in a static way and DCUT inequalities in a dynamic fashion. In order to provide a tightest lower bound, the subgradient method would be used to approximately solve the lagrangian dual problem. In this direction, two different BC algorithms may be drawn: delayed relax-and-cut (DRC) and non-delayed relax-and-cut (NDRC). The main difference between DRC and NDRC would be basically the point where the relaxed inequalities are separated and dualized: after computing the LP or at every subgradient iteration. As a further research step, one could also investigate stabilization methods for the CRG procedure, other than the one already explored in this work, and improve the algorithms to solve the pricing subproblem. This would enable the implementation of branch-and-price-and-cut algorithms that take advantage of the limits obtained with the configuration-based formulations. It will also be important to proceed with the characterization valid inequalities, such as capacity constraints that benefit from the cardinality constraint, to separate them into our algorithms.

Appendix A

Supplementary results

The detailed computational results for instances from Bardossy and Raghavan [2010] for BCF-C, BCF-T and BCD, when solving the p-ASP, are given in supplementary Tables A.1, A.2, A.3 and A.4. The first two columns of the tables provide respectively the instance name, the corresponding value of p and the number of arcs on instances of subset being considered. Columns 4 to 10 show respectively the lower bounds (RLB) found at root node, best lower (BLB) and upper (BUB) bounds found during the search, the CPU time (in seconds) at the root node (\mathbf{t}_r) of the enumeration tree, the overall CPU total time (\mathbf{t}_a), the number of branch-and-bound nodes evaluated ($\#nodes$) and the corresponding duality gap ($BUB - BLB / BUB$) provided by BCF-C. Columns 11 to 17 show the same data for BCF-T. Finally, the results obtained by BCD are displayed in columns 18 to 24.

Detailed computational results for algorithms BBF, BCF and BCD, when solving the p-CSP, are given in Tables A.5 and A.6. The first two columns of the tables provide respectively the instance name and the corresponding upper bound provided by heuristic. Columns 3 to 8 show respectively the best lower (BLB) and upper (BUB) bounds found during the search, the corresponding duality gap $(BUB - BLB / BUB)$, the CPU time (in seconds) at the root node (\mathbf{t}_r) of the enumeration tree, the overall CPU total time (\mathbf{t}_a) and the number of branch-and-bound nodes evaluated ($\#nodes$) provided by BBF. Columns 9 to 14 show the same data for BCF. Finally, the results obtained by BCD are displayed in columns 15 to 20. Whenever an instance was not solved to proven optimality within the time limit, an indication “-” is provided.

Bibliography

- Agarwal., P. K. and Procopiuc, C. M. (1998). Exact and approximation algorithms for clustering. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 658--667, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.
- Agarwal, P. K. and Sharir, M. (1998). Efficient algorithms for geometric optimization. *ACM Comput. Surv.*, 30(4):412--458. ISSN 0360-0300.
- Aioffi, W. M., Valle, C. A., Mateus, G. R., and da Cunha, A. S. (2011). Balancing message delivery latency and network lifetime through an integrated model for clustering and routing in wireless sensor networks. *Computer Networks*, 55(13):2803 – 2820. ISSN 1389-1286.
- Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422. ISSN 1389-1286.
- Alemdar, H. and Ersoy, C. (2010). Wireless sensor networks for healthcare: A survey. *Computer Networks*, 54(15):2688--2710. ISSN 1389-1286.
- Amor, H. M. B., Desrosiers, J., and Frangioni, A. (2009). On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6):1167 – 1184. ISSN 0166-218X. Reformulation Techniques and Mathematical Programming.
- Anastasi, G., Marco, Di Francesco Conti, A. M., and Passarella (2009). Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537--568. ISSN 1570-8705.
- Arampatzis, T., Lygeros, J., Member, S., and Manesis, S. (2005). A survey of applications of wireless sensors and wireless sensor networks. In *Proceedings of the 13th Mediterranean Conference on Control and Automation*, pages 719--724, Limassol, Cyprus.

