O PROBLEMA DE OTIMIZAÇÃO INTEIRA 0-1 MINMAX REGRET SOB INCERTEZA INTERVALAR: COMPLEXIDADE E HEURÍSTICAS

IAGO AUGUSTO DE CARVALHO

O PROBLEMA DE OTIMIZAÇÃO INTEIRA 0-1 MINMAX REGRET SOB INCERTEZA INTERVALAR: COMPLEXIDADE E HEURÍSTICAS

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: Thiago Ferreira de Noronha
Coorientador: Christophe Duhamel

Belo Horizonte

Janeiro de 2020

IAGO AUGUSTO DE CARVALHO

THE MINMAX REGRET 0-1 INTEGER LINEAR PROGRAMMING PROBLEM UNDER INTERVAL UNCERTAINTY: COMPLEXITY AND HEURISTICS

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Thiago Ferreira de Noronha
Co-Advisor: Christophe Duhamel

Belo Horizonte
January 2020

Carvalho, Iago Augusto de.

C331m The minmax regret 0-1 integer linear programming problem under interval uncertainty [manuscrito]: complexity and heuristics / Iago Augusto de Carvalho.- 2020.
xxiv, 72 f. il.

Orientador: Thiago Ferreira de Noronha.
Coorientador: Christophe Didier Duhamel.
Tese (Doutorado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f.63-72
.
1. Computação – Teses. 2. Complexidade computacional – Teses. 3. Incerteza intervalar – Tese. 4. Heurística – Teses. I. Noronha, Thiago Ferreira de.II. Duhamel, Christophe Didier.III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. IV.Título.

CDU 519.6*61(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

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

FOLHA DE APROVAÇÃO

The Minmax regret 0-1 Integer Linear Programming Problem under Interval Uncertainty: Complexity and Heuristics

IAGO AUGUSTO DE CARVALHO

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

Thiago F. Noronha
PROF. THIAGO FERREIRA DE NORONHA - Orientador
Departamento de Ciência da Computação - UFMG

[illegible]
PROF. CHRISTOPHE DIDIER DUHAMEL - Coorientador
LITIS - Université Le Havre Normandie

[illegible]
PROF. RAFAEL CASTRO DE ANDRADE
Departamento de Estatística e Matemática Aplicada - UFC

Vinicius Fernandes dos Santos
PROF. VINÍCIUS FERNANDES DOS SANTOS
Departamento de Ciência da Computação - UFMG

[illegible]
PROF. CRISTIANO ARBEX VALLE
Departamento de Ciência da Computação - UFMG

[illegible]
PROF. PUCA HUACHI VAZ PENNA
Departamento de Computação - UFOP

Belo Horizonte, 30 de Janeiro de 2020.

Dedico este trabalho a todos os professores e cientistas que executam seu trabalho com excelência mesmo em tempos de crise

"In the middle of chaos lies opportunity."

(Bruce Lee)

Resumo

Minmax regret é um *framework* para abordar incerteza no processo de tomada de decisão. Nesta tese, nós investigamos problemas de otimização minmax regret onde o coeficiente das variáveis da função objetivo é desconhecido, mas é assumido ser restrito por um intervalo. O Problema da Programação Linear Inteira 0-1 Minmax regret sob Incerteza Intervalar (M-ILP) é investigado. Nós provamos que este problema é completo para o segundo nível da hierarquia polinomial, sendo Σ_2^p-Completo. Além disso, nós introduzimos as heurísticas *Fix-and-Optimize* (FAO), que podem ser generalizadas para qualquer problema de otimização minmax regret sob incerteza intervalar. Nós avaliamos a qualidade das heurísticas propostas realizando experimentos computacionais em duas instâncias de M-ILP: o problema da Cobertura de Conjuntos Ponderado Minmax regret sob Incerteza Intervalar e o Problema do Caminho Mais Curto de Fonte Única Minmax regret sob Incerteza Intervalar. Para o primeiro, nós mostramos que ele está contido na classe Σ_2^p. Além disso, nós extendemos algoritimos exatos baseados na Decomposição de Benders para este problema, propomos duas variantes das heurísticas FAO e comparamos os resultados obtidos com os resultados da literatura para este problema. Para o último, nós mostramos que o problema é NP-Difícil mesmo em digrafos em camadas com 3 camadas, obtemos soluções ótimas resolvendo uma formulação *multi-commodities* usando um algoritmo *branch-and-bound*, e implementamos as duas mesmas variações das heurísticas FAO. Os resultados obtidos pelas heurísticas FAO também são comparados com os resultados das heurísticas estado-da-arte para este problema. Experimentos computacionais realizados em instâncias clássicas da literatura demonstraram que uma das heurísticas FAO propostas é significativamente melhor que as heurísticas da literatura quando resolvendo ambos os problemas estudados.

Palavras-chave: Minmax regret, Incerteza intervalar, Complexidade, Cobertura de Conjuntos ponderada, Caminho Mais Curto de Fonte Única, Heurísticas.

Abstract

Minmax regret is a framework to tackle uncertainty in the decision-making process. In this thesis, we investigate minmax regret optimization problems where the coefficient of the variables on the objective function is unknown, but it is assumed to be bounded by an interval. The Minmax regret 0-1 Integer Linear Programming Problem under Interval Uncertainty (M-ILP) is investigated. We prove that this problem is complete for the second level of the polynomial hierarchy, being Σ_2^p-Complete. Furthermore, we introduce the Fix-and-Optimize (FAO) heuristics, which can be generalized for any minmax regret optimization problem under interval uncertainty. We assess the quality of the proposed heuristics by performing computational experiments on two instances of M-ILP: the Minmax regret Weighted Set Covering Problem under Interval Uncertainty and the Minmax regret Single-Source Shortest Path Problem under Interval Uncertainty. For the former, we show that it is contained in the class Σ_2^p. Furthermore, we extend exact algorithms based on the Bender's Decomposition for this problem, propose two variants of the FAO heuristics, and compare the obtained results with those of the literature for this problem. For the latter, we show that the problem is NP-Hard even on a layered digraph with 3 layers, obtain optimal solutions by solving a compact multi-commodities formulation using a branch-and-bound algorithm, and implement the same FAO heuristic variants. The results obtained by the FAO heuristics are also compared with those of the state-of-the-art heuristics for this problem. Computational experiments performed on classical instances from the literature demonstrated that one of the proposed Fix-and-Optimize heuristics significantly outperformed the literature heuristics for solving both of the studied problems.

Palavras-chave: Minmax regret, Interval uncertainty, Complexity, Weighted Set Covering, Single-source Shortest Path, Heuristics.

List of Figures

1.1 (a) A graph under interval uncertainty and (b) one possible scenario 2

4.1 (a) Scenario considered by AM and (b) scenario considered by AU 28

5.1 An WSCP instance 34
5.2 A SS-SPP instance 37
5.3 Reduction of the 2PP to the MSS-SPP. Arcs in A_0 are solid, arcs in A_1 are dashed and arcs in A_2 are dotted. 41

6.1 A layered digraph with $L = 4$ layers of $\omega = 3$ nodes 52

List of Tables

2.1 Summary of the literature review for minmax regret optimization problems under interval uncertainty 11

6.1 Results for the exact algorithms on *OR-Library* instances 45
6.2 Results for the exact algorithms on *Shunji* instances 45
6.3 Results for the FAO heuristics on *OR-Library* instances 46
6.4 Results for the FAO heuristics on *Shunji* instances 46
6.5 Results for the heuristics on *OR-Library* instances 48
6.6 Results for the heuristics on *Shunji* instances 48
6.7 p-values obtained by the Nemenyi's test on *OR-Library* and *Shunji* instances 50
6.8 Results for the MILP formulation on random graphs 54
6.9 Results for the MILP formulation on layered graphs 54
6.10 Reduction achieved by FO-SBA and FO-LR on random digraphs 54
6.11 Reduction achieved by FO-SBA and FO-LR on layered digraphs 58
6.12 Results for the heuristics on random digraphs 58
6.13 Results for the heuristics on layered digraphs 58
6.14 p-values obtained by the Nemenyi's test on random and layered digraphs . 59

List of Acronyms

2PP	2-Partition Problem	39
AMU	Algorithm Mean Upper	23
AM	Algorithm Mean	24
ATM	Alternating Turing Machine with k alternations	15
AU	Algorithm Upper	24
BC	Branch-and-Cut	23
BLD	Benders-like Decomposition	23
EB	Extended Benders	23
FAO	Fix-and-Optimize	5
FO-LR	Fix-and-Optimize through Linear Relaxation	30
FO-SBA	Fix-and-Optimize through Scenario-based Algorithm	30
ILP	0-1 Integer Linear Programming Problem	1
Knapsack	0-1 Knapsack Problem	17
LPH	Linear Programming Based Heuristic	23
LP	Linear Program	31
MP	Master problem	24
M-ILP	Minmax regret 0-1 Integer Linear Programming Problem under Interval Uncertainty	2
M-Knapsack	Minmax regret 0-1 Knapsack Problem under Interval Uncertainty	16
M-WSCP	Minmax regret Weighted Set Covering Problem under Interval Uncertainty	5
M-SPP	Minmax regret Shortest Path Problem under Interval Uncertainty	51

MRC-SPP	Minmax regret Resource-Constrained Shortest Path Problem under Interval Uncertainty 51
MILP	Mixed-Integer Linear Programming 5
MSS-SPP	Minmax regret Single-Source Shortest Path Problem under Interval Uncertainty 5
RM-ILP	Restricted Minmax regret 0-1 Integer Linear Programming Problem under Interval Uncertainty 30
RO	Robust Optimization 1
SBA	Scenario-based Algorithm 23
SS-SPP	Single-Source Shortest Path Problem 33
SP	Subproblem 24
VND	MILP-basedVariable Neighborhood Descent 55
WSCP	Weighted Set Covering Problem 33

Contents

Resumo **xiii**

Abstract **xv**

List of Figures **xvii**

List of Tables **xix**

List of Acronyms **xxi**

1 Introduction **1**

2 Related work **7**

2.1 Robust optimization criterions 7

2.2 Uncertainty sets for robust optimization problems 8

2.3 Literature review on minmax regret optimization problems under interval uncertainty 11

3 Worst-case complexity for M-ILP **15**

3.1 The Σ_2^p proof 16

3.2 The Decision M-Knapsack problem 17

3.3 The hardness proof 20

4 Algorithms for M-ILP **23**

4.1 Exact algorithms for M-ILP 23

4.1.1 Benders-like Decomposition (BLD) 23

4.1.2 Extended Benders-like Decomposition (EB) 25

4.1.3 Branch-and-Cut (BC) 25

4.2 Heuristics for M-ILP 26

4.2.1 Algorithm Mean Upper (AMU) 27

4.2.2 Scenario-Based Algorithm (SBA) 27
4.2.3 Linear Programming Heuristic (LPH) 29
4.2.4 Fix-and-Optimize through Scenario-based Algorithm (FO-SBA) 30
4.2.5 Fix-and-Optimize through Linear Relaxation (FO-LR) 30

5 Case studies 33
5.1 The Weighted Set-Covering problem 33
5.1.1 The Minmax regret Weighted Set Covering problem under Interval Uncertainties 34
5.1.2 Worst-case complexity for the Minmax regret Weighted Set Covering problem under Interval Uncertainty 35
5.2 The Single-Source Shortest Path problem 36
5.2.1 The Minmax regret Single-Source Shortest Path Problem under Interval Uncertainty 38
5.2.2 Worst-case complexity for the Minmax regret Single-Source Shortest Path Problem under Interval Uncertainty 39

6 Computational experiments 43
6.1 Experiments with the M-WSCP 43
6.1.1 Results for the exact algorithms 44
6.1.2 Results on the preprocessing step of the fix-and-optimize heuristics 45
6.1.3 Comparison of the heuristics 47
6.2 Experiments with the MSS-SPP 50
6.2.1 Results for the MILP formulation 51
6.2.2 Results on the preprocessing step of the fix-and-optimize heuristics 53
6.2.3 Comparison of the heuristics 55

7 Conclusions and future works 61

Bibliography 63

Chapter 1

Introduction

The 0-1 Integer Linear Programming Problem (ILP) is a mathematical optimization problem, with linear objective function and constraints, in which the domain of all variables is $\{0,1\}$. ILP can be formulated by the objective function (1.1) and the constraints (1.2) and (1.3), where b and c are n-dimensional vectors of coefficients, A is a $m \times n$-dimensional matrix of coefficients, and x is a n-dimensional vector of binary variables. ILP is NP-Hard [Karp, 1972]. However, in many cases it can be solved, on small- and medium-sized instances, by branch-and-bound based algorithms available in state-of-the-art ILP solvers, such as CPLEX[1] and GUROBI[2] [Anand et al., 2017].

$$\text{(ILP)} \quad \min c^T x \tag{1.1}$$

$$Ax \leq b \tag{1.2}$$

$$x \in \{0,1\}^n \tag{1.3}$$

This thesis deals with ILP problems where the coefficients in c are uncertain. There are many techniques to tackle uncertainty in the decision making process, such as Stochastic Programming [Spall, 2005], Dynamic Optimization [Kamien and Schwartz, 2012], Fuzzy Logic [Ross, 2016], and Robust Optimization [Kouvelis and Yu, 1997]. We focus on Robust Optimization (RO), which is an approach to deal with uncertainty in decision making where the variability of the data is represented as an uncertainty set.

RO was initially set 50 years ago to deal with financial problems [Gupta and Rosenhead, 1968]. It was originally applied as a way of self-protection against undesir-

[1]https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer
[2]http://www.gurobi.com

able outcomes due to uncertain or imprecise data, being later adapted to optimization problems. This term describes a wide range of methodologies for modeling uncertainty in the decision-making process. RO is specially useful when the decision maker is interested in the outcome of all potential scenarios, and not only the expected or the most likely to happen. This situation is common when the decision has to be made once and cannot be changed easily, or when the decision maker does not want to assume the risk of under-performing for some scenario.

Several models for representing uncertainty within RO problems exist [Dokka et al., 2019]. We focus on RO problems where data uncertainty is modeled as *intervals* [Kouvelis and Yu, 1997]. In this model, the cost of each coefficient c_i is uncertain, but it is assumed to be bounded by an interval $[l_i, u_i]$, with $0 \leq l_i \leq u_i$.

The resolution of a RO problem consists in optimizing a RO criterion. In this thesis, we focus on the *minmax regret criterion* for RO problems, which is the most studied RO criterion [Aissi et al., 2009; Gabrel et al., 2013]. The Minmax regret 0-1 Integer Linear Programming Problem under Interval Uncertainty (M-ILP) is formally described as follows.

Definition 1 *A solution $X = \{x_1, x_2, \ldots, x_n\}$ for both ILP and M-ILP is an attribution of values to all binary variables x.*

Definition 2 *A scenario S is an assignment of a single value $c_i^S \in [l_i, u_i]$ for each uncertain coefficient c_i, for $0 \leq i < n$.*

Figure 1.1 gives an example of a graph under interval uncertainty and one possible scenario. There are infinitely many scenarios and M-ILP aims at finding a solution that is robust against all of them.

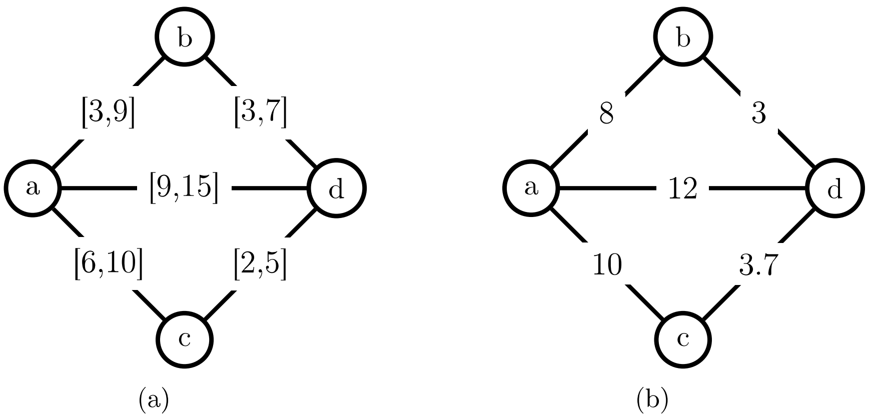

Figure 1.1. (a) A graph under interval uncertainty and (b) one possible scenario

Let Γ be the set of all scenarios, and Φ be the set of all feasible solutions to the constraints in (1.2) and (1.3).

Definition 3 *The cost of a solution $X \in \Phi$ in a specific scenario $S \in \Gamma$ is given by*

$$F(X, S) = \sum_{i=0}^{n} c_i^S x_i.$$

Definition 4 *The best possible solution $Y^S \in \Phi$ for the specific scenario $S \in \Gamma$ is*

$$Y^S = \arg\min_{Y \in \Phi} F(Y, S) = \arg\min_{Y \in \Phi} \sum_{i=0}^{n} c_i^S y_i,$$

i.e., Y^S is the optimal ILP solution for the scenario S.

Definition 5 *The regret of a solution $X \in \Phi$ in a specific scenario $S \in \Gamma$ is the difference between the cost of X and the cost of Y^S in S [Kouvelis and Yu, 1997], i.e., $F(X, S) - F(Y^S, S)$.*

Definition 6 *The robustness cost $Z(X)$ of a solution $X \in \Phi$ is defined as the maximum regret of X among all scenarios in Γ, i.e.,*

$$Z(X) = \max_{S \in \Gamma} \left\{ F(X, S) - F(Y^S, S) \right\}.$$

Despite the fact that $|\Gamma| = \infty$, it is proven in Aissi et al. [2009]; Averbakh [2001] that the scenario on which the regret of $X \in \Phi$ is the maximum is the scenario $S^X \in \Gamma$, such that

$$c_i^{S^x} = l_i + (u_i - l_i)x_i,$$

i.e., $c_i^{S^X} = u_i$ if $x_i = 1$, and $c_i^{S^X} = l_i$ otherwise. From this result, we have that

$$F(X, S^X) = \sum_{i=1}^{n} u_i x_i,$$

$$F(Y^{S^X}, S^X) = \min_{Y \in \Phi} \sum_{i=1}^{n} \big(l_i + (u_i - l_i)x_i\big) y_i,$$

and

$$Z(X) = F(X, S^X) - F(Y^{S^X}, S^X).$$

We refer to this scenario as the *worst-case scenario* induced by solution X.

It is worth noticing that $F(Y^{S^X}, S^X)$ is still an ILP as in this case x_i is constant. Therefore, the robust cost of a solution X can be computed by solving a single ILP problem in the scenario S^X. M-ILP aims at finding the solution with minimum robustness cost, *i.e.*,

$$\min_{X\in\Phi} Z(X) = \min_{X\in\Phi} \left\{ F(X, S^X) - F(Y^{S^X}, S^X) \right\}.$$

This problem can be rewritten as the 0-1 Bilevel Integer Linear Program in Equation (1.4) by replacing $F(X, S^X)$ and $F(Y^{S^X}, S^X)$.

$$\text{(M-ILP)} \qquad \min_{X\in\Phi} \left\{ \sum_{i=1}^{n} u_i x_i - \min_{Y\in\Phi} \sum_{i=1}^{n} \big(l_i + (u_i - l_i) x_i\big) y_i \right\} \tag{1.4}$$

A ILP formulation for M-ILP can obtained by replacing $F(Y^{S^X}, S^S)$ with a free variable θ and adding a new set of linear constraints that bounds the value of θ to the value of $F(Y^{S^X}, S^X)$. The resulting formulation (1.5)-(1.9) has an exponentially large number of constraints (1.6). It is worth noticing that the constraints in (1.6) are linear as, in this case, y_i is constant. As far as we can tell, in the general case, there is no technique in the literature to obtain a compact formulation for any minmax regret optimization problem under interval uncertainty whose classical counterpart does not have a formulation with the integrality property.

$$\min_{X\in\Phi} \sum_{i=0}^{n} u_i x_i - \theta \tag{1.5}$$

$$\theta \leq \sum_{i=0}^{n} (l_i + (u_i - l_i) x_i)\, y_i, \quad \forall\, Y \in \Phi \tag{1.6}$$

$$\sum_{i=0}^{n} a_{ji} x_i \leq b_j, \quad \forall j \in \{0, 1, \ldots, m\} \tag{1.7}$$

$$x_i \in \{0, 1\}, \quad \forall i \in \{0, 1, \ldots, n\} \tag{1.8}$$

$$\theta \geq 0 \tag{1.9}$$

All minmax regret combinatorial optimization problems under interval uncertainty in the literature can be modeled as a M-ILP [Kouvelis and Yu, 1997; Aissi et al., 2009; Kasperski and Zieliński, 2016]. Examples of such problems are the Minmax regret Minimum-Spanning Tree problem under Interval Uncertainty [Aron and Hentenryck, 2004; Montemanni, 2006], the Minmax regret Assignment problem under Interval Uncertainty [Pereira and Averbakh, 2011], the Minmax regret Shortest Path problem under Interval Uncertainty [Karaşan et al., 2001; Montemanni et al., 2004], the Minmax regret Traveling Salesman problem under Interval Uncertainty [Montemanni et al., 2007], the Minmax regret Knapsack problem under Interval Uncertainty [Deineko and Woeginger, 2010; Furini et al., 2015], and the Minmax regret Weighted Set Covering problem under Interval Uncertainty [Pereira and Averbakh, 2013; Coco et al., 2016]. These problems are NP-Hard, even those whose classical combinatorial optimization problem is solvable in polynomial-time. For those whose classical combinatorial optimization are NP-Hard, it is NP-Hard even to compute the robustness cost of a solution.

In this thesis, we propose the Fix-and-Optimize (FAO) heuristics. These heuristics are based on the Fix-and-Optimize framework of Gintner et al. [2005], which was originally employed for solving a variant of a vehicle routing problem. This framework consists of two steps: (*i*) the preprocessing step, where a heuristic is used to fix the value some variables in the Mixed-Integer Linear Programming (MILP) formulation of the problem; and (*ii*) the solving step, where the resulting formulation is solved without the fixed variables. This framework was successfully adapted to several other classical combinatorial optimization problems, such as fleet sizing [Dastjerd and Ertogral, 2019], scheduling [Guo et al., 2018; Turhan and Bilgen, 2017], lot sizing [Chen, 2015], and timetabling [Dorneles et al., 2014]. However, as far as we can tell, FAO heuristics has never been applied to minmax regret optimization problems.

In this thesis, we show that the decision version of M-ILP lies on the second level of the polynomial hierarchy, being Σ_2^p-Complete [Stockmeyer, 1976]. Furthermore, we perform case studies with two instances of M-ILP problems: (*i*) the Minmax regret Weighted Set Covering Problem under Interval Uncertainty (M-WSCP); and (*ii*) the Minmax regret Single-Source Shortest Path Problem under Interval Uncertainty (MSS-SPP). For M-WSCP, we prove that this problem is on the second class of the polynomial hierarchy, being Σ_2^p. Furthermore, we solve the ILP formulation proposed by Pereira and Averbakh [2013] for M-WSCP through three exact algorithms based on the Bender's decomposition [Benders, 1962] and implement two FAO variants, comparing their results with those given by the heuristics on the literature for this problem. For MSS-SPP, we prove that this problem is NP-Hard. Furthermore, we solve the compact ILP formulation proposed by Catanzaro et al. [2011] through a branch-and-bound algorithm and implement the same two FAO heuristics, comparing their results with those given by the heuristics on the literature for this problem.

The remaining of this thesis is organized as follows. Initially, Chapter 2 presents a literature review on RO. A proof for the worst-complexity of the M-ILP is given in

Chapter 3. Next, Chapter 4 shows the most prominent exact and heuristic algorithms from the literature to solve minmax regret optimization problems under interval uncertainty and proposes the FAO heuristics. Then, Chapter 5 formally defines M-WSCP and MSS-SPP, also presenting their ILP formulations and complexity proofs. Finally, an extensive set of computational experiments is presented and discussed on Chapter 6 and the conclusions of this thesis are drawn in the last chapter.

Chapter 2

Related work

This chapter provides a literature review on three topics. The first topic, presented in Section 2.1, reviews the most well-known RO criteria for problems under interval uncertainty. The second topic is presented in Section 2.2 and describes the most prominent models for representing uncertainty within the RO framework. The third topic, shown in Section 2.3, reviews the literature for works on minmax regret optimization problems under interval uncertainty.

2.1 Robust optimization criterions

The most well-known RO criterions for problems under interval uncertainty are the Minmax, the Minmax regret, and the Minmax relative regret [Kouvelis and Yu, 1997]. The criterions vary on the level of conservativeness of the resulting model. Furthermore, a RO problem solved with different criteria may have different worst-case complexity. For example, the Shortest Path Problem under Interval Uncertainty, which is solvable in polynomial-time with the minmax criterion [Aissi et al., 2009] and is NP-Hard with the minmax regret criterion [Kouvelis and Yu, 1997] and with the minmax relative regret criterion [Averbakh, 2005a]. We refer to the book by Kouvelis and Yu [1997] for a comprehensive introduction to the Minmax, the Minmax regret, and to the Minmax relative regret criterions. In addition, we refer to Coco et al. [2014b] for a survey on other criteria for RO problems.

The Minmax creterion is the most conservative one. It aims at finding a solution that has the smallest worst case value, *i.e.*, it finds a solution $X^* \in \Phi$ such that

$$X^* = \arg\min_{X \in \Phi} F(X, S^X) = \arg\min_{x \in \Phi} \sum_{i=1}^{n} u_i x_i.$$

Minmax optimization problems under interval uncertainty have the same complexity of the underlying deterministic problem, since it solves an instance of the deterministic problem in the upper scenario $s^u \in \Gamma$, where the cost of each uncertain coefficient is set to its upper value, *i.e.* $c_i^{s^u} = u_i$, for all uncertain coefficients c_i.

The Minmax regret criterion, as shown in Chapter 1, is the most used criterion for RO problems under interval uncertainty. It aims at finding a solution that has the smallest regret, *i.e.*, it finds a solution X^* such that

$$X^* = \arg\min_{X \in \Phi} \left\{ F(X, S^X) - F(Y^{S^X}, S^X) \right\}.$$

The complexity of Minmax regret optimization problems under Interval Uncertainty greatly varies, as shown in Section 2.3. However, most of the problems within this class are known to be NP-Hard [Kouvelis and Yu, 1997; Aissi et al., 2009].

The Minmax relative regret is the less conservative one among the three reviewed criterions [Averbakh, 2005b]. It aims at finding a solution X^* such that

$$X^* = \arg\min_{x \in \Phi} \left\{ \frac{F(X, \underline{S}^X) - F(Y^{\underline{S}^X}, \underline{S}^X)}{F(X, \underline{S}^X)} \right\}.$$

As shown by Coco et al. [2017], the robust deviation of a minmax relative regret problem under interval uncertainty is not the maximum in the worst-case scenario induced by solution X, *i.e.*, exists a scenario $\underline{S} \in \Gamma, \underline{S} \neq S^X$ such that the relative regret is the minimum. This scenario can be computed using a binary search algorithm, as pointed out by Averbakh [2000].

2.2 Uncertainty sets for robust optimization problems

There exists different possibilities for expressing data uncertainty within the RO framework. The uncertainty on cost coefficients c_i can be modeled through a given uncertainty set $\mathfrak{U}$ and the decision maker is interested in computing a solution that is protected against all possible outcomes without further knowledge of a probability distribution.

For the sake of simplicity, in this section we will describe the different uncertainty sets using the minmax criterion as an example. Let

$$\min \sum_{i=0}^{n} c_i x_i$$

denote the objective function of ILP, as described in Equation (1.1). Let $C = \{c_1, c_2, \ldots, c_n\}$ be a vector of cost coefficients associated with the variables X. A minmax optimization problem model can be expressed as

$$\min \max_{C \in \mathfrak{U}} \sum_{i=0}^{n} c_i x_i.$$

In RO problems, it is commonly assumed that the shape of the uncertainty set $\mathfrak{U}$ is known in advance. In Chapter 1, we described the case where $\mathfrak{U}$ is represented by intervals. However, the shape of $\mathfrak{U}$ can also be discrete, ellipsoidal, budgeted, or polyhedral, among other possibilities [Dokka et al., 2019]. They are presented below.

Interval uncertainty sets [Kouvelis and Yu, 1997; Aissi et al., 2009] represent uncertain data as bounded intervals. In this case,

$$\mathfrak{U} = \{C : c_i \in [l_i, u_i]\},$$

where $0 \leq l_i \leq u_i$. The worst-case complexity of minmax regret optimization problems under interval uncertainty greatly varies and will be discussed in details on Section 2.3.

The discrete case explicitly shows the uncertain data [Kouvelis and Yu, 1997]. In this case, one can represent

$$\mathfrak{U} = \left\{C^1, C^2, \ldots, C^k\right\},$$

where each vector of cost coefficients $C^i \in \mathfrak{U}$ represents a possible observation of the uncertain data. In the case that $|\mathfrak{U}|$ is known, many polynomial-solvable problems turns out to be NP-Hard. In addition, if $|\mathfrak{U}|$ is unbounded or given as part of the input of the problem, in most of the cases, the underlying RO problem is NP-Hard in the strong sense [Kouvelis and Yu, 1997].

Ellipsoidal uncertainty sets [Ben-Tal and Nemirovski, 1998, 1999] can be described as

$$\mathfrak{U} = \left\{C \in \mathbb{R}^n_{\geq 0} : (C - \hat{C})\Sigma^{-1}(C - \hat{C}) \leq r^2\right\},$$

where $\hat{C}$ is the expected value of the uncertain cost coefficients C and Σ is the covariance matrix given by the values in C. Furthermore, parameter r represents the risk that the decision maker is willing to assume, such that as greater is the value of r, then more conservative is the computed solution. The worst-case complexity of minmax regret optimization problems under ellipsoidal uncertainty are not greater than that of the deterministic problem at hand, as this representation imposes only a polynomial number of additional convex constraints.

Budgeted uncertainty sets [Bertsimas and Sim, 2003, 2004] were based on interval uncertainty sets. The former was developed in an attempt to reduce the conservatism level of the latter. It assumes that at most $\Delta < n$ values of the cost coefficients in C are greater than the midpoint value, *i.e.*, that at most Δ coefficients c_i are greater than $c_i^m = \frac{(l_i+u_i)}{2}$. One can state the budgeted uncertainty set as

$$\mathfrak{U} = \left\{ C : c_i = c_i^m + (u_i - c_i^m)\delta_i, \forall c_i \in C, 0 \leq \delta_i \leq 1, \sum_{i=0}^{n} \delta_i \leq \Delta \right\}.$$

The resolution of a minmax regret optimization problem under budgeted uncertainty involves solving $\mathcal{O}(n)$ times the deterministic problem with different cost coefficients. Therefore, the budgeted variant of deterministic polynomial-time problems still can be efficiently solved [Dokka et al., 2019].

Polyhedral uncertainty sets are defined as

$$\mathfrak{U} = \{C : VC \leq e, C \geq 0\},$$

where V is a $m \times n$-dimensional matrix of coefficients and e is a n-dimensional vector of coefficients. Thus, the uncertainty is contained within a polyhedron $\mathcal{P}$ defined by constraints $VC \leq e$, which can assume different formats (*e.g.*, a knapsack [Poss, 2018] or a continuous multidimensional knapsack problem [Minoux, 2009]). Minmax regret optimization problems under polyhedral uncertainty are likely to be NP-Hard, as pointed out by Buchheim and Kurtz [2018].

In a recent study, Dokka et al. [2019] proposed the use of multiple uncertainty sets within a RO problem. The authors build a new objective function

$$\min_{X \in \Phi} \sum_{u \in U} p_u \max_{C \in \mathfrak{U}_u} \sum_{i=0}^{n} c_i x_i,$$

where U is a set of uncertainty sets and p_u is a weight which gives the importance of uncertainty set $\mathfrak{U}_u$. Thee authors argue that using multiple uncertainty sets in conjoint can better represent the uncertain data and produce solutions that are more robust against data variability.

Table 2.1. Summary of the literature review for minmax regret optimization problems under interval uncertainty

Problem name	Complexity
Linear Programming	Strongly NP-Hard [Averbakh and Lebedev, 2005]
Shortest Path	Strongly NP-Hard [Averbakh and Lebedev, 2004]
Single-Source Shortest Path	**NP-Hard**
Minimum Spanning Tree	Strongly NP-Hard [Averbakh and Lebedev, 2004]
Assignment	Strongly NP-Hard [Aissi et al., 2005a]
Min Cut	Polynomial [Aissi et al., 2005b]
Min s-t Cut	Strongly NP-Hard [Aissi et al., 2005b]
Selection	Polynomial [Averbakh, 2001]
Representatives Selection	Polynomial [Dolgui and Kovalev, 2012]
Weighted Independent Set	NP-Hard [Kasperski and Zieliński, 2015]
1-median (edge uncertainty)	Strongly NP-Hard [Averbakh, 2003]
1-median (node uncertainty)	Polynomial [Averbakh and Berman, 2000b]
1-center (edge uncertainty)	Strongly NP-Hard [Averbakh, 2003]
1-center (node uncertainty)	Polynomial [Yu et al., 2008]
Scheduling $1\|\|\Sigma C_j$	NP-Hard [Lebedev and Averbakh, 2006]
0-1 Integer Programming	**$\boldsymbol{\Sigma_2^p}$-Complete**
Restricted Shortest Path	NP-Hard
Traveling Salesman Problem	NP-Hard
Weighted Set Covering	**$\boldsymbol{\Sigma_2^p}$**, NP-Hard
0-1 Knapsack	Σ_2^p-Complete [Deineko and Woeginger, 2010]

2.3 Literature review on minmax regret optimization problems under interval uncertainty

Minmax regret optimization problems under interval uncertainty were widely studied in the literature. This class of problems were studied from the theoretical point, where one is interested in determining the problem's complexity, and from a algorithmic view, where one is interested in proposing algorithms for the resolution of problems within this class. This section reviews the most well-known works from the literature regarding minmax regret optimization problems under interval uncertainty, which are summarized in Table 2.1. This table shows, in the first column, the name of the problem and the complexity of the problem in the second column. The results obtained in this thesis are highlithed in bold.

A general result for minmax regret optimization problems is that the uncertain variant is, at least, as harder to solve than the deterministic version of the problem [Kasperski and Zieliński, 2016]. This is due to the fact that, for computing the minmax regret of a solution, one must solve the deterministic problem on its worst-case induced scenario, as presented in Definition 5.

The Minmax regret Linear Programming Problem under Interval Uncertainty was shown to be strongly NP-Hard by Averbakh and Lebedev [2005]. This problem was solved by the repetitional use of the simplex method in Inuiguchi and Sakawa [1995]. This algorithm was later improved by Mausser and Laguna [1998] using a stronger ILP formulation than that of Inuiguchi and Sakawa [1995].

The Minmax regret Shortest Path Problem under Interval Uncertainty was first proposed in Kouvelis and Yu [1997] and was shown to be Strongly NP-Hard by Averbakh and Lebedev [2004]. Later, this problem was shown to be NP-Hard even on series-parallel digraphs by Kasperski and ZielińSki [2006]. The first ILP formulation for this problem was proposed by Yu and Yang [1998]. A branch-and-bound algorithm and a Benders-like decomposition algorithm were presented in Montemanni et al. [2004] and Montemanni and Gambardella [2005], respectively. An scenario-based heuristic was applied for this problem by Kasperski and Zieliński [2007]. Two preprocessing algorithms for fixing arcs were proposed by Catanzaro et al. [2011]. Most recently, Pérez-Galarce et al. [2018] proposed a Branch-and-Cut algorithm for this problem along with a local search and a simulated annealing metaheuristics.

The Minmax regret Single-Source Shortest Path under Interval Uncertainty was first studied by Neumann [2010]. An ILP formulation for this problem was also given by Neumann [2010]. Later, Carvalho et al. [2016] extended two scenario-based heuristics from the literature to this problem, while Carvalho et al. [2018] proposed the use of a Variable Neighborhood Descent algorithm where the local search is performed by solving an ILP formulation. In this thesis, we show that this problem is NP-Hard.

The Minmax regret Minimum Spanning Tree Problem under Interval Uncertainty was shown to be Strongly NP-Hard by Averbakh and Lebedev [2004]. An ILP formulation for this problem was presented in Yaman et al. [2001]. Furthermore, preprocessing algorithms for fixing variables for this problem were presented by Salazar-Neumann [2007]. Regarding exact algorithms, we can cite the branch-and-bound of Montemanni and Gambardella [2005] and algorithms based on Benders' decomposition by Montemanni [2006] and Pérez-Galarce et al. [2014]. Heuristics and metaheuristics for this problem include the simulated annealing by Nikulin [2008], a tabu-search heuristic by Kasperski et al. [2012], and a scenario-based heuristic by Goncalves et al. [2017].

The Minmax regret Assignment Problem under Interval Uncertainty was shown to be strongly NP-Hard by Aissi et al. [2005a]. Two ILP formulations for this problem were proposed respectively by Kasperski and Zielinski [2004] and Pereira and Averbakh [2011]. Several exact and heuristic algorithms for this problem were also proposed by Pereira and Averbakh [2011], which includes an exact algorithm based on Benders' decomposition, a Variable Neighborhood Search, and two hybrid population-based metaheuristics. This problem was also solved through a scenario-based heuristic by Goncalves et al. [2017].

The Minmax regret Min Cut Problem under Interval Uncertainty was shown to be polynomially solvable by Aissi et al. [2005b]. However, a variant of this problem, the Minmax regret Min st Cut under Interval Uncertainty, was shown to be strongly NP-Hard by Aissi et al. [2005b].

The Minmax regret Selection Problem under Interval Uncertainty was shown to be polynomial-time solvable by Averbakh [2001]. The authors also proposed a polynomial-time algorithm for this problem. Furthermore, another selection problem, denominated Minmax regret Representative Selection Problem under Interval Uncertainty, was studied by Dolgui and Kovalev [2012], on which the authors presented a polynomial-time algorithm for this problem.

The Minmax regret Maximum Weighted Independent Set Problem under Interval Uncertainty was shown to be NP-Hard by Kasperski and Zieliński [2015]. A 2-approximation algorithm for this problem was also given by Kasperski and Zieliński [2015]. Despite that, no ILP formulation, exact, or heuristic algorithms for this problem were proposed in the literature.

Two variants of the Minmax regret 1-median Problem under Interval Uncertainty were studied in the literature. This problem was shown to be strongly NP-Hard if the uncertainty is located on the edges of the graph [Averbakh, 2003], but solvable in polynomial-time if the uncertainty is located on the nodes [Averbakh and Berman, 2000b]. For the latter problem, a polynomial-time algorithm was given by Averbakh and Berman [2000b], which was later improved by Yu et al. [2008]. The former problem was also studied when the underlying instance is a tree, on which it is solvable in polynomial-time. In this case, a first polynomial-time was proposed by Chen and Lin [1998] and later improved by Averbakh [2003].

Two variants of the Minmax regret 1-center Problem under Interval Uncertainty were studied in the literature. This problem was shown to be strongly NP-Hard if the uncertainty is located on the edges of the graph [Averbakh, 2003], but solvable in polynomial-time if the uncertainty is located on the nodes [Averbakh and Berman, 2000b]. The former problem was also shown to be solvable in polynomial-time in a tree [Averbakh and Berman, 2003]. In the case that uncertainty is presented in both the edges and the nodes of the graph and the underlying instance is a tree, the problem was shown to be polynomial-time solvable. A first algorithm for this case was presented by Averbakh and Berman [2000a], which was later improved by Burkard and Dollani [2002] using techniques of computational geometry.

A Minmax regret Scheduling Problem under Interval Uncertainty was studied by Lebedev and Averbakh [2006]. This problem consists in scheduling a set of jobs in a single machine, such that the time of each job is uncertain and the objective is to minimize the maximum regret of the total flow time. For the general case, this problem was shown to be NP-Hard. However, the authors presented specific cases where this problem is polynomial-time solvable. An ILP formulation for this problem, along with a branch-and-bound algorithm and two heuristics, were proposed by Daniels and Kouvelis [1995].

The Minmax regret Restricted Shortest Path Problem under Interval Uncertainty was only studied by Assunção et al. [2017]. The authors proposed an ILP formulation for the problem and two heuristics, whereas one was a scenario-based algorithm and the other was based on linear programming. The proposed ILP formulation was solved by an exact algorithm based on Benders' decomposition. Computational experiments demonstrated that the linear programming-based heuristic outperformed the scenario-based heuristic.

The Minmax regret Traveling Salesman Problem under Interval Uncertainty was first studied by Montemanni et al. [2006], where the authors proposed two scenario-based heuristics and a preprocessing algorithm. Montemanni et al. [2007] proposed an ILP formulation for this problem, along with three exact algorithms. The first was a branch-and-bound algorithm, while the others were based on Benders' decomposition.

The Minmax regret Weighted Set Covering Problem under Interval Uncertainty was first studied by Pereira and Averbakh [2013]. The authors proposed an ILP formulation for this problem, which was solved by three algorithms based on Benders' decomposition. Several heuristics were also proposed for this problem. A Genetic Algorithm and a Hybrid Genetic Algorithm were proposed by Pereira and Averbakh [2013], a scenario-based heuristic with a path-relinking strategy was proposed by Coco et al. [2015], while a hybrid heuristic based on the Benders' decomposition algorithm and a scenario-based heuristic were developed by Coco et al. [2016].

The Minmax regret 0-1 Knapsack Problem under Interval Uncertainty was studied by Deineko and Woeginger [2010] and by Furini et al. [2015]. The former work demonstrated that this problem is complete for the second level of the polynomial hierarchy, being Σ_2^p-Complete. This result implies that the problem is not in the class NP unless the polynomial hierarchy do not collapses to some finite level, which is almost as unlikely as P = NP. The former work developed four exact algorithms for this problem based on Benders' decomposition, as so as an Iterated Local Search metaheuristic.

Chapter 3

Worst-case complexity for M-ILP

An Alternating Turing Machine with k alternations (ATM) [Chandra and Stockmeyer, 1976] is a non-deterministic Turing Machines whose states are divided into existential states ($\exists$) and universal states ($\forall$). Let $\mathfrak{D}$ be a decision problem. The existential state relates to the definition of NP: if there is a solution for $\mathfrak{D}$ such with answer YES, then the computation is accepted. The universal state relates to the definition of co-NP: if all possible solutions for $\mathfrak{D}$ have answer YES, then the computation is accepted. An ATM alternates between the existential state and the universal state k times, and it has a answer YES if and only if all of its states have answer YES.

The Polynomial Hierarchy [Stockmeyer, 1976] is a hierarchy of complexity classes which generalizes the classes P, NP, and co-NP to oracle machines. We must define

$$\Delta_0^p = \Sigma_0^p = \Pi_0^p = P,$$

where P is the set of all decision problems which can be solved in polynomial-time by a deterministic Turing Machine. Then, for $i > 0$, we have that

$$\Delta_{i+1}^p = \mathrm{P}^{\Sigma_i^p},$$

$$\Sigma_{i+1}^p = \mathrm{NP}^{\Sigma_i^p},$$

and

$$\Pi_{i+1}^p = \text{co-NP}^{\Pi_i^p},$$

where $\mathrm{P}^{\mathfrak{L}}$ is the set of decision problems solved in polynomial-time by a Turing Machine coupled with a non-deterministic oracle, $\mathrm{NP}^{\mathfrak{L}}$ is the set of decision problems solved in non-polynomial time by a Turing Machine coupled with a non-deterministic oracle, and co-$\mathrm{NP}^{\mathfrak{L}}$ is the analogue for co-NP problems. One may note that NP $= \Sigma_1^p$, co-NP $= \Pi_1^p$, and $\Delta_2^p = \mathrm{P}^{\mathrm{NP}}$ defines the class of problems solved by a Turing Machine coupled with a non-deterministic oracle for some NP-Complete problem.