- Arulsevan, A., Bley, A., Gollowitzer, S., Ljubic, I., and Maurer, O. (2011). Mip modeling of incremental connected facility location. In Pahl, J., Reiners, T., and VoB, S., S., editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 490–502. Springer Berlin Heidelberg.
- Avella, P., Sassano, A., and Vasil'ev, I. (2006). Computational study of large-scale p-median problems. *Mathematical Programming*, 109(1):89–114. ISSN 1436-4646.
- Balas, E. and Oosten, M. (2000). On the cycle polytope of a directed graph. *Networks*, 36(1):34–46. ISSN 1097-0037.
- Baldacci, R., Dell'Amico, M., and González, J. S. (2007). The capacitated m-ring-star problem. *Operations Research*, 55(6):1147–1162.
- Barbaros C. Tansel, Richard L. Francis, T. J. L. (1983). Location on networks: A survey. part ii: Exploiting tree network structure. *Management Science*, 29(4):498–511. ISSN 00251909, 15265501.
- Bardossy, M. G. and Raghavan, S. (2010). Dual-based local search for the connected facility location and related problems. *INFORMS Journal on Computing*, 22(4):584–602.
- Basagni, S., Carosi, A., Melachrinoudis, E., Petrioli, C., and Wang, Z. M. (2008). Controlled sink mobility for prolonging wireless sensor networks lifetime. *Wirel. Netw.*, 14(6):831–858. ISSN 1022-0038.
- Calik, H., Leitner, M., and Luipersbeck, M. (2017). A benders decomposition based framework for solving cable trench problems. *Computers & Operations Research*, 81:128–140. ISSN 0305-0548.
- Cardei, M., Thai, M. T., Li, Y., and Wu, W. (2005). Energy-efficient target coverage in wireless sensor networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1976–1984 vol. 3. ISSN 0743-166X.
- Carrabs, F., Cerulli, R., Ambrosio, C. D., and Raiconi, A. (2015). Exact and heuristic approaches for the maximum lifetime problem in sensor networks with coverage and connectivity constraints. Technical report, University of Salerno.
- Castano, F., Rossi, A., Sevaux, M., and Velasco, N. (2014). A column generation approach to extend lifetime in wireless sensor networks with coverage and connec-

- tivity constraints. *Computers & Operations Research*, 52, Part B:220 – 230. ISSN 0305-0548. Recent advances in Variable neighborhood search.
- Chen, Y. and Zhao, Q. (2005). On the lifetime of wireless sensor networks. *IEEE Communications Letters*, 9(11):976–978. ISSN 1089-7798.
- Chouman, M., Crainic, T. G., and Gendron, B. (2016). Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science*, 0(2016).
- Chu, Y. J. and Liu, T. H. (1965). On the shortest arborescence of a directed graph. *Science Sinica*, 4:1396–1400.
- Contreras, I., Fernández, E., and Marín, A. (2010). The tree of hubs location problem. *European Journal of Operational Research*, 202(2):390 – 400. ISSN 0377-2217.
- Cornuejols, G. and Nemhauser, M. L. F. G. L. (1977). Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science*, 23(8):789–810.
- Current, J. R. and Schilling, D. A. (1994). The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research*, 73(1):114 – 126. ISSN 0377-2217.
- de Sá, E. M., Contreras, I., Cordeau, J.-F., de Camargo, R. S., and de Miranda, G. (2015). The hub line location problem. *Transportation Science*, 49(3):500–518.
- Desrosiers, J. and Lübbecke, M. E. (2005). *Column Generation*, chapter A Primer in Column Generation, pages 1–32. Springer US.
- Dong, Q. (2005). Maximizing system lifetime in wireless sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN, Piscataway, NJ, USA.
- du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation. *Discrete Mathematics*, 194(1):229 – 237. ISSN 0012-365X.
- Edmonds, J. (1967). Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.
- Efrat, A., Har-Peled, S., and Mitchell, J. S. B. (2005). Approximation algorithms for two optimal location problems in sensor networks. In *Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on*, pages 714–723 Vol. 1.

- Feillet, D. (2010). A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407--424. ISSN 1614-2411.
- Fischetti, M., Salazar-Gonzalez, J.-J., and Toth, P. (2007). The generalized traveling salesman and orienteering problems. In Gutin, G. and Punnen, A., editors, *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*, pages 609--662. Springer US.
- F.Senne, E. L. and Lorena, L. A. N. (2000). *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, chapter Lagrangean/Surrogate Heuristics for p-Median Problems, pages 115--130. Springer US, Boston, MA.
- Gendron, B. and Larose, M. (2014). Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design. *EURO Journal on Computational Optimization*, 2(1-2):55--75.
- Gendron, B., Lucena, A., da Cunha, A. S., and Simonetti, L. (2014). Benders decomposition, branch-and-cut, and hybrid algorithms for the minimum connected dominating set problem. *INFORMS Journal on Computing*, 26(4):645--657.
- Gentili, M. and Raiconi, A. (2013). α -coverage to extend network lifetime on wireless sensor networks. *Optimization Letters*, 7(1):157--172. ISSN 1862-4480.
- Gollowitzer, S., Gendron, B., and Ljubić, I. (2013). A cutting plane algorithm for the capacitated connected facility location problem. *Computational Optimization and Applications*, 55(3):647--674. ISSN 0926-6003.
- Gollowitzer, S. and Ljubić, I. (2011). Mip models for connected facility location: A theoretical and computational study. *Computers & Operations Research*, 38(2):435--449. ISSN 0305-0548.
- Hakimi, S. L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3):450--459.
- Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13(3):462--475.
- Hang, N. and Seah, W. K. (2009). How long is the lifetime of a wireless sensor network? In *International Conference on Advanced Information Networking and Applications*, pages 763--770. ISSN 1550-445X.