Decision problems in class Σ_k^p or Π_k^p are the set of problems solved in polynomial-time by ATM (Theorem 17.8 of Papadimitriou [1994]). Let $\mathfrak{D}$ be a decision problem whose variables are partitioned into k sets $D_1, D_2, \ldots, D_k$. Problem $\mathfrak{D}$ is said to be in Σ_k^p if there is an ATM in the form

$$\exists D_1 \forall D_2 \exists D_3 \ldots f(D_1, D_2, \ldots, D_k),$$

i.e., there is an assignment of values for variables in D_1, such that, for all assignments of values in D_2, there exists an assignment of values to variables in D_3, ... such that function $f(D_1, D_2, \ldots, D_k)$ can be evaluated in polynomial-time and returns an answer YES? Similarly, a problem in said to be in Π_{k+1}^p if there is an ATM in the form

$$\forall D_1 \exists D_2 \forall D_3 \ldots f(D_1, D_2, \ldots, D_k),$$

such that function $f(D_1, D_2, \ldots, D_k)$ can be evaluated in polynomial-time..

A problem is said to be Σ_k^p-Complete (or Π_k^p-Complete) if every other problem in Σ_k^p (or Π_k^p) can be reduced to it in polynomial-time. A compedium of problems known to be complete for the second or higher levels of the polynomial hierarchy can be found in Schaefer and Umans [2002]. Furthermore, a complete and precise introduction to the polynomial hierarchy is given by Stockmeyer [1976].

In this chapter, we prove that the decision version of M-ILP is Σ_2^p-Complete. First, we show that it is in the class Σ_2^p. Next, we show that M-ILP is complete for the class Σ_2^p through a reduction from the Minmax regret 0-1 Knapsack Problem under Interval Uncertainty (M-Knapsack), which is complete for this class [Deineko and Woeginger, 2010]. For this proof, we define the decision version of M-Knapsack, propose a reduction from the decision version of M-Knapsack to the decision version of M-ILP, and show that a solution for the former is valid if and only if it is also valid for the latter.

3.1 The Σ_2^p proof

DECISION M-ILP

Input: A n-dimensional vector of coefficients b, a $m \times n$-dimensional matrix of coefficients A, a n-dimensional vector of binary variables x, two n-dimensional vector

of coefficients l and u, with $l_i \leq u_i, \forall i$, and a minimization problem as defined by Equations (1.5)–(1.9).
Question: Is there a solution $X \in \Phi$ for M-ILP such that $Z(X) \leq \zeta$?

Lemma 1 *Problem* DECISION M-ILP *lies in class* Σ_2^p.

Proof. From Definition 6, we recall that $Z(X) = \max\limits_{S \in \Gamma} \left\{ F(X,S) - F(Y^S,S) \right\}$. We can rewrite this equation as

$$Z(X) = \left\{ \sum_{i=1}^{n} u_i x_i - \min_{Y \in \Phi} \sum_{i=1}^{n} \big(l_i + (u_i - l_i) x_i\big) y_i \right\}.$$

Thus, the question for DECISION M-ILP is: exists a solution $X \in \Phi$ such that $Z(X) \leq \zeta$? One may observe that this question is on the form $\exists x \forall y f(x,y)$, where the Boolean predicate $f(x,y)$ can be evaluated in polynomial time. Theorem 17.8 from Papadimitriou [1994] states that problems with an existential quantifier followed by an universal quantifier (which is the exact form of the question stated above) are in the complexity class Σ_2^p, which completes the proof. $\square$

If P $\neq$ NP, we can assume that the polynomial hierarchy do not collapses to the second level. Therefore, we can state the following corollary.

Corollary 2 *Problem* DECISION M-ILP *is not contained in class NP.*

Furthermore, by using the Kannan's Theorem [Kannan, 1982], we can state the following corollary.

Corollary 3 *Do not exists a polynomial-sized formulation for M-ILP.*

The latter corollary can be easily verifiable. If there is a polynomial-sized formulation for M-ILP, then there exists an algorithm which verifies the cost of its solution in polynomial-time and, thus, it is in NP. This is clearly a contradiction, since problems in the second level or higher of the polynomial-hierarchy are not contained in NP [Papadimitriou, 1994].

3.2 The Decision M-Knapsack problem

The 0-1 Knapsack Problem (Knapsack) [Martello et al., 1999] is a classic NP-Hard combinatorial optimization problems. In this problem, we are given a set of items $\mathcal{H}$, such that each item $h \in \mathcal{H}$ is associated with an weight $w_h \geq 0$ and a profit p_h. The objective is to choose $I \subseteq H$ such that $\sum_{h \in I} w_h$ is smaller than a predefined capacity T and that $\sum_{h \in I} p_h$ is the maximum.

Knapsack can be formulated as following. It uses decision variables $\{ z_h \in \mathbb{B} : h \in \mathcal{H} \}$ such that $z_h = 1$ if item $h \in \mathcal{H}$ belongs to the optimal solution I and $z_h = 0$ otherwise. The Knapsack formulation is given by objective function (3.1) and the constraints in (3.2)–(3.3).

$$\max \sum_{h \in \mathcal{H}} p_h z_h \tag{3.1}$$

$$s.t. \sum_{h \in \mathcal{H}} w_h z_h \leq T \tag{3.2}$$

$$z_h \in \{0, 1\}, \qquad \forall h \in \mathcal{H} \tag{3.3}$$

M-Knapsack, as defined by Deineko and Woeginger [2010], is a RO variant of Knapsack. This problem was shown to be in the second layer of the polynomial hierarchy, being Σ_2^p-Complete [Deineko and Woeginger, 2010]. In M-Knapsack, the exact profit p_h of items $h \in \mathcal{H}$ is unknown, but it is assumed to be bounded by an interval $[l_h, u_h]$, with $0 \leq l_h \leq u_h$. The formal definition of M-Knapsack is given below. It is closely related to the definition of M-ILP presented in Chapter 1.

Definition 7 *A* solution $I \subseteq \mathcal{H}$ *for both Knapsack and M-Knapsack is a subset of items such that* $\sum_{h \in I} w_h \leq T$.

Definition 8 *A* scenario L *is an assignment of a single value* $p_h^L \in [l_h, u_h]$ *for all uncertain profits associated with items* $h \in \mathcal{H}$.

Let Λ be the set of all scenarios, and Υ be the set of all feasible solutions to the problem.

Definition 9 *The* value *of a solution* $I \in \Upsilon$ *in a specific scenario* $L \in \Lambda$ *is given by*

$$F(I, L) = \sum_{h \in I} p_h^L.$$

Definition 10 *The best possible solution* $J^L \in \Upsilon$ *for the specific scenario* $L \in \Lambda$ *is*

$$J^L = \arg\max_{J \in \Upsilon} F(J, L) = \arg\max_{J \in \Upsilon} \sum_{h \in J} p_h^L,$$

i.e., J^L *is the optimal Knapsack solution for the scenario* L.

Definition 11 *The* regret *of a solution $I \in \Upsilon$ in a specific scenario $L \in \Lambda$ is the difference between the value of J^L in L and the value of I [Kouvelis and Yu, 1997], i.e., $F(J^L, L) - F(I, L)$.*

Definition 12 *The* robustness cost *$Z(I)$ of a solution $I \in \Upsilon$ is defined as the maximum regret of I among all scenarios in Λ, i.e.,*

$$Z(I) = \max_{L\in\Lambda} \left\{F(J^L, L) - F(I, L)\right\}.$$

Despite the fact that $|\Lambda| = \infty$, it is proven in Furini et al. [2015] that the scenario where the regret of $I \in \Upsilon$ is the maximum is the scenario $L^I \in \Lambda$, such that

$$\begin{cases} c_h^{L^I} = l_h, & if \text{ item } h \in I \\ c_h^{L^I} = u_h, & otherwise \end{cases}.$$

From this result, we have that

$$F(I, L^I) = \sum_{i\in I} l_h,$$

$$F(J^{L^I}, L^I) = \max_{J\in\Upsilon} \left(\sum_{h\in J} u_h - \sum_{h\in\mathcal{H}\setminus J} l_h \right),$$

and

$$Z(I) = F(J^{L^I}, L^I) - F(I, L^I).$$

We refer to this scenario as the *worst-case scenario* induced by solution I.

One can note that the robust cost of a solution I can be computed by solving a single Knapsack problem in the scenario L^I. M-Knapsack aims at finding the solution with minimum robustness cost, *i.e.*,

$$\min_{I\in\Upsilon} Z(I) = \min_{I\in\Upsilon} \left\{ F(J^{L^I}, L^I) - F(I, L^I)) \right\}.$$

This problem can be rewritten as the 0-1 Bilevel Integer Linear Program in (3.4) by replacing $F(J^{L^I}, L^I)$ and $F(I, L^I)$.

$$\text{(M-Knapsack)} \qquad \min_{I\in\Upsilon} \left\{ \max_{J\in\Upsilon} \left(\sum_{h\in J} u_h - \sum_{h\in\mathcal{H}\setminus J} l_h \right) - \sum_{h\in I} l_h \right\} \tag{3.4}$$

Let $\{ z_h \in \mathbb{B} : h \in \mathcal{H} \}$ be such that $z_h = 1$ if item $h \in \mathcal{H}$ belongs to the optimal solution of problem (3.4) and $z_h = 0$ otherwise. Furthermore, let $\{ j_h \in \mathbb{B} : h \in \mathcal{H} \}$ be such that $j_h = 1$ if item $h \in \mathcal{H}$ belongs to the optimal solution of $F(J^{L^I}, L^I)$ and $j_h = 0$ otherwise. We can rewrite problem (3.4) as in Equation (3.5).

$$\text{(M-Knapsack)} \qquad \min_{I \in \Upsilon} \left\{ \max_{J \in \Upsilon} \left(\sum_{h \in \mathcal{H}} u_h + (l_h - u_h) z_h) j_h \right) - \sum_{h \in \mathcal{H}} l_h z_h \right\} \tag{3.5}$$

M-Knapsack can be formulated as an ILP by replacing $F(J^{L^I}, L^I)$ with a free variable Ω and adding a new set of linear constraints that bounds the value of Ω to the value of $F(J^{L^I}, L^I)$. As is the case of M-ILP, the resulting formulation (3.6)–(3.10) for M-Knapsack has an exponentially large number of constraints (3.7).

$$\min_{I \in \Upsilon} \; \Omega - \sum_{h \in \mathcal{H}} l_h z_h \tag{3.6}$$

$$\Omega \geq \sum_{h \in \mathcal{H}} (u_h + (l_h - u_h) z_h)\, j_h, \quad \forall\, J \in \Upsilon \tag{3.7}$$

$$\sum_{h \in \mathcal{H}} w_h z_h \leq T \tag{3.8}$$

$$z_h \in \{0, 1\}, \quad \forall h \in \mathcal{H} \tag{3.9}$$

$$\Omega \geq 0 \tag{3.10}$$

Now, we are able to define the decision version of M-Knapsack. It is as follows.

DECISION M-KNAPSACK

Input: An item set $\mathcal{H}$, an weight $w_h \geq 0$ for each item $h \in \mathcal{H}$, a profit interval $[l_h, u_h]$ for each item $h \in \mathcal{H}$, and a capacity T.

Question: Is there a solution $I \in \Upsilon$ such that $Z(I) \leq \xi$ and $\sum_{h \in I} w_h \leq T$?

3.3 The hardness proof

Given a DECISION M-KNAPSACK instance, we obtain a DECISION M-ILP instance as follows. We set the vector of coefficients b with dimension 1, the matrix of coefficients A with dimension $1 \times |\mathcal{H}|$, and a set $\{ x_h \in \mathbb{B} : h \in \mathcal{H} \}$ of variables. Also, we set $b = \{ T \}$, $A_{1,h} = w_h : \forall h \in \mathcal{H}$, the lower value l_h to the profit upper value u_h, and the upper value

u_h to the profit lower value l_h. One may note that, in this construction, $\Phi = \Upsilon$ since all feasible solutions for M-ILP are feasible for M-Knapsack and vice-versa. Besides, in this construction, $\Gamma = \Lambda$.

Theorem 4 *The decision version of M-ILP is Σ_2^p-Complete.*

Proof. Consider a M-ILP instance as constructed above. We argue that M-ILP has a solution $X \in \Phi$ with $Z(X) \leq \zeta$ if and only if M-Knapsack has a solution $I \in \Upsilon$ with $Z(I) = -Z(X)$.

$\Rightarrow$: Suppose there exists a solution $X \in \Phi$ for the above-constructed instance of M-ILP such that $Z(X) \leq \zeta$. If this solution sets variable $x_h = 1$, then item $h \in \mathcal{H}$ is in the solution for M-Knapsack. On the other hand, if this solution sets variable $x_h = 0$, then item $h \in \mathcal{H}$ is not in the solution for M-Knapsack. As $A_{1,h} = w_h : \forall h \in \mathcal{H}$ and $b = \{T\}$, we guarantee that the capacity of the knapsack is not violated.

We are left to show that $-Z(X) = Z(I)$. If we rewrite M-ILP as a maximization problem, then $-Z(X) > -\zeta$. Furthermore, definitions 4–6 should be rewritten for the case of a maximization problem. Thus, we restate definitions 4, 5, and 6 as definitions 13, 14, and 15, respectively.

Definition 13 *The best possible solution $Y^S \in \Phi$ for the specific scenario $S \in \Gamma$ is*

$$Y^S = \arg\max_{Y \in \Phi} F(Y, S) = \arg\max_{Y \in \Phi} \sum_{h=1}^{n} c_h^S y_h,$$

i.e., Y^S is the optimal ILP solution for the scenario S.

Definition 14 *The regret of a solution $X \in \Phi$ in a specific scenario $S \in \Gamma$ is the difference between the cost of Y^S in S and the cost of X [Kouvelis and Yu, 1997], i.e., $F(Y^S, S) - F(X, S)$.*

Let $\mathcal{Z}(X) = -Z(X)$ be the robustness cost of solution X when M-ILP is formulated as a maximization problem.

Definition 15 *The robustness cost $\mathcal{Z}(X)$ of a solution $X \in \Phi$ is defined as the maximum regret of X among all scenarios in Γ, i.e.,*

$$\mathcal{Z}(X) = \max_{S \in \Gamma} \left\{ F(Y^S, S) - F(X, S) \right\}.$$

Despite the fact that $|\Gamma| = \infty$, it is proven in Aissi et al. [2009]; Averbakh [2001] that the scenario where the regret of $X \in \Phi$ is the maximum is the scenario $S^X \in \Gamma$, such that

$$c_h^{S^x} = u_h + (l_h - u_h)x_h,$$

i.e., $c_h^{S^X} = l_h$ if $x_h = 1$, and $c_h^{S^X} = u_h$ otherwise. From this result, we have that

$$F(X, S^X) = \sum_{i=1}^{n} l_h x_h,$$

$$F(Y^{S^X}, S^X) = \min_{Y \in \Phi} \sum_{h=1}^{n} \big(u_h + (l_h - u_h)x_h\big) y_h,$$

and

$$\mathcal{Z}(X) = F(Y^{S^X}, S^X) - F(X, S^X).$$

We refer to this scenario as the *worst-case scenario* induced by solution X.

The maximization version of M-ILP aims at finding the solution with minimum robustness cost, *i.e.*,

$$\min_{X \in \Phi} \mathcal{Z}(X) = \min_{X \in \Phi} \left\{ F(Y^{S^X}, S^X) - F(X, S^X) \right\}.$$

This problem can be rewritten as the 0-1 Bilevel Integer Linear Program in Equation (3.11) by replacing $F(Y^{S^X}, S^X)$ and $F(X, S^X)$.

$$\text{(Maximization M-ILP)} \qquad \min_{X \in \Phi} \left\{ \max_{Y \in \Phi} \sum_{h=1}^{n} \big(l_h + (u_h - l_h)x_h\big) y_h - \sum_{h=1}^{n} u_h x_h \right\} \tag{3.11}$$

One may note that the objective function (3.11) is the same of the M-Knapsack, as defined in Equation (3.5). This is due to the fact that $\Phi = \Upsilon$, the lower value l_h is equal to the profit upper value u_h, and the upper value u_h is equal to the profit lower value l_h, and $x_h = 1 : \forall h \in I$. Therefore, we have that $\mathcal{Z}(X) = Z(I) = -Z(X)$.

$\Leftarrow$: Suppose there exists a solution $I \in \Upsilon$ for M-Knapsack such that $Z(I) \leq \xi$. By definition, this solution does not violate the capacity T of the knapsack, *i.e.*, $\sum_{h \in I} w_h \leq T$. As $A_{1,h} = w_h : \forall h \in \mathcal{H}$ and $b = \{T\}$, we guarantee that constraint (1.7) of M-ILP is not violated.

We are left to show that $Z(X) = Z(I)$, which can be demonstrated using the same argument employed above for the proof of the *if* case. $\square$

Chapter 4

Algorithms for M-ILP

This chapter presents the literature algorithms for minmax regret optimization problems under interval uncertainty and instantiates them for M-ILP. Furthermore, it also presents the two FAO heuristics proposed in this thesis. First, in Section 4.1, we present three exact algorithms for M-ILP, namely the Benders-like Decomposition (BLD), the Extended Benders (EB), and the Branch-and-Cut (BC). Next, in Section 4.2, we present the heuristics Algorithm Mean Upper (AMU), Scenario-based Algorithm (SBA), Linear Programming Based Heuristic (LPH), and the two FAO heuristics.

4.1 Exact algorithms for M-ILP

The M-ILP formulation given by objective function (1.5) and constraints in (1.6)–(1.9) have an exponential number of the constraints (1.6). Enumerating all these constraints is a #P-Complete problem [Cook et al., 1992], since there is one constraint for each feasible solution of the problem. Therefore, this formulation cannot be explicitly solved by commercial MIP solvers, and more sophisticated methods are needed. We present below the three exact algorithms which are widely applied to solve particular M-ILP instances.

4.1.1 Benders-like Decomposition (BLD)

BLD relies on a cutting plane algorithm inspired by the Benders' Decomposition [Benders, 1962]. It differs from the classic algorithm because the Bender's subproblem is not a linear program. BLD was successfully applied to solve the Minmax regret Traveling Salesman Problem under Interval Uncertainty [Montemanni et al., 2007], the Minmax regret Knapsack problem under Interval Uncertainty [Furini et al., 2015], the Minmax regret Restricted Shortest Path problem under Interval Uncertainty [Assunção et al., 2017], and the Minmax regret Set Covering problem under Interval Uncertainty [Pereira and Averbakh, 2013], among other problems.

BLD decomposes M-ILP into two problems: (*i*) the Master problem (MP); and (*ii*) the Subproblem (SP). The MP consists in solving the formulation defined by objective function (1.5) and the constraints in (1.7)–(1.9),(4.1), *i.e.*, it solves a M-ILP using only a subset of the constraints in (1.6). The SP consists in solving ILP where the cost coefficients of the objective function are given by the worst-case scenario induced by the optimal solution of the MP. BLD solves the MP and the SP iteratively and stops when the solution given by the MP has a robustness cost equal to that of the smallest robustness cost of a solution computed by the SP. Assunção et al. [2016] proved that this algorithm always returns the optimal solution for M-ILP.

$$\theta \leq \sum_{i=1}^{n} \left(l_i + (u_i - l_i)x_i\right) y_i, \quad \forall\, Y \in \Phi^h \tag{4.1}$$

Let $\Phi^h \subseteq \Phi$ be the subset of solutions generated by BLD until iteration h. Algorithm 1 gives a pseudo-code of BLD. It receives as input the matrix of coefficients A, the vector of variables x, and the vector of coefficients b of ILP. Additionally, it receives the interval uncertainties $[l_i, u_i]$ for all uncertain coefficients. Initially, as suggested by Montemanni et al. [2007] to avoid an unbounded MP, BLD solves M-ILP using the Algorithm Mean (AM) and Algorithm Upper (AU) algorithms (which will be described in Section 4.2.1) on lines 1 and 2, respectively. Then, it initializes Φ^0 with solutions obtained by these heuristics in line 3. In lines 4 and 5, it keeps respectively the smallest robustness cost between solutions computed by AM and AU and the solution which gives the smallest robustness cost. Furthermore, it initializes the the iteration count in line 6. The loop on lines 7–14 corresponds to an iteration of BLD and is run until an optimal solution is found. First, the MP is solved in line 8 using constraints (4.1). Then, the SP corresponding to the solution obtained by previous iteration of the MP is solved in line 9. Next, the primal bound ϑ of BLD is computed as the minimum between the previous value of ϑ and the cost of the solution of the SP on line 10 and the solution with the smallest robustness cost is kept on line 11. At the end of the loop, the iteration number is increased in line 12 and Φ^h is augmented by incorporating the solution found by the SP in line 13. Finally, the best feasible solution X^* computed is returned in line 15.

Algorithm 1: Benders-like Decomposition (BLD)

```
   input  : $\langle A, x, b, [l_i, u_i] \forall c_i \in c \rangle$
   output: $X^* \in \Phi$
1  $X^= \leftarrow AM(A, x, b, [l_i, u_i] \forall c_i \in c)$
2  $X^+ \leftarrow AU(A, x, b, [l_i, u_i] \forall c_i \in c)$
3  $\Phi^0 \leftarrow \{X^=, X^+\}$
4  $X^* \leftarrow \arg\min_{X \in \{X^= X^+\}} Z(X)$
5  $\vartheta \leftarrow \min_{X \in \{X^=, X^+\}} Z(X)$
6  $h \leftarrow 0$
7  do
8      $X \leftarrow MP(A, x, b, \Phi^h, [l_i, u_i] \forall c_i \in c)$
9      $Y \leftarrow SP(A, x, b, S^X)$
10     $\vartheta \leftarrow \min\left(\vartheta, F\left(Y^{(S^X)}, S^X\right)\right)$
11     $X^* \leftarrow \arg\min_{X \in \{X^*, Y\}} Z(X)$
12     $h \leftarrow h + 1$
13     $\Phi^h \leftarrow \Phi^{h-1} \cup \{Y\}$
14 while $(Z(X) < \vartheta)$
15 return $X^*$
```

4.1.2 Extended Benders-like Decomposition (EB)

EB employs a methodology introduced by Fischetti et al. [2010] for selecting Benders' cuts. The proposed methodology consists in solving a SP for each incumbent solution found by the MP instead of using only the MP's optimal solution. Thus, the expected number of iterations in EB is smaller than of BLD. Pereira and Averbakh [2013] first implemented EB for the M-WSCP and demonstrated that its outperformed BLD running time. However, Coco [2017] found that BLD outperformed EB when solving the Minmax regret Maximum Covering Location problem under Interval Uncertainty.

A pseudo-code for EB is very similar with that of BLD, which was described in Algorithm 1. The main difference is that, instead of solving an unique SP on line 10, it solves a SP for each integer solution found by the MP solved in line 8.

4.1.3 Branch-and-Cut (BC)

BC was initially proposed by Montemanni et al. [2007] for solving the Minmax regret Traveling Salesman Problem under Interval Uncertainty. The authors noticed that BLD may be computationally inefficient, as each iteration of this algorithm runs an branch-and-bound algorithm from scratch to solve the Bender's Master Problem. In BC, each incumbent solution found by the algorithm is processed by a Subproblem (which is the same of the BLD and EB) to generate a new cut on the original branch-and-bound algorithm. BC differs from EB because the computed cuts are inserted globally, being propagated to all active nodes in the branch-and-bound tree, while the Master Problem of BLD and EB is reinitialized after each iteration of it's Subproblem. These cuts are inserted using a branch-and-cut framework. BC was shown to outperforms BLD when solving the Minmax regret Travelling Salesman Problem under Interval Uncertainty [Montemanni et al., 2007] and the Minmax regret Knapsack Problem under Interval Uncertainty [Furini et al., 2015]. However, Coco [2017] experiments demonstrated that both BLD and EB outperformed BC when solving the Minmax regret Set Covering Problem under Interval Uncertainty and the Minmax regret Maximal Coverage Location Problem under Interval Uncertainty.

Algorithm 2 gives a pseudo-code for BC. It receives as input the matrix of coefficients A, the vector of variables x, and the vector of coefficients b of ILP. Additionally, it receives the interval uncertainties $[l_i, u_i]$ for all uncertain coefficients. Initially, as suggested by Montemanni et al. [2007] to avoid an unbounded Master problem, BCsolves M-ILP using the AM and AU algorithms (which will be described in Section 4.2.1) on lines 1 and 2, respectively. Then, it initializes $\Phi' \subseteq \Phi$ with solutions obtained by these heuristics in line 3. Next, the Branch-and-Cut framework is performed on line 4, which stops when an optimal solution is found or when a predefined time limit is met. Finally, the best feasible solution X^* computed is returned in line 5.

Algorithm 2: Branch-and-Cut (BC)

```
  input  : $\langle A, x, b, [l_i, u_i] \forall c_i \in c \rangle$
  output: $X^* \in \Phi$
1 $X^= \leftarrow AM(A, x, b, [l_i, u_i] \forall c_i \in c)$
2 $X^+ \leftarrow AU(A, x, b, [l_i, u_i] \forall c_i \in c)$
3 $\Phi' \leftarrow \{X^=, X^+\}$
4 $X^* \leftarrow$ Branch-and-Cut$(A, x, b, \Phi', [l_i, u_i] \forall c_i \in c)$
5 return $X$
```

4.2 Heuristics for M-ILP

Heuristics for minmax regret optimization problems under interval uncertainty generally rely on solving an ILP into different pre-determined scenarios and returning the best computed solution or uses the dual counterpart of the linear relaxation of M-ILP. This section extends these heuristics from the literature to M-ILP. In addition, this section also introduces the FAO heuristics, which is a new heuristic strategy for solving M-ILP.

4.2.1 Algorithm Mean Upper (AMU)

AMU [Kasperski and Zieliński, 2006] is a 2-approximative heuristic framework for M-ILP. It is the most widely applied heuristic to solve specific instances of this problem (*e.g.*, Pereira and Averbakh [2013]; Furini et al. [2015], and Carvalho et al. [2016]). The worst-case complexity of the algorithms developed through this framework is the same as solving the classical counterpart of the robust optimization problem at hand. AMU relies on two other heuristics, AM and AU. Thus, we first describe AM and AU and, then, introduce AMU.

AM solves the deterministic counterpart of M-ILP at the midpoint scenario s^m. In s^m, the cost of each uncertain coefficient is set to its mean value, *i.e.* $c_i^{s^m} = \frac{l_i+u_i}{2}$, for all uncertain coefficients $c_i \in c$. Next, the maximum regret of the computed solution is evaluated and returned. The cost of the solution obtained through AM is bounded by a factor of 2 from the optimal solution, as proved by Kasperski and Zieliński [2006].

AU is similar to AM. However, instead of solving M-ILP for scenario s^m, AU solves for the upper scenario s^u, where the cost of each uncertain coefficient is set to its upper value, *i.e.* $c_i^{s^u} = u_i$, for all uncertain coefficients $c_i \in c$. Unlike AM, the cost of the solution obtained by AU is not bounded by any factor.

AMU combines both AM and AU, returning the solution with the smallest robust cost. As AMU uses the solution from AM, it is also a 2-approximation algorithm for any minmax optimization problem under interval uncertainty. Figure 4.1 shows an example of a scenarios s^m and s^u solved by AMU considering the digraph under interval uncertainty shown in Figure 1.1(a).

4.2.2 Scenario-Based Algorithm (SBA)

SBA was first proposed for the M-WSCP [Coco et al., 2015, 2016]. SBA extends AMU, where target scenarios between the lower scenario $s^l \in \Gamma$ (a scenario where the cost of the arcs are set to their respective lower, *i.e.* $c_{ij}^{s^l} = l_{ij}$) and the upper scenario (s^u) are investigated. SBA inspects a set $Q = \{q_1, q_2, \dots, q_p\} \subseteq \Gamma$ of target scenarios. These scenarios are computed using three parameters:(i) the initial scenario α; (ii) the final scenario β; and (iii) the step size γ. All parameters are real-valued in the interval $[0, 1]$. The cost of the uncertain coefficients for each target scenario $q_j \in Q$ is computed as $c_i^{q_j} = l_i + (\alpha + \delta \cdot \gamma) \cdot (u_i - l_i)$, $\delta \in \mathbb{N}$, for all values of δ such that $\alpha + \delta\gamma \leq \beta$.

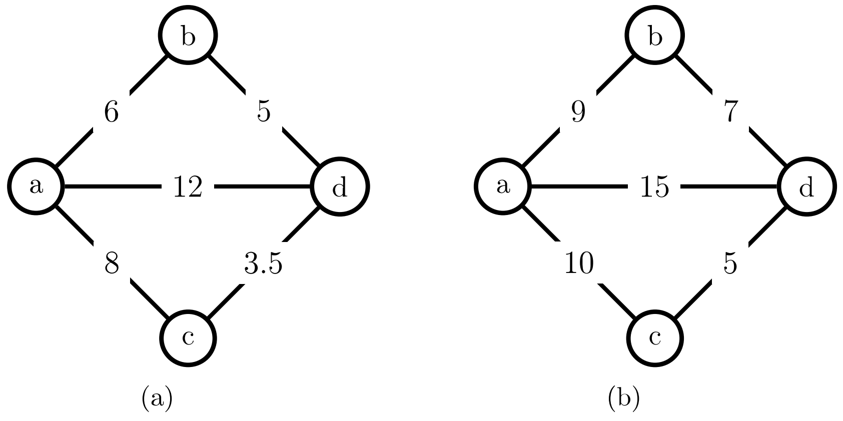

Figure 4.1. (a) Scenario considered by AM and (b) scenario considered by AU

Algorithm 3 presents a pseudocode of SBA. It receives as input the matrix of coefficients A, the vector of variables x, and the vector of coefficients b of ILP. Additionally, it receives the interval uncertainties $[l_i, u_i]$, for all uncertain coefficients, and parameters α, β, and γ. Initially, γ is set to α in line 1 and the value of *primal* is set to infinity in line 2. The algorithm consists in the main loop from lines 3–8. It solves the deterministic counterpart of M-ILP on scenario q_j on line 4. Then, if the regret of the computed solution is smaller than that of the previous best solution, it updates the value of the best solution found in line 6 and stores the solution found in line 7. The value of γ is incremented in line 8. This loop runs until γ is greater than β. The best solution found X is then returned in line 9.

Algorithm 3: Scenario-based Algorithm (SBA)

```
   input  : $\langle A, x, b, \alpha, \beta, \gamma, [l_i, u_i] \forall c_i \in c \rangle$
   output: $X^* \in \Phi$
1  $\delta \leftarrow \alpha$
2  $\vartheta \leftarrow \infty$
3  while $\delta \leq \beta$ do
4      $X \leftarrow$ M-ILP$(A, x, b, [l_i, u_i] \forall c_i \in c, \delta)$        // Solve M-ILP with $c_i^{q_j}, \forall c_i \in c$
5      if $\left(\sum_{i=0}^{n} c_i^{q_j} y_i < \vartheta\right)$ then
6          $\vartheta \leftarrow \sum_{i=0}^{n} c_i^{q_j} y_i$
7          $X^* \leftarrow Y$
8      $\delta \leftarrow \delta + \gamma$
9  return $X^*$
```

4.2.3 Linear Programming Heuristic (LPH)

LPH was introduced by Assunção et al. [2017]. It proposes a formulation whose solutions are guaranteed to be feasible for the original minmax regret optimization problem under interval uncertainty and that the optimal value of the objective function approximates the value of that of the original problem. This heuristic was applied to solve the Minmax regret Restricted Shortest Path Problem under Interval Uncertainty and the Minmax regret Set Covering Problem under Interval Uncertainty. Computational experiments indicated that LPH outperformed AMU when solving both problems.

For the case of M-ILP, the formulation solved by LPH is constructed as follows. Equations (4.2)–(4.4) represent linear relaxation of the problem associated with second term of the 0-1 Bilevel Integer Linear Program defined in Equation (1.4), *i.e.*, $F(Y^{S^X}, S^X)$.

$$\min \sum_{i=0}^{n} \big(l_i + (u_i - l_i)x_i\big)y_i \tag{4.2}$$

$$\sum_{i=0}^{n} a_{ji}x_i \leq b_j, \quad \forall j \in \{0, 1, \ldots, m\} \tag{4.3}$$

$$y_i \in [0, 1], \quad \forall i \in \{0, 1, \ldots, n\} \tag{4.4}$$

One can denote the dual of the formulation given by Equations (4.2)–(4.4) by the objective function (4.5) and the constraints in (4.6)–(4.7).

$$\max \sum_{j=0}^{m} b_j \nu_j \tag{4.5}$$

$$\sum_{j=0}^{m} a_{ji}\nu_j \geq l_i + (u_i - l_i)x_i \tag{4.6}$$

$$\nu_j \geq 0, \quad \forall j \in \{0, 1, \ldots, m\} \tag{4.7}$$

Thus, the formulation solved by LPH consists in replacing $F(Y^{S^X}, S^X)$ in the problem defined in Equation (1.4) by its associated dual defined above. The resulting formulation is given by objective function (4.8) and the constraints in (1.6)–(1.8), (4.6)–(4.7). This formulation is compact, as the number of constraints in (4.6) grows polynomially with $|m|$. LPH consists in solving this formulation and returning the best solution found.

$$\min \sum_{i=0}^{n} u_i x_i - \sum_{j=0}^{m} b_j \nu_j \tag{4.8}$$

4.2.4 Fix-and-Optimize through Scenario-based Algorithm (FO-SBA)

The Fix-and-Optimize through Scenario-based Algorithm (FO-SBA) relies on the SBA heuristic described in Section 4.2.2. The intuition behind this heuristic is that the variables which have a zero value in the optimal solution of all M-ILP scenarios solved by SBA are less likely to appear in the in the optimal solution than the other. Therefore, these variables are removed from the problem and M-ILP is solved on the reduced instance.

FO-SBA constructs a Restricted Minmax regret 0-1 Integer Linear Programming Problem under Interval Uncertainty (RM-ILP) by setting some of the variables of M-ILP to zero. Let V be the set of variables such that $\{ i = 0 : x_i \in X \}$ for all feasible solutions X computed though SBA. RM-ILP consists in solving the formulation denoted by objective function (1.5) and the constraints in (1.6)–(1.9) and (4.9).

$$x_i = 0, \quad \forall i \in V \tag{4.9}$$

Algorithm 4 shows the pseudo-code of FO-SBA. It receives as input the matrix of coefficients A, the vector of variables x, and the vector of coefficients b of ILP. Additionally, it receives the interval uncertainties $[l_i, u_i]$, for all uncertain coefficients, and parameters α, β, and γ of SBA. Initially, γ is set to α in line 1 and V is initialized in line 2. The preprocessing step consists of the loop in lines 3–6. At each iteration of this loop, it solves the deterministic counterpart of M-ILP on scenario q_j on line 4. Then, it stores the variables which were in the optimal solution of the previously solved problem on line 5 and increases the value of γ in line 6. The solving step is performed in line 7, which consists in solving the resulting RM-ILP formulation. One may observe that this formulation can be solved with any exact algorithm for M-ILP. Finally, FO-SBA returns the computed solution X^* on line 8.

4.2.5 Fix-and-Optimize through Linear Relaxation (FO-LR)

The Fix-and-Optimize through Linear Relaxation (FO-LR) relies on the linear relaxation of the M-ILP formulation defined by Equations (1.5)–(1.9), *i.e.*, it solves the

Algorithm 4: FO-SBA

```
  input  : $\langle A, x, b, \alpha, \beta, \gamma, [l_i, u_i] \forall c_i \in c \rangle$
  output: $X^* \in \Phi$
1 $\delta \leftarrow \alpha$
2 $V \leftarrow X$
3 while $\delta \leq \beta$ do
4     $X \leftarrow$ M-ILP$(A, x, b, [l_i, u_i] \forall c_i \in c, \delta)$        // Solve M-ILP with $c_i^{q_j}, \forall c_i \in c$
5     $V \leftarrow V \setminus \{i : x_i \in X, x_i = 1\}$
6     $\delta \leftarrow \delta + \gamma$
7 $X^* \leftarrow$ RM-ILP$(A, x, b, [l_i, u_i] \forall c_i \in c, V)$
8 return $X^*$
```

Linear Program (LP) corresponding to relaxing the integrality of constraints in (1.9). The intuition behind this heuristic is that the variables which have a zero value in the optimal solution of the LP relaxation are less likely to appear in the optimal solution of M-ILP than the others. Therefore, the variables which are equal to zero in the LP are removed and M-ILP is solved on the reduced instance.

Let LM-ILP denotes the linear relaxation of the M-ILP formulation defined by Equations (1.5)–(1.9). Similarly to FO-SBA, FO-LR builds a RM-ILP by setting some of the variables of M-ILP to zero. Let V be the set of variables such that $\{\, i = 0 : x_i \in X \,\}$ for the optimal solution X of LM-ILP. RM-ILP consists in solving the formulation denoted by objective function (1.5) and the constraints in (1.6)–(1.9) and (4.9).

Algorithm 5 shows the pseudo-code of FO-LR, which receives as input the matrix of coefficients A, the vector of variables x, and the vector of coefficients b of ILP. Additionally, it receives the interval uncertainties $[l_i, u_i]$ for all uncertain coefficients. The preprocessing step consists in lines 1 and 2. In line 1, it solves RM-ILP, which can be done using any exact algorithm for M-ILP. In line 2, it stores in V the variables which were equal to zero in the optimal solution of LM-ILP. The solving step is performed in line 3, and consists in solving the RM-ILP formulation. Finally, the resulting solution X^* is returned on line 4.

Algorithm 5: FO-LR

```
  input  : $\langle A, x, b, [l_i, u_i] \forall c_i \in c \rangle$
  output: $X^* \in \Phi$
1 $X \leftarrow$ LM-ILP$(A, x, b, [l_i, u_i] \forall c_i \in c)$
2 $V \leftarrow \{y_i : y_i \in Y, y_i > 0\}$
3 $X^* \leftarrow$ RM-ILP$(A, x, b, [l_i, u_i] \forall c_i \in c, V)$
4 return $X^*$
```

Chapter 5

Case studies

In this chapter we present two problems which will be used as case studies for solving M-ILP. The first is the M-WSCP [Pereira and Averbakh, 2013], which is a Minmax regret variant of the classic Weighted Set Covering Problem (WSCP) [Edmonds, 1962]. The second is the MSS-SPP [Catanzaro et al., 2011], which is a Minmax regret variant of the well-known Single-Source Shortest Path Problem (SS-SPP) [Cormen et al., 2009]. Both problems can be formulated as M-ILP.

5.1 The Weighted Set-Covering problem

The WSCP [Edmonds, 1962], also known as the Weighted Set Cover Problem, is one of the most studied combinatorial optimization problems. Its decision version was one of the original 21 problems proved to be NP-Complete by Karp [1972]. WSCP applications range from data partitioning [Li et al., 2017] to vehicle routing problems [Bai et al., 2015]. For the reader interested in a deeper knowledge on this problem, we indicate the annotated bibliography of Ceria et al. [1997].

WSCP is defined by a set of objects $\mathcal{N}$ and a set $\mathcal{S}$ of subsets of $\mathcal{N}$, such that each subset $s \in \mathcal{S}$ contains one or more elements of $\mathcal{N}$. Furthermore, we associate an weight w_s to each subset $s \in \mathcal{S}$. The objective of WSCP is to find $\mathcal{S}^* \subseteq \mathcal{S}$ such that all objects are covered by at least one subset $s \in \mathcal{S}^*$ and that the sum of the weights of the sets in S^* is minimum.

Figure 5.1 shows a WSCP instance with $|\mathcal{N}| = 16$ and $|\mathcal{S}| = 5$. The optimal solution $\mathcal{S}^*$ for this instance is $\mathcal{S}^* = s_1, s_3, s_5$, such that $\sum_{s \in \mathcal{S}^*} w_s = 13$. One can note that subsets s_2 and s_4 are not needed, since the other subsets cover all of their elements.

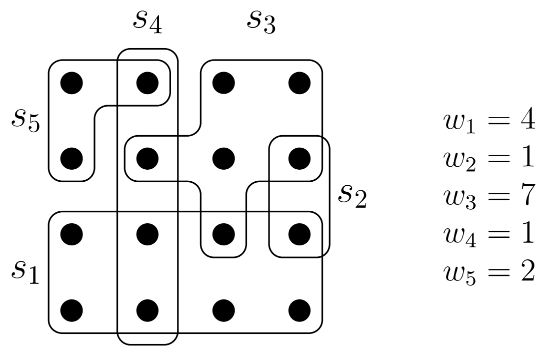

Figure 5.1. An WSCP instance

WSCP can be formulated as following. It uses decision variables $\{ x_s \in \mathbb{B} : s \in \mathcal{S} \}$ such that $x_s = 1$ if subset $s \in \mathcal{S}$ belongs to the optimal WSCP solution and $x_s = 0$ otherwise. Furthermore, let $c_{is} = 1$ if item $i \in \mathcal{N}$ is covered by subset s and $c_{is} = 0$ otherwise. The WSCP formulation consists of the objective function (5.1) and constraints in (5.2)–(5.3). The objective function (5.1) aims at minimizing the weight of subsets $s \in \mathcal{S}$. Inequalities in (5.2) ensure that all items $i \in \mathcal{N}$ are covered by at least one subset $s \in \mathcal{S}$. The domain of variables x_s are defined by Constraints in (5.2).

$$\min \sum_{s \in \mathcal{S}} w_s x_s \tag{5.1}$$

$$s.t. \quad \sum_{s \in \mathcal{S}} c_{is} x_s \geq 1, \qquad \forall i \in \mathcal{N} \tag{5.2}$$

$$x_s \in \{0, 1\}, \qquad \forall s \in \mathcal{S} \tag{5.3}$$

5.1.1 The Minmax regret Weighted Set Covering problem under Interval Uncertainties

M-WSCP, as defined by Pereira and Averbakh [2013], is a RO variant of WSCP. In this problem, the weight associated to each subset $s \in \mathcal{S}$ is uncertain, but it is assumed to be bounded by an interval $[l_s, u_s]$, with $0 \leq l_s \leq u_s$. The objective of M-WSCP is to find a solution X^* with the smallest robustness cost, according to the definitions presented in Chapter 1.

Pereira and Averbakh [2013] presented a ILP formulation for M-WSCP. It was obtained by linearizing $F(Y^{(S^X)}, S^X)$, as explained by Aissi et al. [2009]. It makes use of binary decision variables $\{ x_s \in \mathbb{B} : s \in \mathcal{S} \}$ and parameter $\{ c_{is} : i \in \mathcal{N}, s \in \mathcal{S} \}$, as defined for the WSCP. Besides, it also uses an auxiliary variables $\theta \in \mathbb{R}_{\geq 0}$ to compute the cost of the solution in the worst-case scenario, as in the formulation defined by Equations (1.5)–(1.9) for M-ILP. The resulting formulation is defined by the objective function (5.4) and the constraints in (5.5)–(5.8).

$$\min \sum_{s \in \mathcal{S}} (u_s x_s) - \theta \tag{5.4}$$

$$s.t. \sum_{s \in \mathcal{S}} c_{is} x_s \geq 1, \qquad \forall i \in \mathcal{N} \tag{5.5}$$

$$\theta \leq \sum_{s \in \mathcal{S}} (l_s + (u_s - l_s) x_s)\, y_s, \qquad \forall\, Y \in \Phi \tag{5.6}$$

$$x_s \in \{0, 1\}, \qquad \forall s \in \mathcal{S} \tag{5.7}$$

$$\theta \geq 0 \tag{5.8}$$

The objective function (5.4) aims at minimizing the maximum regret. The constraints in (5.5) ensure that all items $i \in \mathcal{N}$ are covered by at least one subset $s \in \mathcal{S}$. The inequalities in (5.6) enforce the correct value of θ. The constraints in (5.7)–(5.8) define the domain of the variables x_s and θ, respectively.