- Hartmann, M. and Özgür Özlük (1999). Solving the traveling circus problem by branch-and-cut. *Electronic Notes in Discrete Mathematics*, 3:72 – 76. ISSN 1571-0653. 6th Twente Workshop on Graphs and Combinatorial Optimization.
- Hartmann, M. and Özgür Özlük (2001). Facets of the p-cycle polytope. *Discrete Applied Mathematics*, 112(1-3):147–178. Combinatorial Optimization Symposium.
- Heinzelman, W. B., Chandrakasan, A. P., and Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, volume 2, page 10.
- Heinzelman, W. B., Chandrakasan, A. P., and Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670.
- Held, M. and Karp, R. M. (1970). The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162.
- Kang, I. and Poovendran, R. (2005). Maximizing network lifetime of broadcasting over wireless stationary ad hoc networks. *Mobile Networks and Applications (MONET)*, 10(6):879–896. Special Issue on Energy Constraints and Lifetime Performance in Wireless Sensor Networks.
- Khuller, S. and Zhu, A. (2002). The general steiner tree-star problem. *Information Processing Letters*, 84(4):215 – 220. ISSN 0020-0190.
- Kim, H. S., Abdelzaher, T. F., and Kwon, W. H. (2003). Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys '03*, pages 193–204, New York, NY, USA. ACM.
- Kuorilehto, M., Hännikäinen, M., and Hämäläinen, T. D. (2005). A survey of application distribution in wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 2005(5):774–788. ISSN 1687-1472.
- Labbé, M., Laporte, G., Martín, I. R., and González, J. J. S. (2004). The ring star problem: Polyhedral analysis and exact algorithm. *Networks*, 43(3):177–189. ISSN 1097-0037.
- Labbé, M., Laporte, G., Martín, I. R., and González, J. J. S. (2005). Locating median cycles in networks. *European Journal of Operational Research*, 160(2):457–470. ISSN 0377-2217. Decision Support Systems in the Internet Age.

- Laporte, G. and Louveaux, F. V. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133--142. ISSN 0167-6377.
- Lee, Y., Lu, L., Qiu, Y., and Glover, F. (1993). Strong formulations and cutting planes for designing digital data service networks. *Telecommunication Systems*, 2(1):261--274. ISSN 1572-9451.
- Leitner, M., Ljubić, I., Salazar-Gonzalez, J.-J., and Sinnl, M. (2015). The connected facility location polytope: Valid inequalities, facets and a computational study. Technical report, Faculty of Business, Economics, and Statistics University of Vienna.
- Leitner, M., Ljubić, I., Salazar-Gonzalez, J.-J., and Sinnl, M. (2017). An algorithmic framework for the exact solution of tree-star problems. *European Journal of Operational Research*, 261(1):54 – 66. ISSN 0377-2217.
- Leitner, M. and Raidl, G. R. (2011). Branch-and-cut-and-price for capacitated connected facility location. *Journal of Mathematical Modelling and Algorithms*, 10(3):245–267. ISSN 1570-1166.
- Ljubić, I. (2007). A hybrid VNS for connected facility location. In *Hybrid Metaheuristics*, volume 4771 of *Lecture Notes in Computer Science*, pages 157--169. Springer Berlin Heidelberg.
- Lucena, A., Simonetti, L., and da Cunha, A. S. (2015). The tree-star problem: A formulation and a branch-and-cut algorithm. In *Lecture Notes in Computer Science, Proceedings of the 7th International Network Optimization Conference*.
- Maculan, N. (1987). The steiner problem in graphs. In *Annals of Discrete Mathematics*, volume 31, pages 185--212. Elsevier Science Publishers, North-Holland, Amsterdam.
- Magnanti, T. L. and Wolsey, L. A. (1995). *Handbooks in Operations Research and Management Science*, volume 7, chapter Optimal trees, pages 503--615. Elsevier.
- Marianov, V., Gutiérrez-Jarpa, G., Obreque, C., and Cornejo, O. (2012). Lagrangean relaxation heuristics for the p-cable-trench problem. *Computers & Operations Research*, 39(3):620 – 628. ISSN 0305-0548.
- Matos, V. O., Arroyo, J. E. C., dos Santos, A. G., and Goncalves, L. B. (2012). An energy-efficient clustering algorithm for wireless sensor networks. *International Journal of Computer Science and Network Security*, 12(10):6--15.