5.1.2 Worst-case complexity for the Minmax regret Weighted Set Covering problem under Interval Uncertainty

In this section, we prove that the decision version of M-WSCP lies on the second level of the polynomial hierarchy, being Σ_2^p [Stockmeyer, 1976]. This proof employs a similar argument as used in Chapter 3 for the decision version of M-ILP. However, the question of whether M-WSCP is complete for this class still remains as an open problem.

Decision M-WSCP
Input: A set of objects $\mathcal{N}$ and a set $\mathcal{S}$ of subsets of $\mathcal{N}$, along with an interval $[l_s, u_s]$ of weights associated to each subset $s \in \mathcal{S}$.
Question: Is there a solution $X \in \Phi$ forM-WSCP such that $Z(X) \leq \rho$?

Lemma 5 *Problem* Decision M-WSCP *lies in class* Σ_2^p.

Proof. From Definition 6, we recall that $Z(X) = \max\limits_{S \in \Gamma} \{F(X, S) - F(Y^S, S)\}$. We can rewrite this equation as

$$Z(X) = \left\{ \sum_{i \in \mathcal{S}} u_i x_i - \min_{Y \in \Phi} \sum_{i \in \mathcal{S}} (l_i + (u_i - l_i) x_i)\, y_i \right\}.$$

Thus, the question for DECISION M-WSCP is: exists a solution $X \in \Phi$ for M-WSCP such that

$$\left\{ \sum_{i \in \mathcal{S}} u_i x_i - \min_{Y \in \Phi} \sum_{i \in \mathcal{S}} \left(l_i + (u_i - l_i) x_i \right) y_i \right\} \leq \rho ?$$

One may observe that this question is on the form $\exists x \forall y P(x, y)$, where the Boolean predicate $P(x, y)$ can be evaluated in polynomial time. Theorem 17.8 from Papadimitriou [1994] states that problems with an existential quantifier followed by an universal quantifier (which is the exact form of the question stated above) are in the complexity class Σ_2^p, which completes the proof. □

If P $\neq$ NP, we can assume that the polynomial hierarchy do not collapses to the second level. Therefore, we can state the following corollary.

Corollary 6 *Problem* DECISION M-WSCP *is not contained in class NP.*

Furthermore, by using the Kannan's Theorem [Kannan, 1982], we can state the following corollary.

Corollary 7 *Do not exists a polynomial-sized formulation for M-WSCP.*

The latter corollary can be easily verifiable. If there is a polynomial-sized formulation for M-WSCP, then there exists an algorithm which verifies the cost of its solution in polynomial-time and, thus, it is in NP. This is clearly a contradiction, since problems in the second level or higher of the polynomial-hierarchy are not contained in NP [Papadimitriou, 1994].

5.2 The Single-Source Shortest Path problem

The SS-SPP [Cormen et al., 2009], also known as the Shortest Path Tree Problem, is a shortest path problem where one is interested in computing the shortest path from a predefined node r to all other nodes of a graph. The solution of SS-SPP is a tree, which is known to be the shortest path tree of the graph. SS-SPP find applications in multicast routing in computer networks [Zhang and Mouftah, 2002; Shelby and Bormann, 2011], which usually has a better performance than other routing topologies [Nguyen and Xu, 2007]. One can solve SS-SPP using classic polynomial-time algorithms, such as Dijkstra's and Bellman-Ford's [Cormen et al., 2009].

SS-SPP is defined on a connected digraph $G = (N, A)$ with a set N of nodes and a set A of arcs, where each arc $(i, j) \in A$ is associated with a cost coefficient $c_{ij} \geq 0$. Furthermore, we are also a node $r \in N$, which is the root of the arborescence. The objective of SS-SPP is to find $A^* \subseteq A$ such that A^* is an arborescence and contains the shortest paths from r to all other nodes in $N \setminus \{r\}$.

Figure 5.2 shows a SS-SPP instance with $|N| = 4$ and $|A| = 5$. The optimal solution A^* for this instance is $A^* = \{\,(r,a),(r,b),(a,c)\,\}$, such that the cost of the path from r to a is 8, the cost of the path from r to b is 10, and the cost of the path from r to c is 11. One can note that A^* is also an spanning arborescence of G, as $|A^*| = |N| - 1$ and it spans all nodes of G.

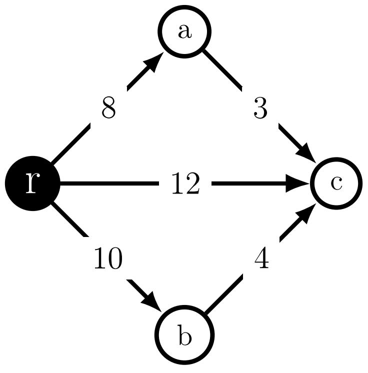

Figure 5.2. A SS-SPP instance

SS-SPP can be formulated as following. It uses decision variables $\{\, x_{ij} \in \mathbb{B} : (i,j) \in A \,\}$ and $\{\, x_{ij}^k \in \mathbb{B} : (i,j) \in A, k \in N \setminus \{r\} \,\}$, such that $x_{ij} = 1$ if the arc $(i,j) \in A^*$ and $x_{ij} = 0$ otherwise. Additionally, $x_{ij}^k = 1$ if the arc $(i,j) \in A^*$ is in the path from r to k and $x_{ij}^k = 0$ otherwise. The SS-SPP formulation consists of the objective function (5.9) and the constraints in (5.10)–(5.14).

$$\min \sum_{k \in N \setminus \{r\}} \left[\sum_{(i,j) \in A} c_{ij} x_{ij}^k \right] \tag{5.9}$$

$$s.t. \sum_{(j,i) \in A} x_{ji}^k - \sum_{(i,j) \in A} x_{ij}^k = \begin{cases} 1, & \text{if } j = r \\ -1, & \text{if } j = k \\ 0, & \text{otherwise} \end{cases}, \qquad \forall j \in N,\ k \in N \setminus \{r\} \tag{5.10}$$

$$x_{ij}^k \leq x_{ij}, \qquad \forall (i,j) \in A, k \in N \setminus \{r\} \tag{5.11}$$

$$\sum_{(i,j) \in A} x_{ij} = |N| - 1 \tag{5.12}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{5.13}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in A, k \in N \setminus \{r\} \tag{5.14}$$

The objective function (5.9) aims at minimizing the maximum regret. The constraints in (5.10) are the classic flow conservation constraints which ensure the connectivity of the paths from r to every other node in N. The inequalities in (5.11) enforce the path from r to $k \in N \setminus \{r\}$ to be set on the arborescence induced by variables x_{ij}, for all $(i, j) \in A$. The constraint in (5.12), together with the constraints in (5.10) and (5.11), guarantees that variables x_{ij} induce an arborescence. The constraints in (5.13)–(5.14) define the domain of the variables x_{ij} and x_{ij}^k, respectively.

5.2.1 The Minmax regret Single-Source Shortest Path Problem under Interval Uncertainty

MSS-SPP, as defined by Catanzaro et al. [2011], is a RO variant of SS-SPP. In this problem, the cost of each arc $(i, j) \in A$ is uncertain, but it is assumed to be bounded by an interval $[l_{ij}, u_{ij}]$, with $0 \leq l_{ij} \leq u_{ij}$. The objective of MSS-SPP is to find a solution X^* with the smallest robustness cost, according to the definitions presented in Chapter 1.

Neumann [2010] presented a ILP formulation for MSS-SPP. It was obtained by linearizing $F(Y^{(S^X)}, S^X)$, as explained by Aissi et al. [2009]. It makes use of binary decision variables $\{\, x_{ij} \in \mathbb{B} : (i, j) \in A \,\}$ and $\{\, x_{ij}^k \in \mathbb{B} : (i, j) \in A, k \in N \setminus \{r\} \,\}$, as defined for the SS-SPP. Besides, it also uses auxiliary variables $\{\, z_k \in \mathbb{R}_{\geq 0} : k \in N \setminus \{r\} \,\}$ to keep the cost of the shortest path from r to k in the worst-case scenario defined by variables x_{ij}. The resulting formulation is defined by the objective function (5.15) and the constraints in (5.16)–(5.23).

$$\min \sum_{k\in N\setminus\{r\}} \left[\sum_{(i,j)\in A} u_{ij}x_{ij}^k - z_k \right] \tag{5.15}$$

$s.t.$

$$\sum_{(j,i)\in A} x_{ji}^k - \sum_{(i,j)\in A} x_{ij}^k = \begin{cases} 1, & \text{if } j = r \\ -1, & \text{if } j = k \\ 0, & \text{otherwise} \end{cases}, \qquad \forall j \in N,\ k \in N\setminus\{r\} \tag{5.16}$$

$$x_{ij}^k \le x_{ij}, \qquad \forall (i,j) \in A, k \in N\setminus\{r\} \tag{5.17}$$

$$\sum_{(i,j)\in A} x_{ij} = |N| - 1 \tag{5.18}$$

$$z_j \le z_i + l_{ij} + (u_{ij} - l_{ij})x_{ij}, \qquad \forall (i,j) \in A \tag{5.19}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{5.20}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in A, k \in N\setminus\{r\} \tag{5.21}$$

$$z_r = 0, \tag{5.22}$$

$$z_k \ge 0, \qquad \forall k \in N\setminus\{r\} \tag{5.23}$$

The objective function (5.15) aims at minimizing the maximum regret. The constraints in (5.16) are the classic flow conservation constraints which ensure the connectivity of the paths from r to every other node in N. The inequalities in (5.17) enforce the path from r to $k \in N\setminus\{r\}$ to be set on the arborescence induced by variables x_{ij}, for all $(i,j) \in A$. The constraint in (5.18), together with the constraints in (5.16) and (5.17), guarantees that variables x_{ij} induce an arborescence. The inequalities in (5.19) enforce the correct value of z_k. The constraints in (5.20)–(5.23) define the domain of the variables x_{ij}, x_{ij}^k, z_r, and z_k, respectively.

5.2.2 Worst-case complexity for the Minmax regret Single-Source Shortest Path Problem under Interval Uncertainty

In this section, we prove MSS-SPP is NP-Hard. We obtain this result by demonstrating that the decision version of MSS-SPP is NP-Complete through a reduction from the 2-Partition Problem (2PP) [Karp, 1972], employing a similar technique as used by Kasperski and ZielińSki [2006]. First, we define the decision versions of MSS-SPP and 2PP. Then, we demonstrate how to reduce a 2PP instance to an MSS-SPP instance and show that a solution for MSS-SPP is valid if and only if it is also valid for 2PP. We show that this proof holds even on layered digraphs with only 3 layers.

DECISION MSS-SPP
Input: A connected digraph $G = (N, A)$, an interval $[l_{ij}, u_{ij}]$ for each arc $(i, j) \in A$ with $0 \leq l_{ij} \leq u_{ij}$, a source node $r \in N$, and an integer $e \geq 0$.
Question: Is there an arborescence X such that $Z(X) = e$?

2-PARTITION PROBLEM
Input: A finite set I and a weight $a_i \geq 0, \forall i \in I$.
Question: Is there a partition of I into two disjoint subsets P and Q such that $\sum_{i \in P} a_i = \sum_{i \in Q} a_i = b$?

Definition 16 *A* layered digraph $G = (N, A)$ *is a directed acyclic digraph whose set N is partitioned into disjoint subsets $N_0, N_1, \ldots, N_h$, such that if arc $(i, j) \in A$, then node $u \in N_i$, node $v \in N_j$, and $i < j$.*

Given a 2PP instance, we obtain a MSS-SPP instance $G = (N, A)$, which is a layered digraph with 3 layers, as follows. Let N be partitioned into 3 disjoint subsets of nodes N_0, N_1, and N_2. Subset N_0 contains only one node, r. Subset N_1 contains $2m$ nodes, and subset N_2 contains m nodes. Each node $u_i \in N_2$ represents an item $a_i \in I$. Besides, we define three disjoint subsets of arcs A_0, A_1, and A_2 such that $A = A_0 \cup A_1 \cup A_2$. Subset A_0 contains $2m$ arcs and establishes a complete bipartite digraph between N_0 and N_1. Subset A_1 contains m arcs, such that there is an arc from each node $n_i \in N_1$ to a node $u_i \in N_2$, for $0 \leq i < m$. Subset A_2 also contains m arcs, such that there is an arc from each node $n_{m+i} \in N_1$ to a node $u_i \in N_2$, for $0 \leq i < m$. We denote as A_1^i the arc from A_1 that reaches node $u_i \in N_2$. Similarly, we denote as A_2^i the arc from A_2 that reaches node $u_i \in N_2$. Finally, let $e = \frac{3}{2}b$

We define the intervals of the arcs as follows. The interval of arcs in A_0 are set to $[0, 0]$. In addition, we set the intervals of arc $A_1^i \in A_1$ to $[0, \frac{3}{2}a_i]$ and define the intervals of arc $A_2^i \in A_2$ to $[a_i, a_i]$. This construction corresponds to a layered digraph with 3 layers, as shown in Figure 5.3. In this construction, it is clear that the robustness cost of the paths for all nodes in $N_0 \cup N_1$ are zero. Furthermore, one can note that only one arc can be used to reach each node in N_2.

Theorem 8 *The decision version of MSS-SPP is NP-Complete, even in layered digraphs with only 3 layers.*

Proof. Consider a 2PP instance and a MSS-SPP instance as constructed above. It is easy to see that the problem is in NP, as the cost of a solution can be verified in

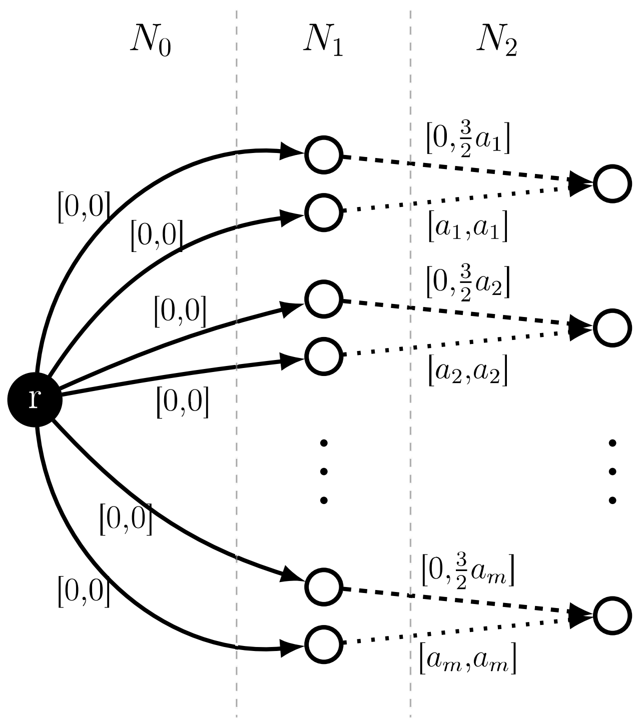

Figure 5.3. Reduction of the 2PP to the MSS-SPP. Arcs in A_0 are solid, arcs in A_1 are dashed and arcs in A_2 are dotted.

polynomial-time by running two Dijkstra's algorithms. We argue that a 2PP of I exists *if and only if* an arborescence $X \subseteq A$ exists such that $Z(X) = \frac{3}{2}b$.

$\Rightarrow$: Suppose there exists a partition of I in two subsets P and Q such that $\sum_{i \in P} a_i = \sum_{i \in Q} a_i = b$. A shortest path arborescence X uses arc A_1^i to reach node $u_i \in N_2$ if item $a_i \in P$, and uses arc A_2^i if item $a_i \in Q$. Thus, using the definition of the worst-case scenario [Aissi et al., 2009; Averbakh, 2001], we have that

$$Z(X) = \frac{3}{2}\sum_{i \in P} a_i + \sum_{i \in Q} a_i - \sum_{i \in P} a_i - \sum_{i \in Q} 0.$$

By the 2PP definition, we rewrite the above equation as

$$\begin{aligned} Z(X) &= \frac{3}{2}b + b - b \\ &= \frac{3}{2}b. \end{aligned}$$

$\Leftarrow$: Assume that $Z(X) = \frac{3}{2}b$. A partition of I into subsets P and Q can be represented as follows. If arc $A_1^i \in X$, then item $i \in P$. Otherwise, item $i \in Q$. Let $\sum_{i \in P} a_i = b_1$ and $\sum_{i \in Q} a_i = b_2$. Using the definition of the worst-case scenario [Aissi et al., 2009; Averbakh, 2001], we have that

$$Z(X) = \frac{3}{2}\sum_{i\in P} a_i + \sum_{i\in Q} a_i - \sum_{i\in P} a_i - \sum_{i\in Q} 0 = \frac{3}{2}b.$$

It follows that

$$\frac{3}{2}b = \frac{3}{2}b_1 + b_2 - b_1.$$

Since $b_1 + b_2 = 2b$, the above equation can only be true if $b_1 = b_2 = b$, which is a valid partition of I into subsets P and Q. □

From Theorem 8 we immediately obtain the computational complexity of MSS-SPP.

Corollary 9 *MSS-SPP is NP-Hard.*

Chapter 6

Computational experiments

The computational experiments were performed on a single core of an Intel Xeon CPU E5645 with a 2.4 GHz clock and 32 GB of RAM, running under the Linux Ubuntu operating system. The MILP and LP formulations were solved by the ILOG CPLEX solver, version 12.6, with default parameter settings. All algorithms were implemented in C++, along with the ILOG Concert Technology, and compiled with the GNU g++ 8.2.0. The parameters of SBA for both problems were set accordingly to Coco et al. [2015]. In addition, the maximum running time of all algorithms in any of the experiments was set to 3600 seconds.

6.1 Experiments with the M-WSCP

We used two set of benchmark instances in our computational experiments, namely *OR-Library* and *Shunji*. In both sets, the interval $[l_s, u_s]$, for all $s \in \mathcal{S}$, was computed as suggested by Karaşan et al. [2001]. Initially, a random value c_s was obtained from the original instance. Then, we set $l_s = \mathcal{U}\big((1-b)c_s, (1+b)c_s\big)$ and $u_s = \mathcal{U}\big(l_s + 1, (1+b)c_s\big)$, where $b \in \{0.3, 0.6, 0.9\}$ is a parameter that defines the degree of uncertainty of the instance, *i.e.*, the greater is the value of b, the greater is the difference between u_s and l_s, on average.

The set *OR-Library* of instances was retrieved from the OR-Library[1], a collection of test data sets for a variety of Operations Research (OR) problems. It was originally described by Beasley [1990] and encompass instances for several combinatorial optimization problems. We retrieved the instance sets 4, 5, and 6, which were first used for experiments for the WSCP by Balas and Ho [1980] and later adapted

[1]http://people.brunel.ac.uk/~mastjjb/jeb/info.html

for the M-WSCP [Pereira and Averbakh, 2013; Assunção et al., 2017]. Set 4 contains 10 instances with $|\mathcal{N}| = 200$ and $|\mathcal{S}| = 1000$, while set 5 contains 10 instances with $|\mathcal{N}| = 200$ and $|\mathcal{S}| = 2000$ and set 6 contains 5 instances with $|\mathcal{N}| = 200$ and $|\mathcal{S}| = 1000$. The weight values range between 1 and 100 for all instances. For each instance obtained from the OR-Library, we generated three instances for the M-WSCP, such that each generated instance use a different value of b. Therefore, there are 75 instances in this set.

The set *Shunji* of instances was generated using the instances generator for WSCP developed by Shunji Umetani[2]. This generator implements the scheme for constructing instances for WSCP proposed by Balas and Ho [1980] and guarantees that every elements in $\mathcal{N}$ is covered by, at least, two sets in $\mathcal{S}$, and guarantees that every set $s \in \mathcal{S}$ covers at least one element in $\mathcal{N}$. The instances within this set have $|\mathcal{N}| = 1000$ and $|\mathcal{S}| = 2000$. The weight values for all instances range between 1 and 100. Furthermore, it has a parameter $\mathcal{D} \in [0, 1]$ which controls the density of the covering matrix. Given the parameters $\mathcal{D} \in \{0.02, 0.05, 0.10\}$ and $b \in \{0.3, 0.6, 0.9\}$, we generated 5 random instances for each combination of these parameters. Therefore, there are 45 instances in this set.

6.1.1 Results for the exact algorithms

The first set of experiments aims at evaluating the efficiency of BLD, EB, and BC to solve M-WSCP using the instances in sets *OR-Library* and *Shunji*. The results are reported in tables 6.1 and 6.2. The first and second column respectively report the value of b and the set that the instance belongs. For BLD, the third, fourth, fifth, and sixth columns show respectively (*i*) the number of instances solved to optimality; (*ii*) the average number of cuts inserted by BLD and its standard deviation; (*iii*) the average relative optimality gap for the instances whose optimal solution was not found and the standard deviation of this same value; and (*iv*) the average running time (in seconds) for the instances whose optimal solutions were not found and the standard deviation of this same value. This same data is reported for EB in the seventh, eight, ninth, and tenth columns and for BC in the eleventh, twelfth, thirteenth, and fourteenth columns. When optimal solutions were not found for any of the five instances in a group, the fifth column is filled with a '$-$'. We recall that, for each value of b, sets 4 and 5 have 10 instances, whereas set 6, as so as the instances in *Shunji*, only have 5 instances each.

Regarding the set *OR-Library* of instances, it can be seen from Table 6.1 that all algorithms found optimal solutions for all evaluated instances. Furthermore, as greater

[2]https://sites.google.com/site/shunjiumetani/benchmark

Table 6.1. Results for the exact algorithms on *OR-Library* instances

		BLD				EB				BC			
b	set	opt	cuts	gap (%)	t (s)	opt	cuts	gap (%)	t (s)	opt	cuts	gap (%)	t (s)
	4	10	13.8 ± 9.0	0.0 ± 0.0	0.1 ± 0.1	10	103.4 ± 149.5	0.0 ± 0.0	0.1 ± 0.1	10	15.8 ± 9.0	0.0 ± 0.0	1.0 ± 1.2
0.3	5	10	13.5 ± 8.2	0.0 ± 0.0	4.2 ± 4.7	10	84.2 ± 89.8	0.0 ± 0.0	11.1 ± 14.7	10	16.3 ± 9.0	0.0 ± 0.0	2.0 ± 1.9
	6	5	6.8 ± 2.9	0.0 ± 0.0	6.6 ± 3.0	5	27.8 ± 14.5	0.0 ± 0.0	15.2 ± 9.5	5	8.2 ± 3.4	0.0 ± 0.0	6.6 ± 3.4
	4	10	41.2 ± 16.3	0.0 ± 0.0	19.9 ± 12.5	10	417.7 ± 409.5	0.0 ± 0.0	43.5 ± 46.9	10	48.4 ± 19.5	0.0 ± 0.0	6.7 ± 6.7
0.6	5	10	57.3 ± 29.7	0.0 ± 0.0	49.8 ± 48.0	10	1493.4 ± 1864.8	0.0 ± 0.0	263.9 ± 365.7	10	74.5 ± 42.3	0.0 ± 0.0	11.1 ± 7.3
	6	5	10.6 ± 5.0	0.0 ± 0.0	10.6 ± 8.7	5	51.2 ± 48.7	0.0 ± 0.0	25.2 ± 33.3	5	13.8 ± 4.0	0.0 ± 0.0	9.2 ± 5.7
	4	10	101.7 ± 49.8	0.0 ± 0.0	230.6 ± 378.7	10	1107.4 ± 1429.8	0.0 ± 0.0	163.0 ± 257.3	10	128.9 ± 64.0	0.0 ± 0.0	16.8 ± 18.0
0.9	5	10	158.7 ± 79.2	0.0 ± 0.0	804.9 ± 1125.4	10	3501.2 ± 2610.8	0.0 ± 0.0	1171.1 ± 1200.5	10	203.3 ± 103.2	0.0 ± 0.0	64.3 ± 62.8
	6	5	37.2 ± 9.4	0.0 ± 0.0	27.8 ± 13.7	5	380.8 ± 171.7	0.0 ± 0.0	74.4 ± 34.0	5	59.8 ± 30.3	0.0 ± 0.0	13.4 ± 9.3

Table 6.2. Results for the exact algorithms on *Shunji* instances

		BLD				EB				BC			
b	density	opt	cuts	gap (%)	t (s)	opt	cuts	gap (%)	t (s)	opt	cuts	gap (%)	t (s)
	0.02	4	15.4 ± 6.2	0.0 ± 0.0	2613.2 ± 1218.3	1	37.0 ± 15.6	0.1 ± 0.0	3071.0 ± 0.0	1	19.4 ± 6.8	0.3 ± 0.1	1194.0 ± 0.0
0.3	0.05	5	15.4 ± 7.8	0.0 ± 0.0	1369.4 ± 1339.1	3	52.4 ± 28.6	0.1 ± 0.1	1130.7 ± 750.5	4	20.4 ± 12.0	0.3 ± 0.0	1174.5 ± 948.7
	0.10	5	15.8 ± 7.4	0.0 ± 0.0	344.2 ± 218.6	5	121.4 ± 77.5	0.0 ± 0.0	1195.4 ± 839.7	5	28.8 ± 23.4	0.0 ± 0.0	430.8 ± 362.7
	0.02	0	16.6 ± 7.5	0.1 ± 0.1	–	0	19.8 ± 13.6	0.2 ± 0.1	–	0	16.4 ± 9.0	0.5 ± 0.1	–
0.6	0.05	1	40.0 ± 23.6	0.1 ± 0.0	3358.0 ± 0.0	0	76.4 ± 52.3	0.1 ± 0.0	–	1	63.4 ± 46.7	0.3 ± 0.2	3474.0 ± 0.0
	0.10	5	51.0 ± 35.8	0.0 ± 0.0	791.4 ± 830.6	3	461.6 ± 286.6	0.1 ± 0.0	1477.0 ± 1175.0	5	75.6 ± 45.5	0.0 ± 0.0	462.8 ± 401.7
	0.02	0	23.4 ± 8.4	0.1 ± 0.0	–	0	72.0 ± 55.9	0.2 ± 0.1	–	0	65.4 ± 40.2	0.3 ± 0.1	–
0.9	0.05	1	40.6 ± 18.4	0.1 ± 0.0	816.0 ± 0.0	1	323.6 ± 194.2	0.1 ± 0.0	2092.0 ± 0.0	2	239.0 ± 114.4	0.1 ± 0.1	2143.5 ± 1924.0
	0.10	3	77.2 ± 19.2	0.0 ± 0.0	1138.0 ± 647.5	0	706.8 ± 53.3	0.1 ± 0.0	–	5	138.0 ± 54.9	0.0 ± 0.0	989.0 ± 605.1

was the value of b, then greater was the average running time of the algorithms. BLD inserted the smallest average number of cuts for all evaluated instance sets, whereas EB inserted the greatest number of cuts, on average. One can see that BC was the fastest among the evaluated algorithms, being able to solve all instance sets within 65 seconds, on average. This smaller running time may be due to the fact that BC do not reinitialize its branching tree, as presented in Section 4.1.3, while BLD and EB restarts their branching tree at each iteration. Regarding the set *Shunji* of instances, it can be seen from Table 6.2 that, as denser is the instance, less time the algorithms need to found the optimal solution. EB found a smaller number of optimal solutions than BLD and BC. Furthermore, BLD found a greater number of optimal solutions for instances with $b = 0.3$, while BC found a greater number of optimal solutions for instances with $b = 0.9$. Despite that, the average relative gap achieved by BLD, EB, and BC were up to $0.5\,\%$, on average, being very close to the optimal solution. We choose to use BLD for solving the remaining formulation in FO-SBA and FO-LR, as it was able to found a greater number of optimal solutions than EB and BC.

6.1.2 Results on the preprocessing step of the fix-and-optimize heuristics

The second set of experiments evaluates how efficient are FO-SBA and FO-LR to fix the value of variables in the M-WSCP formulation defined by the objective function (5.4)

Table 6.3. Results for the FAO heuristics on *OR-Library* instances

		FO-SBA				FO-LR			
b	set	$\lvert\mathcal{S}'\rvert$	$\frac{\lvert\mathcal{S}'\rvert}{\lvert\mathcal{S}\rvert}$ (%)	t_p (s)	t_t (s)	$\lvert\mathcal{S}'\rvert$	$\frac{\lvert\mathcal{S}'\rvert}{\lvert\mathcal{S}\rvert}$ (%)	t_p (s)	t_t (s)
	4	73.7 ± 8.5	7.4	0.6 ± 1.1	0.1 ± 0.3	68.6 ± 9.5	6.9	0.0 ± 0.0	0.0 ± 0.0
0.3	5	76.4 ± 14.2	3.8	1.3 ± 1.8	0.3 ± 0.5	72.0 ± 10.7	3.6	0.1 ± 0.3	0.1 ± 0.3
	6	50.0 ± 11.0	5.0	8.2 ± 3.3	0.8 ± 0.8	56.2 ± 7.7	5.6	1.4 ± 0.8	1.5 ± 1.6
	4	86.0 ± 8.2	8.6	0.7 ± 0.8	0.1 ± 0.3	75.0 ± 8.0	7.5	0.0 ± 0.0	0.1 ± 0.3
0.6	5	89.0 ± 12.3	4.4	1.1 ± 1.1	0.2 ± 0.4	69.5 ± 7.0	3.5	0.0 ± 0.0	0.1 ± 0.3
	6	52.6 ± 9.5	5.3	5.8 ± 2.2	0.4 ± 0.5	47.8 ± 9.8	4.8	0.8 ± 0.8	0.6 ± 0.5
	4	91.6 ± 8.4	9.2	0.0 ± 0.0	0.0 ± 0.0	71.5 ± 5.7	7.1	0.0 ± 0.0	0.0 ± 0.0
0.9	5	96.2 ± 13.7	4.8	0.6 ± 0.7	0.2 ± 0.4	70.9 ± 8.0	3.5	0.0 ± 0.0	0.0 ± 0.0
	6	53.6 ± 7.4	5.4	2.2 ± 1.6	0.4 ± 0.5	45.6 ± 7.4	4.6	0.4 ± 0.5	0.0 ± 0.0

Table 6.4. Results for the FAO heuristics on *Shunji* instances

		FO-SBA				FO-LR			
b	density	$\lvert\mathcal{S}'\rvert$	$\frac{\lvert\mathcal{S}'\rvert}{\lvert\mathcal{S}\rvert}$ (%)	t_p (s)	t_t (s)	$\lvert\mathcal{S}'\rvert$	$\frac{\lvert\mathcal{S}'\rvert}{\lvert\mathcal{S}\rvert}$ (%)	t_p (s)	t_t (s)
	0.02	150.4 ± 23.6	7.5	1235.8 ± 585.1	81.6 ± 42.8	174.2 ± 34.0	8.7	241.4 ± 146.6	395.0 ± 365.2
0.3	0.05	90.2 ± 6.8	4.5	518.2 ± 399.3	38.8 ± 26.1	112.2 ± 4.1	5.6	72.8 ± 35.7	815.6 ± 706.1
	0.10	52.4 ± 8.4	2.6	222.0 ± 212.9	12.6 ± 15.1	69.4 ± 3.9	3.5	42.6 ± 35.6	36.0 ± 33.8
	0.02	143.6 ± 23.3	7.2	1592.2 ± 2345.1	33.0 ± 25.8	150.0 ± 32.6	7.5	430.8 ± 710.9	1160.2 ± 2361.5
0.6	0.05	98.6 ± 15.3	4.9	453.0 ± 389.1	206.4 ± 391.1	107.8 ± 4.9	5.4	80.6 ± 47.0	1203.0 ± 1503.4
	0.10	53.0 ± 7.3	2.6	64.2 ± 40.7	3.0 ± 0.7	61.6 ± 3.2	3.1	8.6 ± 4.8	15.8 ± 13.7
	0.02	184.8 ± 17.5	9.2	434.4 ± 384.5	81.6 ± 113.0	166.0 ± 13.0	8.3	109.6 ± 131.4	275.2 ± 368.8
0.9	0.05	99.8 ± 11.7	5.0	154.2 ± 102.5	6.4 ± 2.2	88.0 ± 11.4	4.4	26.8 ± 14.6	13.0 ± 6.4
	0.10	53.2 ± 5.1	2.7	29.8 ± 10.4	1.6 ± 1.3	50.8 ± 6.8	2.5	5.0 ± 2.3	1.4 ± 0.5

and the constraints in (5.5)–(5.8). As the number of binary variables is the same as the cardinality of the set $\mathcal{S}$, we measure as $|\mathcal{S}'|$ how many subsets of $\mathcal{S}$ remain in the instance after the preprocessing step of each heuristic and how long it takes to solve the resulting formulation.

The results of this experiment are presented in tables 6.3 and 6.4 for the *OR-Library* and *Shunji* instances, respectively. The first column reports the value of b, while the second shows the set of the instance (for *OR-Library* instances) or the density value $\mathcal{D}$ (for *Shunji* instances). The third column gives the average and the standard deviation of the number $|\mathcal{S}'|$ of subsets of $\mathcal{S}$ that remained in the instance after the preprocessing step of FO-SBA, while the fourth column presents the average ratio $\mathcal{S}'/\mathcal{S}$, *i.e.*, the proportion of subsets of $\mathcal{S}$ that remained in the instance. The average running times, along with their standard deviations, for the preprocessing and solving steps of FO-SBA are presented in the fifth and sixth columns, respectively. The same data is reported for FO-LR in the last four columns. As pointed out in Section 6.1.1, the solving step of both FO-SBA and FO-LR employs BLD.

Regarding the *OR-Library* instances, one can see from Table 6.3 that both heuristics obtained similar results. For both FO-SBA and FO-LR, the greatest average reduction was achieved in instances from set 5, while the smallest average reduction was achieved in instances from set 4. FO-SBA average preprocessing running time never exceeded 9 seconds and was able to fix more than 90 % of the variables of the formulation, on average, for all evaluated instances. FO-LR was faster than FO-SBA, such that the maximum running time of its preprocessing step was 1.4 seconds for instances from set 6 with $b = 0.3$. Using this reduced variable set, the BLD average running time never exceeded 1 second for FO-SBA and 1.5 second for FO-LR. In addition, one can also see that the accumulated running times of the preprocessing step and the solving step of FO-SBA and FO-LR, computed as $t_p + t_t$, were significantly smaller than those of the exact algorithms, as reported in Table 6.1.

Regarding the *Shunji* instances, one can see from Table 6.4 that, the denser was the instance, then smaller was the running time of FO-SBA and FO-LR, on average. Besides, the heuristics also fixed a greater number of variables in denser instances than on the sparser instances, such that only 2.7 % and 2.5 % of the original variables were not fixed for the instances with $b = 0.3$ and $\mathcal{D} = 0.10$ using FO-SBA and FO-LR, respectively. The average running time of FO-SBA preprocessing step was up to 1592 seconds on intances with $b = 0.6$ and $\mathcal{D} = 0.05$, while that of FO-LR was up to only 430 seconds on instances with $b = 0.6$ and $\mathcal{D} = 0.05$. However, the solving step of FO-SBA was faster than that of FO-LR for all evaluated instance subsets, being up to 206 seconds on instances with $b = 0.6$ and density of 0.05.

6.1.3 Comparison of the heuristics

The last set of experiments with M-WSCP compares the results of the proposed heuristics. The results reported in tables 6.5 and 6.6 for the *OR-Library* and *Shunji* instances, respectively. The first column reports the value of b, while the second shows the set of the instance (for *OR-Library* instances) or the density value $\mathcal{D}$ (for *Shunji* instances). The third and fourth columns report the results of SBA. Let $\bar{X}$ the best known solution for each instance (found by any of the exact or heuristic algorithms used in our experiments), the third column gives the *average relative deviation* $\frac{Z(SBA)-Z(\bar{X})}{Z(\bar{X})}$, where $Z(SBA)$ is the robustness cost of the solution obtained by SBA, along with the standard deviation of this same metric. The fourth column presents the average and the standard deviation of the running time of SBA. The same data is reported for LPH in the fifth and sixth columns, for FO-SBA in the seventh and eighth columns, and for FO-LR in the ninth and tenth columns.

Regarding the *OR-Library* instances, one can see from Table 6.5 that the average

Table 6.5. Results for the heuristics on *OR-Library* instances

		SBA		LPH		FO-SBA		FO-LR	
b	set	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)
0.3	4	4.9 ± 9.2	0.6 ± 1.1	8.6 ± 9.7	0.2 ± 0.4	4.6 ± 8.2	0.7 ± 1.1	6.5 ± 10.0	0.0 ± 0.0
	5	9.0 ± 11.6	1.3 ± 1.8	13.7 ± 12.6	0.1 ± 0.3	9.0 ± 11.6	1.6 ± 1.6	10.1 ± 10.8	0.2 ± 0.4
	6	11.9 ± 7.6	8.2 ± 3.3	54.4 ± 29.0	0.4 ± 0.5	9.4 ± 6.8	9.0 ± 3.8	9.4 ± 6.8	3.0 ± 2.3
0.6	4	10.4 ± 17.0	0.7 ± 0.8	10.2 ± 14.8	0.3 ± 0.5	9.9 ± 17.2	0.8 ± 1.0	10.7 ± 16.9	0.1 ± 0.3
	5	7.0 ± 6.3	1.1 ± 1.1	6.6 ± 6.8	0.8 ± 0.6	7.0 ± 6.3	1.3 ± 0.9	7.6 ± 7.8	0.1 ± 0.3
	6	2.8 ± 6.2	5.8 ± 2.2	15.7 ± 2.4	0.6 ± 0.5	2.8 ± 6.2	6.2 ± 2.2	5.1 ± 5.7	1.4 ± 1.1
0.9	4	9.0 ± 11.3	0.1 ± 0.0	7.2 ± 11.6	0.3 ± 0.5	9.0 ± 11.4	0.0 ± 0.0	9.7 ± 11.1	0.0 ± 0.0
	5	2.9 ± 2.8	0.6 ± 0.7	1.7 ± 2.6	0.7 ± 0.7	2.9 ± 2.8	0.8 ± 0.6	3.8 ± 3.2	0.0 ± 0.0
	6	2.7 ± 1.9	2.2 ± 1.6	2.8 ± 1.7	1.0 ± 0.0	2.4 ± 2.2	2.6 ± 1.5	3.5 ± 2.9	0.4 ± 0.5

Table 6.6. Results for the heuristics on *Shunji* instances

		SBA		LPH		FO-SBA		FO-LR	
b	density	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)
0.3	0.02	4.9 ± 3.3	1235.8 ± 585.1	73.1 ± 23.0	48.2 ± 53.2	4.9 ± 3.3	1317.4 ± 571.8	4.9 ± 3.3	636.4 ± 403.6
	0.05	6.9 ± 7.6	518.2 ± 339.3	79.4 ± 24.5	39.8 ± 33.1	6.9 ± 7.6	557.0 ± 421.7	3.3 ± 7.5	888.4 ± 729.1
	0.10	7.2 ± 5.1	222.0 ± 212.9	71.1 ± 29.9	62.0 ± 58.2	6.0 ± 6.1	234.6 ± 209.7	6.0 ± 6.1	78.6 ± 52.9
0.6	0.02	0.9 ± 1.0	1592.2 ± 2345.1	25.8 ± 5.6	90.6 ± 68.6	0.9 ± 1.0	1625.2 ± 2327.7	2.1 ± 2.5	1591.0 ± 3072.2
	0.05	2.6 ± 2.8	453.0 ± 389.1	34.6 ± 12.5	335.4 ± 569.0	2.6 ± 2.8	659.4 ± 770.4	3.8 ± 2.3	1283.6 ± 1534.1
	0.10	4.9 ± 3.5	64.2 ± 40.7	21.7 ± 7.7	25.8 ± 23.6	4.9 ± 3.5	67.2 ± 41.3	6.6 ± 5.7	24.4 ± 18.2
0.9	0.02	0.0 ± 0.0	434.4 ± 384.5	6.8 ± 1.9	184.8 ± 202.4	0.0 ± 0.0	516.0 ± 493.4	2.4 ± 2.5	384.8 ± 499.8
	0.05	0.7 ± 1.0	154.2 ± 102.5	8.6 ± 3.5	166.8 ± 120.5	0.7 ± 1.0	160.6 ± 103.2	1.4 ± 0.9	39.8 ± 18.8
	0.10	2.2 ± 2.3	29.8 ± 10.4	5.0 ± 2.1	29.2 ± 13.6	1.7 ± 2.5	31.4 ± 10.5	2.8 ± 2.7	6.4 ± 2.3

running time of the heuristics never exceeded 9 seconds, which was the case of FO-SBA for instances from set 6 with $b = 0.3$. LPH was the fastest among the evaluated heuristics. However, it also found the greater average relative deviations among the evaluated heuristics, being up to 54.4 % for instances from set 6 with $b = 0.3$. The best results for all instances were obtained by FO-SBA, such that it achieved a maximum average relative deviation of 9.9 % for instances from set 4 with $b = 0.6$. Besides, one can observe that FO-SBA average running time never exceeds 9 seconds, and that it was able to improve the average relative deviations of SBA for 4 out of the 9 subset of instances evaluated. These results indicate that FO-SBA was able to fastly compute near-optimal solutions, since the optimal solution for all *OR-Library* instances were given by BLD, EB, and BC, as shown in Table 6.1.

Regarding the *Shunji* instances, one can see from Table 6.6 that the results were very similar to those obtained for the *OR-Library* instances. LPH obtained the greatest average relative deviations for all subset of instances. On the other hand, the FO-SBA solutions had the smallest average relative deviation among the evaluated heuristics for all instance subsets, except for instances with $b = 0.3$ and $\mathcal{D} = 0.05$. One can observe that FO-SBA was able to improve the results of SBA for instances with $b = 0.3$ and

$\mathcal{D} = 0.10$ and for instances with $b = 9$ and $\mathcal{D} = 0.10$. Furthermore, FO-LR obtained competitive results, being faster than FO-SBA for 7 out of the 9 subset of instances evaluated and computing solutions with an average relative deviation at least as good as those of FO-SBA for 3 out of the 9 subset of instances evaluated.

The results from tables 6.5 and 6.6 point out to the fact that FO-SBA obtained the best results among the evaluated heuristics. To test this observation for *OR-Library* and *Shunji* instances, we analyzed our experimental data following the statistical procedure of Garcia and Herrera [2008], which is composed of three steps. The first step verifies the non-normality of the relative deviations of SBA, LPH, FO-SBA, and FO-LR. Next, the second step evaluates whether there is a statistical significant difference among the four heuristics. Then, in case such a difference exists, the third step verifies whether the relative deviations of the heuristics are significantly different among them. These steps are detailed below. They assume a significance level $\alpha = 0.05$, i.e., the null hypothesis is rejected if the p-value is smaller than 0.05.

In the first step, we applied a Shapiro-Wilk test of normality [Shapiro and Wilk, 1965] to verify whether the relative deviations of SBA, LPH, FO-SBA and FO-LR follow a normal distribution. With a p-value of 0.001, the test indicated that the data of the three heuristics does not follow a normal distribution. Thus, a non-parametric test is used in the next step.

In the second step, we applied the Friedman's test [Friedman, 1937] to verify whether there is a statistically significant difference between at least two of the evaluated heuristics. The null hypothesis was that SBA, LPH, FO-SBA and FO-LR have the same relative deviation, on average. The data were ranked according to Carvalho [2019]. With a p-value of 0.002, the test rejected the null hypothesis for both *OR-Library* and *Shunji* instances. Therefore, there is indeed a significant difference in the relative deviations of SBA, LPH, FO-SBA and FO-LR.