- Mladenović, N., Brimberg, J., Hansen, P., and Moreno-Pérez, J. A. (2007). The p -median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*, 179(3):927–939. ISSN 0377-2217.
- Morais, V., da Cunha, A. S., and Mahey, P. (2016a). A branch-and-cut-and-price algorithm for the stackelberg minimum spanning tree game. *Electronic Notes in Discrete Mathematics*, 52:309 – 316. ISSN 1571-0653. INOC 2015 - 7th International Network Optimization Conference.
- Morais, V., Gendron, B., and Mateus, G. R. (2015). Formulating and solving the coverage constrained p -tree problem. In *CORS–INFORMS International Meeting (abstract)*.
- Morais, V., Gendron, B., and Mateus, G. R. (2019). The p -arborescence star problem: Formulations and exact solution approaches. *Computers & Operations Research*, 102:91 -- 101. ISSN 0305-0548.
- Morais, V. and Mateus, G. R. (2019). Configuration-based approach for topological problems in the design of wireless sensor networks. *International Transactions in Operational Research*, 26(3):836–855.
- Morais, V., Mateus, G. R., and Gendron, B. (2016b). The p -cycle star problem: Formulations and cutting-plane methods. In *4th International Symposium on Combinatorial Optimization*.
- Morais, V., Souza, F. S. H., and Mateus, G. R. (2016c). *Optimization Problems, Models, and Heuristics in Wireless Sensor Networks*, chapter Handbook of Heuristics, pages 1--21. Springer International Publishing.
- Narula, S. C., Ogbu, U. I., and Samuelsson, H. M. (1977). An algorithm for the p -median problem. *Operations Research*, 25(4):709–713.
- Nguyen, V. H. and Knippel, A. (2007). On tree star network design. In *Proceedings of the 7th International Network Optimization Conference*, pages 1--6.
- Nguyen, V. H. and Maurras, J.-F. (2001). On the linear description of the k -cycle polytope. *International Transactions in Operational Research*, 8(6):673--692. ISSN 1475-3995.
- Padberg, M. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM*, 33(1):60--100.

- Pereira, D. L. and da Cunha, A. S. (2018). Reformulations and branch-and-price algorithm for the minimum cost hop-and-root constrained forest problem. *Computers & Operations Research*, 98:38 – 55. ISSN 0305-0548.
- Pessoa, A., Uchoa, E., de Aragão, M. P., and Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, 2(3–4):259–290.
- Raiconi, A. and Gentili, M. (2011). *Network Optimization: 5th International Conference, INOC 2011, Hamburg, Germany, June 13-16, 2011. Proceedings*, chapter Exact and Metaheuristic Approaches to Extend Lifetime and Maintain Connectivity in Wireless Sensors Networks, pages 607–619. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Raicu, I., Schwiebert, L., Fowler, S., and Gupta, S. K. (2005). Local load balancing for globally efficient routing in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 1(2):163–185.
- Rawat, P., Singh, K. D., Chaouchi, H., and Bonnin, J. M. (2014). Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of Supercomputing*, 68(1):1–48. ISSN 1573-0484.
- Resende, M. G. and Werneck, R. F. (2004). A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88.
- Rousseau, L.-M., Gendreau, M., and Feillet, D. (2007). Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668. ISSN 0167-6377.
- Simonetti, L., da Cunha, A. S., and Lucena, A. (2011). The minimum connected dominating set problem: Formulation, valid inequalities and a branch-and-cut algorithm. In Pahl, J., Reiners, T., and VoB, S., editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 162–169. Springer Berlin Heidelberg.
- Swamy, C. and Kumar, A. (2004). Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269. ISSN 1432-0541.
- Tansel, B. C., Francis, R. L., and Lowe, T. J. (1983). State of the art-location on networks: A survey. part i: The p-center and p-median problems. *Management Science*, 29(4):482–497.

- Vasko, F. J., Barbieri, R. S., Rieksts, B. Q., Reitmeyer, K. L., and Stott, K. L. (2002). The cable trench problem: combining the shortest path and minimum spanning tree problems. *Computers & Operations Research*, 29(5):441 – 458. ISSN 0305-0548.
- Wolsey, L. A. and Nemhauser, G. L. (1999). *Integer and Combinatorial Optimization*, chapter I.4 – Polyhedral Theory, pages 83--113. John Wiley & Sons Interscience.