In the third step, we applied a non-parametric two-tailed Nemenyi's post-hoc test, also known as the Nemenyi–Damico–Wolfe–Dunn post-hoc test [Nemenyi, 1962], which compares the results of multiple algorithms. This test evaluated the pair of hypothesis

$$\begin{cases} H_0: & \mu_i \geq \mu_j \\ H_1: & \mu_i \neq \mu_j \end{cases}, \qquad \forall(\mu_i, \mu_j) \in M,$$

where $M = \{\, \mu_{\text{sba}}, \mu_{\text{lph}}, \mu_{\text{fo-sba}}, \mu_{\text{fo-lr}} \,\}$, such that $\mu_{\text{sba}}, \mu_{\text{lph}}, \mu_{\text{fo-sba}}$, and $\mu_{\text{fo-lr}}$ are, respectively, the average of the rankings obtained by SBA, LPH, FO-SBA, and FO-LR in the second step. The null hypothesis (H_0) states that the average ranking of μ_i and μ_j was not significantly different among them, thus implying that the results of one of the evaluated heuristics is not significantly better than those of the other. On the other hand, the alternative hypothesis (H_1) implies that indeed the results of μ_i is significantly different than those of μ_j.

Table 6.7 presents the results of the Nemenyi's test for *OR-Library* and *Shunji* instances. Each cell of this table displays the p-value obtained by comparing the heuristics displayed on the top of the column and on the beginning of the row.

Table 6.7. p-values obtained by the Nemenyi's test on *OR-Library* and *Shunji* instances

	OR-Library			Shunji		
	SBA	LPH	FO-SBA	SBA	LPH	FO-SBA
LPH	0.52	-	-	0.01	-	-
FO-SBA	0.93	0.20	-	1.00	0.01	-
FO-LR	0.14	0.86	0.03	0.74	0.01	0.61

Regarding the *OR-Library* instances, one can see from Table 6.7 that only FO-SBA and FO-LR significantly differ among them, with a p-value of 0 03. For the remaining of the comparisons, the test could not reject the null hypothesis and, thus, we cannot assume that the average relative deviation of the heuristics significantly differ among them. Thus, we conclude that SBA, LPH, and FO-SBA results do not significantly differ among them. For this set of instances, the recommended heuristic is LPH, since it had a smaller running time in comparison to SBA and FO-SBA, as shown in Table 6.5.

Regarding the *Shunji* instances, one can see from Table 6.7 that LPH was the worst heuristic for these instances, as the Nemenyi's test returned a p-value of 0 01 for the comparison of LPH with all other heuristics. The comparison of SBA and FO-SBA returned a p-value of 1 00, while the comparisons of FO-LR with SBA and FO-SBA returned a p-value of 0 74 nad 0 61, respectively. Thus, the test could not reject the null hypothesis for these comparisons and we cannot assume that the average relative deviations of SBA and FO-SBA significantly differ among them. Additionally, we also cannot assume that the average relative deviations of FO-LR, SBA, and FO-SBA significantly differ among them. Therefore, for this set of instances, the recommended heuristic is FO-LR, since its average running time was smaller than those of SBA and FO-SBA for 7 out of the 9 subset of instances, as shown in Table 6.6.

6.2 Experiments with the MSS-SPP

We used two sets of benchmark instances in our computational experiments, namely *Random* and *Layered*. In both sets, the interval $[l_{ij}, u_{ij}]$, for all $(i, j) \in A$, was computed as suggested by Karaşan et al. [2001]. Initially, a random value c_{ij} is generated from a uniform distribution $\mathcal{U}(1, 100)$. Then, we set $l_{ij} = \mathcal{U}\big((1-b)c_{ij}, (1+b)c_{ij}\big)$ and $u_{ij} = \mathcal{U}\big(l_{ij}+1, (1+b)c_{ij}\big)$, where $b \in \{0.3, 0.6, 0.9\}$ is a parameter that defines the degree of uncertainty of the instance, *i.e.*, the greater is the value of b, the greater is the difference between u_{ij} and l_{ij}, on average.

The set *Random* of instances was used in the computation experiments of Catanzaro et al. [2011]; Taccari [2016], and Martinovic et al. [2020]. It consists of random digraphs generated accordingly to the Gilbert model [Gilbert, 1959]. Given the parameter $D \in \{0.25, 0.50, 0.75, 1.00\}$ of the density coefficient of the graph and the parameter $n \in \{100, 200, 300\}$ of the number of nodes in N, *i.e.*, $n = |N|$, five instances were randomly generated for each of the 36 combinations of the values of D, n and b. Thus, there are 180 instances in this set.

The set *Layered* of instances is one of the most used instances benchmark in the literature of Minmax regret Shortest Path Problem under Interval Uncertainty (M-SPP) and Minmax regret Resource-Constrained Shortest Path Problem under Interval Uncertainty (MRC-SPP) [Pérez-Galarce et al., 2018; Assunção et al., 2017; Montemanni and Gambardella, 2004; Coco et al., 2014a]. They consist of digraphs with L layers, each with the same number ω of nodes, where there is an arc from every node in layer $l \in \{1, \ldots, L-1\}$ to every node in layer $l+1$. Besides, they also contain a source node r and a destination node f, such that there is an arc from node r to every node in the first layer and an arc from every node in the last layer to the destination node f. An example of a layered digraph with $L = 4$ and $\omega = 3$ is shown in Figure 6.1. Given the parameters $\omega \in \{5, 10, 25, 50\}$, $n \in \{102, 152, 202\}$, and $b \in \{0.3, 0.6, 0.9\}$, five instances were randomly generated for each of the 36 combinations of values of ω, $L = (n-2)/\omega$ and b. Thus, there are 180 instances in this set.

6.2.1 Results for the MILP formulation

The first set of experiments aims at evaluating how efficient is the MILP formulation (5.15)–(5.23) to solve the instances in sets *Random* and *Layered*. The results are reported in tables 6.8 and 6.9, respectively. The first and second column respectively report the number of nodes and the number of arcs in each group of five instances generated with the same parameter setting. The third column gives the value of D (for the *Random* digraphs) or L (for the *Layered* digraphs). For the instances with $b = 0.3$, the fourth, fifth and sixth columns show respectively (*i*) the number of instances solved to optimality; (*ii*) the average relative optimality gap for the five instances within 3600 seconds and the standard deviation of this value; and (*iii*) the average running time

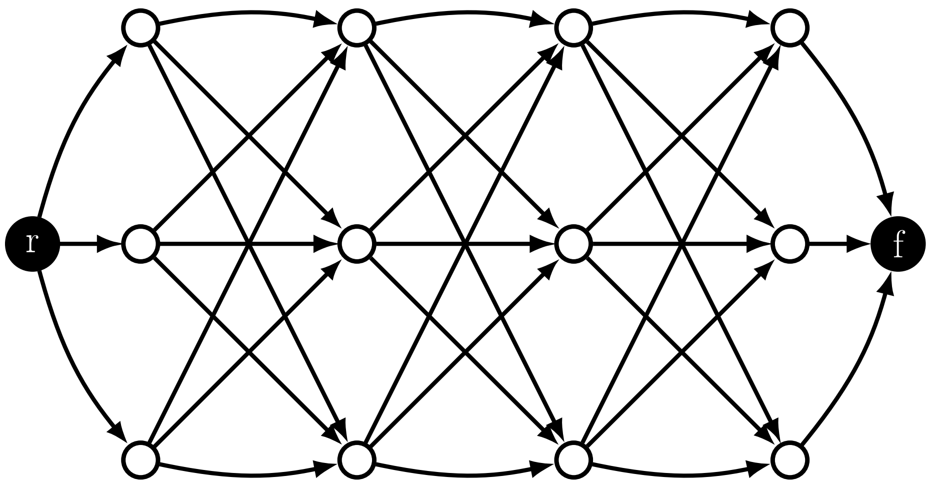

Figure 6.1. A layered digraph with $L = 4$ layers of $\omega = 3$ nodes

(in seconds) for the instances where an optimal solution was found and its standard deviation. When optimal solutions were not found for any of the five instances in a group, the fifth column is filled with a '$-$'. Besides, when the MILP formulation could not be solved due to the lack of available RAM, the fourth and fifth columns are filled with an '$*$'. The same data is reported for the instances with $b = 0.6$ in the seventh, eighth and ninth columns, and for those with $b = 0.9$ in the tenth and eleventh, and twelfth columns, respectively.

Regarding the set *Random* of instances, it can be seen from Table 6.8 that the CPLEX branch-and-bound algorithm could solve all of the instances in this set with up to 200 nodes within 3000 seconds, on average. One can also see that the value of the degree of uncertainty parameter b did not greatly influenced the running time of the algorithm. For the largest instances with 300 nodes, one can observe that only those with the smallest arc density $D = 0.25$ could be solved. The other instances with more than 40,000 arcs could not be solved due to the lack of available RAM. Regarding the set *Layered* of instances, it can be seen from Table 6.9 that, the smaller is the value of b, the longer it takes is to solve the instance. For $b = 0.3$, optimal solutions could be found only for instances with up to $L = 6$ layers, while all the instances with $b = 0.9$ were solved to optimality within 3600 seconds. These results motivate the use of heuristics as many instances could not be solved to optimality and the largest *Random* instances could not even be run due to the large amount of memory necessary to run MILP formulation (5.15)–(5.23).

6.2.2 Results on the preprocessing step of the fix-and-optimize heuristics

The second set of experiments evaluates how efficient are FO-SBA and FO-LR to fix the value of variables in the formulation (5.15)–(5.23). As the number of variables is proportional to the number or arcs, we measure how many arcs remain in the instance after the preprocessing step of each heuristic and how long it takes to solve the resulting formulation.

The results are presented in tables 6.10 and 6.11 for the *Random* and *Layered* instances, respectively. The first column reports the instance size, while the second shows the value of D (for *Random* digraphs) or L (for *Layered* digraphs). The third and forth columns give the average, over the 15 instances with $b = 0.3$, $b = 0.6$ and $b = 0.9$, of the number $|A'|$ of arcs that remained in the instance after the preprocessing step of FO-SBA, and the value of the ratio $|A'|/|A|$, i.e., the proportion of arcs that remained in the instance. The average running times of the preprocessing the solving steps of FO-SBA are presented in the fifth and sixth columns, respectively. When the MILP formulation could not be solved due to the lack of available RAM, columns 3 to 6 are filled with an '$*$'. The same data is reported for FO-LR in the last four columns.

Regarding the *Random* instances, one can see from Table 6.10 that FO-LR could not be run on the largest instances with $|N| = 300$ and $D = 1.00$ due to the lack of RAM. For the other instances, the number $|A'|$ of remaining arcs after the preprocessing step of both heuristics were significantly smaller than $|A|$. The largest average value of $|A'|/|A|$ for FO-SBA and FO-LR was respectively 4.4 % and 5.6 %, on the instances with $|N| = 100$ and $D = 0.25$. The running times t_p (in seconds) of the preprocessing step of FO-SBA were much smaller than those of FO-LR, on average. On the largest instances where both heuristics could be run, *i.e.*, those with $|N| = 300$ and $D = 0.75$, the value of t_p for FO-SBA was 106 5 seconds, while that for FO-LR was 1880 2 seconds. In addition, one can also see that the running times t_s (in seconds) of the solving step of FO-SBA and FO-LR were significantly smaller than those reported in Table 6.8 for the CPLEX branch-and-bound algorithm.

Regarding the *Layered* instances, one can see from Table 6.11 that again the number $|A'|$ of remaining arcs after the preprocessing step of both heuristics were significantly smaller than $|A|$. The reduction observed for this set of instances was not as greater as that observed for *Random* instances. Additionally, one can see the preprocessing step of FO-SBA was more efficient than that of FO-LR for this set of instances. For the largest instances with $|N| = 202$ and $L = 40$, the average value of $|A'|/|A|$ for FO-SBA was 23.6 %, while that for FO-LR was 37.5 %. This result may

Table 6.8. Results for the MILP formulation on random graphs

			$b = 0.3$			$b = 0.6$			$b = 0.9$		
$\|N\|$	$\|A\|$	D	opt	gap (%)	t (s)	opt	gap (%)	t (s)	opt	gap (%)	t (s)
100	2447	0.25	5	0.0 ± 0.0	24.8 ± 3.4	5	0.0 ± 0.0	25.4 ± 3.0	5	0.0 ± 0.0	25.0 ± 3.7
	4935	0.50	5	0.0 ± 0.0	49.6 ± 5.7	5	0.0 ± 0.0	47.0 ± 11.7	5	0.0 ± 0.0	41.4 ± 7.1
	7436	0.75	5	0.0 ± 0.0	45.6 ± 2.2	5	0.0 ± 0.0	52.0 ± 12.6	5	0.0 ± 0.0	52.0 ± 11.1
	9900	1.00	5	0.0 ± 0.0	103.4 ± 16.3	5	0.0 ± 0.0	99.4 ± 4.3	5	0.0 ± 0.0	96.0 ± 16.1
200	9969	0.25	5	0.0 ± 0.0	332.8 ± 25.0	5	0.0 ± 0.0	340.6 ± 15.4	5	0.0 ± 0.0	407.4 ± 85.5
	19848	0.50	5	0.0 ± 0.0	1022.0 ± 164.5	5	0.0 ± 0.0	743.2 ± 65.3	5	0.0 ± 0.0	940.2 ± 110.0
	29909	0.75	5	0.0 ± 0.0	968.4 ± 296.9	5	0.0 ± 0.0	1416.6 ± 395.5	5	0.0 ± 0.0	1504.0 ± 333.7
	39800	1.00	5	0.0 ± 0.0	2660.6 ± 832.1	5	0.0 ± 0.0	2962.4 ± 368.7	5	0.0 ± 0.0	2671.2 ± 354.4
300	22283	0.25	4	0.7 ± 1.6	2361.5 ± 509.9	5	0.0 ± 0.0	2128.8 ± 550.7	4	20.0 ± 44.7	2581.6 ± 618.7
	44742	0.50	0	*	*	0	*	*	0	*	*
	67638	0.75	0	*	*	0	*	*	0	*	*
	89700	1.00	0	*	*	0	*	*	0	*	*

Table 6.9. Results for the MILP formulation on layered graphs

			$b = 0.3$			$b = 0.6$			$b = 0.9$		
$\|N\|$	$\|A\|$	L	opt	gap (%)	t (s)	opt	gap (%)	t (s)	opt	gap (%)	t (s)
102	2600	2	5	0.0 ± 0.0	0.4 ± 0.5	5	0.0 ± 0.0	0.2 ± 0.4	5	0.0 ± 0.0	0.2 ± 0.4
	1925	4	5	0.0 ± 0.0	7.8 ± 2.3	5	0.0 ± 0.0	6.8 ± 2.4	5	0.0 ± 0.0	3.8 ± 0.8
	920	10	0	8.0 ± 5.3	–	5	0.0 ± 0.0	306.2 ± 157.8	5	0.0 ± 0.0	5.2 ± 2.4
	485	20	0	14.0 ± 1.9	–	5	0.0 ± 0.0	232.2 ± 218.6	5	0.0 ± 0.0	4.2 ± 2.3
152	5100	3	5	0.0 ± 0.0	16.4 ± 1.8	5	0.0 ± 0.0	15.0 ± 1.6	5	0.0 ± 0.0	12.2 ± 0.8
	3175	6	2	1.0 ± 1.1	2495.5 ± 1125.0	2	1.8 ± 2.2	1017.0 ± 1118.6	5	0.0 ± 0.0	20.8 ± 3.6
	1420	15	0	27.8 ± 1.9	–	0	5.3 ± 3.8	–	5	0.0 ± 0.0	114.8 ± 110.3
	735	30	0	26.9 ± 1.4	–	3	1.1 ± 1.9	1129.7 ± 1309.0	5	0.0 ± 0.0	19.4 ± 19.3
202	7600	4	5	0.0 ± 0.0	243.6 ± 260.7	5	0.0 ± 0.0	103.2 ± 47.6	5	0.0 ± 0.0	38.6 ± 1.9
	4425	8	0	10.9 ± 2.2	–	0	11.7 ± 3.9	–	5	0.0 ± 0.0	449.2 ± 673.7
	1920	20	0	40.5 ± 3.5	–	0	18.4 ± 5.5	–	5	0.0 ± 0.0	1067.2 ± 1148.8
	985	40	0	35.4 ± 1.2	–	1	9.8 ± 9.2	3183.0 ± 0.0	5	0.0 ± 0.0	98.0 ± 54.5

Table 6.10. Reduction achieved by FO-SBA and FO-LR on random digraphs

		FO-SBA				FO-LR			
$\|N\|$	D	$\|A'\|$	$\frac{\|A'\|}{\|A\|}$ (%)	t_p (s)	t_s (s)	$\|A'\|$	$\frac{\|A'\|}{\|A\|}$ (%)	t_p (s)	t_s (s)
100	0.25	109.8 ± 3.1	4.4	0.0 ± 0.0	0.0 ± 0.0	137.7 ± 6.0	5.6	3.9 ± 0.9	0.6 ± 0.5
	0.50	112.1 ± 3.9	2.3	0.2 ± 0.1	1.0 ± 0.0	139.1 ± 6.8	2.8	6.9 ± 1.9	1.4 ± 0.5
	0.75	113.3 ± 3.4	1.5	0.4 ± 0.2	1.9 ± 0.3	141.6 ± 5.8	1.9	9.5 ± 3.1	1.9 ± 0.5
	1.00	114.9 ± 4.0	1.2	0.6 ± 0.2	2.0 ± 0.0	144.1 ± 8.3	1.5	12.6 ± 3.0	2.3 ± 0.5
200	0.25	224.9 ± 4.2	2.3	0.7 ± 0.3	6.0 ± 0.0	282.6 ± 10.2	2.8	61.4 ± 30.4	6.1 ± 0.4
	0.50	237.6 ± 10.0	1.2	1.5 ± 0.4	12.3 ± 0.5	292.9 ± 11.4	1.5	135.9 ± 77.3	14.3 ± 1.0
	0.75	243.0 ± 7.6	0.8	2.2 ± 0.5	19.3 ± 0.7	304.5 ± 8.0	1.0	191.7 ± 97.8	22.5 ± 0.9
	1.00	246.8 ± 11.1	0.6	2.9 ± 0.5	22.6 ± 0.7	311.5 ± 13.6	0.8	291.5 ± 110.6	27.7 ± 0.7
300	0.25	351.9 ± 8.3	1.6	3.0 ± 0.4	24.3 ± 0.5	438.9 ± 12.0	2.0	805.7 ± 494.9	23.2 ± 0.6
	0.50	366.7 ± 12.1	0.8	4.4 ± 0.4	58.7 ± 1.0	460.6 ± 10.5	1.0	1295.6 ± 603.9	252.7 ± 17.1
	0.75	382.4 ± 11.4	0.6	6.1 ± 0.4	160.5 ± 3.5	483.4 ± 21.7	0.7	1880.2 ± 1010.3	455.3 ± 425.7
	1.00	387.9 ± 11.0	0.4	7.9 ± 0.6	654.1 ± 47.8	*	*	*	*

explain why the running times t_s (in seconds) of the solving step of FO-SBA were significantly smaller than those of FO-LR.

6.2.3 Comparison of the heuristics

The last set of experiments compares the results of the proposed heuristics. The results reported in tables 6.12 and 6.13 for the *Random* and *Layered* instances, respectively. The first column shows the instance size, while the second reports the value of D (for *Random* instances) or L (for *Layered* instances). The third and fourth columns report the results of SBA. Let $\bar{X}$ the best known solution for each instance (found by any of the algorithms used in our experiments), the third column gives the *average relative deviation* $\frac{Z(SBA)-Z(\bar{X})}{Z(\bar{X})}$, over the 15 instances with $b = 0.3$, $b = 0.6$ and $b = 0.9$, where $Z(SBA)$ is the robustness cost of the solution obtained by SBA. Besides, the fourth column presents the average running time of this heuristic over these 15 instances. The same data is reported for MILP-basedVariable Neighborhood Descent (VND) in the fifth and sixth columns, for FO-SBA in the seventh and eighth columns, and for FO-LR in the ninth and tenth columns. When a group of instances could not be solved due to the lack of available RAM, the columns are filled with an '$*$'.

Regarding the *Random* instances, one can see from Table 6.12 that only SBA and FO-SBA were able to solve the largest instances with 300 nodes and $D = 1.00$. SBA was able to solve these instances because it do not rely on the formulation 5.15–5.23, while FO-SBA solves this formulation in a reduced instance. In addition, the largest instances where VND could be run have 300 nodes and $D = 0.25$, and those of FO-LR have 300 nodes and $D = 0.75$. SBA was the fastest among the evaluated heuristics with a maximum average running time of 7 9 seconds. VND was able to improve the results of SBA on all instances where it could run, but it achieved the greatest average running time among the evaluated heuristics, such as the average running times of FO-SBA and FO-LR were always smaller than those of VND. In addition, it can be seen that the average relative deviation of FO-LR, over the best known solution for each instance, was significantly smaller than those of SBA, FO-SBA, and VND. The largest average relative deviation of FO-LR was only 0.4 %, on the instances with $|N| = 100$ and $D = 0.25$, while that of FO-SBA was 30.6 %, on the instances with $|N| = 100$ and $D = 0.50$, and that of VND was 35.4%, on the instances with $|N| = 200$ and $D = 0.25$.

Regarding the *Layered* instances, one can see from Table 6.11 that SBA had the smallest average running time, being up to 0 7 seconds on instances with $|N| = 202$ and $L = 4$. The greatest average running time of FO-LR was 2198 5 seconds, on the instances with $|N| = 202$ and $L = 20$, while that of FO-SBA was 4 5 seconds, on the instances with $|N| = 202$ and $L = 4$, and that of VND was 327 1 seconds for the same instances. This may be explained by the fact that the preprocessing step of FO-LR on these instances is not as efficient to reduce the number of variables as it is on the *Random* instances. However, it can be seen that the average relative deviation of FO-LR, over the best known solution for each instance, was significantly smaller than those of SBA, FO-SBA, and VND. The largest average relative deviation of FO-LR was only 0.7 %, on the instances with $|N| = 102$ and $L = 20$, while that of FO-SBA was 20.5 %, on the instances with $|N| = 202$ and $L = 40$, and that of VND was 15.4 % on the instances with $|N| = 202$ and $L = 20$.

The results from tables 6.12 and 6.13 point out to the fact that FO-LR obtained the best results among the evaluated heuristics. To test this observation for *Random* and *Layered* instances, we analyzed our experimental data following the statistical procedure of Garcia and Herrera [2008]. This is the same methodology applied for evaluating the experiments with the M-WSCP instances, as shown in Section 6.1.3.

In the first step, we applied a Shapiro-Wilk test of normality [Shapiro and Wilk, 1965] to verify whether the relative deviations of SBA, VND, FO-SBA, and FO-LR follow a normal distribution. With a p-value of 0 001, the test indicated that the data of the three heuristics does not follow a normal distribution. Thus, a non-parametric test is used in the next step.

In the second step, we applied the Friedman's test [Friedman, 1937] to verify whether there is a statistically significant difference between at least two of the evaluated heuristics. The null hypothesis was that SBA, VND, FO-SBA and FO-LR have the same relative deviation, on average. The data were ranked according to Carvalho [2019]. With a p-value of 0.001, the test rejected the null hypothesis for both *Random* and *Layered* instances. Therefore, there is indeed a significant difference in the relative deviations of SBA, VND, FO-SBA and FO-LR.

In the third step, we applied a non-parametric two-tailed Nemenyi's post-hoc test, also known as the Nemenyi–Damico–Wolfe–Dunn post-hoc test [Nemenyi, 1962], which compares the results of multiple algorithms. This test evaluated the pair of hypothesis

$$\begin{cases} H_0: & \mu_i \geq \mu_j \\ H_1: & \mu_i \neq \mu_j \end{cases}, \qquad \forall (\mu_i, \mu_j) \in M,$$

where $M = \{\mu_{\text{sba}}, \mu_{\text{vnd}}, \mu_{\text{hd}}, \mu_{\text{lr}}\}$, such that $\mu_{\text{sba}}, \mu_{\text{vnd}}, \mu_{\text{hd}}$, and μ_{lr} are, respectively, the average of the rankings obtained by SBA, VND, FO-SBA, and FO-LR in the second step. The null hypothesis (H_0) states that the average ranking of μ_i and μ_j was not significantly different among them, thus implying that the results of one of the evaluated heuristics is not significantly better than those of the other. On the other hand, the alternative hypothesis (H_1) implies that indeed the results of μ_i is significantly different than those of μ_j.

Table 6.14 presents the results of the Nemenyi's test for *Random* and *Layered* digraphs. Each cell of this table displays the p-value obtained by comparing the heuristics displayed on the top of the column and on the beginning of the row.

One can see from Table 6.14 that the results of the statistical tests were very similar for both *Random* and *Layered* instances. The statistical tests returned a p-value smaller than 0 05 for all performed comparisons. Thus, the test rejected the null hypothesis and stated that there are significant differences between all heuristic pairs, for both *Random* and *Layered* instances. Since the average ranking computed for FO-LR is smaller than those of the other heuristics, we can infer that FO-LR was the best among the evaluated heuristics for solving both instance's set.

Table 6.11. Reduction achieved by FO-SBA and FO-LR on layered digraphs

		FO-SBA				FO-LR			
$\|N\|$	L	$\|A'\|$	$\frac{\|A'\|}{\|A\|}$ (%)	t_p (s)	t_s (s)	$\|A'\|$	$\frac{\|A'\|}{\|A\|}$ (%)	t_p (s)	t_s (s)
102	2	112.3 ± 6.2	4.3	0.1 ± 0.1	0.1 ± 0.2	141.3 ± 1.5	5.4	3.4 ± 0.6	0.6 ± 0.5
	4	119.8 ± 7.7	6.2	0.2 ± 0.1	0.2 ± 0.2	183.7 ± 38.9	9.5	4.4 ± 1.8	0.8 ± 0.6
	10	118.7 ± 9.7	12.9	0.0 ± 0.0	0.1 ± 0.1	201.8 ± 55.2	21.9	5.2 ± 2.6	655.9 ± 1304.6
	20	115.4 ± 7.2	23.8	0.0 ± 0.0	0.1 ± 0.1	179.9 ± 4.4	37.1	2.9 ± 0.4	710.8 ± 135.5
152	3	178.1 ± 14.6	3.5	0.6 ± 0.2	2.4 ± 0.8	254.5 ± 4.6	5.0	19.1 ± 0.7	2.1 ± 0.3
	6	183.0 ± 11.9	5.8	0.5 ± 0.2	1.4 ± 0.5	318.8 ± 8.7	10.0	32.1 ± 2.3	53.3 ± 7.2
	15	182.3 ± 16.9	12.8	0.4 ± 0.1	0.9 ± 0.3	312.2 ± 8.6	22.0	31.0 ± 1.2	1449.0 ± 163.9
	30	173.3 ± 10.2	23.6	0.3 ± 0.1	0.7 ± 0.3	275.2 ± 6.2	37.4	14.5 ± 0.4	1196.9 ± 174.0
202	4	246.1 ± 19.3	3.2	0.7 ± 0.1	3.8 ± 1.1	383.3 ± 8.6	5.0	76.5 ± 3.1	5.9 ± 0.4
	8	246.7 ± 16.4	5.6	0.5 ± 0.2	2.3 ± 0.9	461.2 ± 14.3	10.4	174.8 ± 12.6	1860.2 ± 165.5
	20	243.1 ± 20.5	12.7	0.5 ± 0.2	1.8 ± 0.6	421.3 ± 12.6	21.9	141.2 ± 11.4	2057.3 ± 159.1
	40	232.6 ± 13.4	23.6	0.4 ± 0.2	1.1 ± 0.3	369.7 ± 8.4	37.5	82.5 ± 6.4	1199.4 ± 167.5

Table 6.12. Results for the heuristics on random digraphs

		SBA		VND		FO-SBA		FO-LR	
$\|N\|$	D	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)
100	0.25	39.8 ± 19.2	0.0 ± 0.0	2.1 ± 8.2	110.2 ± 27.9	16.5 ± 16.8	0.1 ± 0.1	0.4 ± 1.1	4.5 ± 1.0
	0.50	31.2 ± 17.8	0.2 ± 0.1	1.4 ± 5.5	199.7 ± 53.8	16.3 ± 14.5	1.2 ± 0.3	0.1 ± 0.3	8.3 ± 1.8
	0.75	29.5 ± 10.7	0.4 ± 0.2	3.1 ± 8.3	268.1 ± 71.7	13.1 ± 11.3	2.3 ± 0.3	0.0 ± 0.0	11.4 ± 3.1
	1.00	21.8 ± 21.9	0.6 ± 0.2	17.1 ± 13.7	224.7 ± 147.7	10.0 ± 6.8	2.6 ± 0.4	0.0 ± 0.0	14.9 ± 3.2
200	0.25	42.3 ± 16.7	0.7 ± 0.3	35.4 ± 12.9	420.1 ± 8.0	24.7 ± 18.5	6.7 ± 0.4	0.2 ± 0.5	67.5 ± 30.4
	0.50	39.4 ± 33.1	1.5 ± 0.4	33.9 ± 18.1	1022.0 ± 67.4	18.5 ± 20.7	13.8 ± 0.8	0.1 ± 0.3	150.1 ± 77.5
	0.75	27.2 ± 11.4	2.2 ± 0.5	23.1 ± 9.0	1555.5 ± 247.7	12.3 ± 5.6	21.5 ± 0.9	0.1 ± 0.2	214.3 ± 98.0
	1.00	29.1 ± 11.3	2.9 ± 0.5	27.7 ± 13.4	2085.0 ± 18.2	15.1 ± 14.4	25.5 ± 0.7	0.1 ± 0.2	319.1 ± 111.0
300	0.25	35.9 ± 18.7	3.0 ± 0.4	28.1 ± 10.1	2451.4 ± 329.4	15.9 ± 9.1	27.3 ± 0.7	0.1 ± 0.1	828.9 ± 495.1
	0.50	33.0 ± 27.1	4.4 ± 0.4	*	*	14.3 ± 6.7	63.1 ± 1.0	0.0 ± 0.0	1548.3 ± 614.9
	0.75	35.5 ± 18.7	6.1 ± 0.4	*	*	15.5 ± 6.9	166.6 ± 3.6	0.0 ± 0.0	2335.5 ± 656.3
	1.00	3.5 ± 1.2	7.9 ± 0.6	*	*	0.0 ± 0.0	662.0 ± 47.8	*	*

Table 6.13. Results for the heuristics on layered digraphs

		SBA		VND		FO-SBA		FO-LR	
$\|N\|$	L	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)
102	2	3.2 ± 1.6	0.1 ± 0.1	0.1 ± 0.2	11.5 ± 0.9	2.4 ± 2.9	0.3 ± 0.1	0.0 ± 0.0	4.0 ± 0.8
	4	9.0 ± 3.4	0.2 ± 0.1	0.0 ± 0.0	35.7 ± 21.0	6.2 ± 4.8	0.4 ± 0.2	0.2 ± 0.4	5.2 ± 1.8
	10	17.6 ± 5.3	0.0 ± 0.0	1.2 ± 2.1	187.8 ± 124.7	10.3 ± 8.5	0.2 ± 0.1	0.1 ± 0.2	661.1 ± 1306.5
	20	29.1 ± 11.4	0.0 ± 0.0	1.6 ± 2.7	221.5 ± 170.0	18.9 ± 1.1	0.2 ± 0.1	0.7 ± 2.4	713.7 ± 1354.7
152	3	10.6 ± 2.9	0.6 ± 0.2	0.0 ± 0.0	90.5 ± 8.7	5.6 ± 4.0	2.0 ± 0.7	0.1 ± 0.1	21.1 ± 7.7
	6	19.1 ± 6.3	0.5 ± 0.2	0.3 ± 0.5	290.1 ± 159.3	11.5 ± 6.9	1.2 ± 0.4	0.2 ± 0.2	85.4 ± 83.1
	15	22.4 ± 14.8	0.4 ± 0.1	7.6 ± 10.0	290.7 ± 117.6	11.8 ± 6.6	0.9 ± 0.3	0.1 ± 0.3	1480.0 ± 1651.0
	30	27.2 ± 16.4	0.3 ± 0.1	8.2 ± 12.5	184.5 ± 159.4	20.1 ± 12.6	0.9 ± 0.3	0.6 ± 1.9	1211.5 ± 1748.3
202	4	12.4 ± 9.8	0.7 ± 0.1	1.1 ± 2.5	327.1 ± 84.4	8.4 ± 5.6	4.5 ± 1.2	0.1 ± 0.1	82.4 ± 37.8
	8	17.8 ± 10.8	0.5 ± 0.2	12.9 ± 11.1	282.7 ± 98.5	14.3 ± 8.1	2.3 ± 1.0	0.1 ± 0.2	2035.0 ± 1758.9
	20	20.7 ± 15.1	0.5 ± 0.2	15.4 ± 14.4	313.3 ± 54.4	13.1 ± 5.5	1.5 ± 0.6	0.2 ± 0.5	2198.5 ± 1673.5
	40	27.8 ± 12.2	0.4 ± 0.2	10.3 ± 12.8	225.1 ± 159.0	20.5 ± 12.8	1.1 ± 0.3	0.6 ± 1.5	1281.9 ± 1724.6

Table 6.14. p-values obtained by the Nemenyi's test on random and layered digraphs

	Random			Layered		
	SBA	VND	FO-SBA	SBA	VND	FO-SBA
VND	0.01	-	-	0.01	-	-
FO-SBA	0.01	0.01	-	0.01	0.02	-
FO-LR	0.01	0.01	0.01	0.01	0.01	0.01

Chapter 7

Conclusions and future works

This thesis focused on the Minmax regret 0-1 Integer Linear Programming Problem under Interval Uncertainty (M-ILP). We proved that this problem is complete for the second level of the polynomial-hierarchy, being Σ_2^p-Complete, considering that the polynomial-hierarchy do not collapses to some finite level. Therefore, M-ILP is considered to be harder than the problems in the class NP.

We proposed two new heuristics for M-ILP: the Fix-and-Optimize through Scenario-based Algorithm (FO-SBA) and the Fix-and-Optimize through Linear Relaxation (FO-LR). Both heuristics were based in the Fix-and-Optimize heuristic framework of Gintner et al. [2005], which was originally proposed for a variant of a Vehicle Routing Problem. These heuristics consist of two steps. In the first step, denominated preprocessing step, a heuristic is used to fix the value of some variables in zero. In the second step, a MILP formulation is solved without the fixed variables. The heuristics differ from each other by the manner they fix variables in zero. The former uses the Scenario-based Algorithm, which was first proposed by Coco et al. [2016] for solving the Minmax regret Weighted Set Covering Problem under Interval Uncertainty, while the latter uses the linear relaxation of the MILP formulation for M-ILP.

The proposed heuristics were applied to solve two M-ILP instances: the Minmax regret Weighted Set Covering Problem under Interval Uncertainty (M-WSCP) and the Minmax regret Single-Source Shortest Path Problem under Interval Uncertainty (MSS-SPP). We proved the worst-case complexity of both M-WSCP and MSS-SPP and performed a set of computational experiments to compare the proposed heuristics with those of the literature for these problems.

For the M-WSCP, we proved that it is on the second level of the polynomial hierarchy, being Σ_2^p. However, we could not show that it was complete for this class. Regarding the heuristics, the computational experiments performed on two instance sets demonstrated that FO-LR was competitive in comparison to the other heuristics when solving the smaller evaluated instances. However, FO-LR was statistically superior than all other assessed heuristics when solving the larger evaluated instances.

For the MSS-SPP, we proved that this problem is NP-Hard. Regarding the heuristics, computational experiments performed on two classic set of instances from the literature demonstrated that FO-LR outperformed all other evaluated heuristics.

Future works may focus on demonstrating that M-WSCP is complete for the class Σ_2^p. From an algorithmic perspective, a future work can also look to improve the ILP formulation for MSS-SPP by introducing the MTZ subtour elimination constrains [Miller et al., 1960] or the lifted MTZ constraints [Akgün and Tansel, 2011]. Additionally, another interesting topic of research is to explore new manners to fix variables in the Fix-and-Optimize heuristic framework and to extend it to other problems.

Bibliography

Aissi, H., Bazgan, C., and Vanderpooten, D. (2005a). Complexity of the min–max and min–max regret assignment problems. *Operations research letters*, 33(6):634--640.

Aissi, H., Bazgan, C., and Vanderpooten, D. (2005b). Complexity of the min-max (regret) versions of cut problems. In *International Symposium on Algorithms and Computation*, pages 789--798. Springer.

Aissi, H., Bazgan, C., and Vanderpooten, D. (2009). Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427--438.

Akgün, İ. and Tansel, B. Ç. (2011). New formulations of the hop-constrained minimum spanning tree problem via miller–tucker–zemlin constraints. *European Journal of Operational Research*, 212(2):263--276.

Anand, R., Aggarwal, D., and Kumar, V. (2017). A comparative analysis of optimization solvers. *Journal of Statistics and Management Systems*, 20(4):623--635.

Aron, I. D. and Hentenryck, P. V. (2004). On the complexity of the robust spanning tree problem with interval data. *Operations Research Letters*, 32(1):36--40.

Assunção, L., Noronha, T. F., Santos, A. C., and Andrade, R. (2017). A linear programming based heuristic framework for min-max regret combinatorial optimization problems with interval costs. *Computers & Operations Research*, 81:51--66.

Assunção, L., Santos, A. C., Noronha, T. F., and Andrade, R. (2016). On the finite optimal convergence of logic-based benders' decomposition in solving 0–1 min-max regret optimization problems with interval costs. In *International Symposium on Combinatorial Optimization*, pages 1--12. Springer.

Averbakh, I. (2000). Minmax regret solutions for minimax optimization problems with uncertainty. *Operations Research Letters*, 27(2):57--65.

Averbakh, I. (2001). On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, 90(2):263--272.

Averbakh, I. (2003). Complexity of robust single facility location problems on networks with uncertain edge lengths. *Discrete Applied Mathematics*, 127(3):505--522.

Averbakh, I. (2005a). Computing and minimizing the relative regret in combinatorial optimization with interval data. *Discrete Optimization*, 2:273--287.

Averbakh, I. (2005b). The minmax relative regret median problem on networks. *INFORMS Journal on Computing*, 17(4):451--461.

Averbakh, I. and Berman, O. (2000a). Algorithms for the robust 1-center problem on a tree. *European Journal of Operational Research*, 123(2):292--302.

Averbakh, I. and Berman, O. (2000b). Minmax regret median location on a network under uncertainty. *INFORMS Journal on Computing*, 12(2):104--110.

Averbakh, I. and Berman, O. (2003). An improved algorithm for the minmax regret median problem on a tree. *Networks: An International Journal*, 41(2):97--103.

Averbakh, I. and Lebedev, V. (2004). Interval data minmax regret network optimization problems. *Discrete Applied Mathematics*, 138:289--301.

Averbakh, I. and Lebedev, V. (2005). On the complexity of minmax regret linear programming. *European Journal of Operational Research*, 160(1):227--231.

Bai, R., Xue, N., Chen, J., and Roberts, G. W. (2015). A set-covering model for a bidirectional multi-shift full truckload vehicle routing problem. *Transportation Research Part B: Methodological*, 79:134--148.

Balas, E. and Ho, A. (1980). Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study. In *Combinatorial Optimization*, pages 37--60. Springer.

Beasley, J. E. (1990). Or-library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069--1072.

Ben-Tal, A. and Nemirovski, A. (1998). Robust convex optimization. *Mathematics of operations research*, 23(4):769--805.

Ben-Tal, A. and Nemirovski, A. (1999). Robust solutions of uncertain linear programs. *Operations research letters*, 25(1):1--13.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238--252.

Bertsimas, D. and Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical programming*, 98(1):49--71.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations research*, 52(1):35--53.

Buchheim, C. and Kurtz, J. (2018). Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization*, 6(3):211--238.

Burkard, R. E. and Dollani, H. (2002). A note on the robust 1-center problem on trees. *Annals of Operations Research*, 110(1-4):69--82.

Carvalho, I. A. (2019). On the statistical evaluation of algorithmic's computational experimentation with infeasible solutions. *Information Processing Letters*, 143:24--27.

Carvalho, I. A., Noronha, T. F., Duhamel, C., and Vieira, L. F. M. (2016). A scenario based heuristic for the robust shortest path tree problem. In *VIII Conference on Manufacturing, Modelling, Management & Control*, pages 443--448.

Carvalho, I. A., Noronha, T. F., Duhamel, C., and Vieira, L. F. M. (2018). A milp-based vnd for the min-max regret shortest path tree problem with interval costs. *Electronic Notes in Discrete Mathematics*, 66:39--46.

Catanzaro, D., Labbé, M., and Salazar-Neumann, M. (2011). Reduction approaches for robust shortest path problems. *Computers & Operations Research*, 38:1610--1619.

Ceria, S., Nobili, P., and Sassano, A. (1997). Set covering problem. In Martello, S. and Maffioli, F., editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 415--428. Wiley.

Chandra, A. K. and Stockmeyer, L. J. (1976). Alternation. In *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, pages 98--108. IEEE.

Chen, B. and Lin, C.-S. (1998). Minmax-regret robust 1-median location on a tree. *Networks: An International Journal*, 31(2):93--103.

Chen, H. (2015). Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems. *Omega*, 56:25--36.

Coco, A. A. (2017). *Robust Covering Problems: Formulations, Algorithms and Application.* PhD thesis, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil.

Coco, A. A., Júnior, J. C. A., Noronha, T. F., and Santos, A. C. (2014a). An integer linear programming formulation and heuristics for the minmax relative regret robust shortest path problem. *Journal of Global Optimization*, 60(2):265–287. ISSN 0925-5001.

Coco, A. A., Júnior, J. C. A., Noronha, T. F., and Santos, A. C. (2017). Erratum to: An integer linear programming formulation and heuristics for the minmax relative regret robust shortest path problem. *Journal of Global Optimization*, 68(2):463--466.

Coco, A. A., Santos, A. C., and Noronha, T. F. (2015). Senario-based heuristics with path-relinking for the robust set covering problem. In *Proceedings of the XI Metaheuristics International Conference (MIC).*

Coco, A. A., Santos, A. C., and Noronha, T. F. (2016). Coupling scenario-based heuristics to exact methods for the robust weighted set covering problem with interval data. In *VIII Conference on Manufacturing, Modelling, Management & Control*, pages 455–460.

Coco, A. A., Solano-Charris, E. L., Santos, A. C., Prins, C., and Noronha, T. F. (2014b). Robust optimization criteria: state-of-the-art and new issues. Technical report UTT-LOSI-14001, Université de Technologie de Troyes.

Cook, W., Hartmann, M., Kannan, R., and McDiarmid, C. (1992). On integer points in polyhedra. *Combinatorica*, 12(1):27--37.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). Single-source shortest paths. In *Introduction to Algorithms*, pages 580--619. MIT Press Cambridge.

Daniels, R. L. and Kouvelis, P. (1995). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, 41(2):363--376.

Dastjerd, N. K. and Ertogral, K. (2019). A fix-and-optimize heuristic for the integrated fleet sizing and replenishment planning problem with predetermined delivery frequencies. *Computers & Industrial Engineering*, 127:778--787.

Deineko, V. G. and Woeginger, G. J. (2010). Pinpointing the complexity of the interval min–max regret knapsack problem. *Discrete Optimization*, 7(4):191--196.

Dokka, T., Goerigk, M., and Roy, R. (2019). Mixed uncertainty sets for robust combinatorial optimization. *Optimization Letters*.

Dolgui, A. and Kovalev, S. (2012). Min–max and min–max (relative) regret approaches to representatives selection problem. *4OR*, 10(2):181--192.

Dorneles, Á. P., de Araújo, O. C., and Buriol, L. S. (2014). A fix-and-optimize heuristic for the high school timetabling problem. *Computers & Operations Research*, 52:29--38.

Edmonds, J. (1962). Covers and packings in a family of sets. *Bulletin of the American Mathematical Society*, 68(5):494--499.

Fischetti, M., Salvagnin, D., and Zanette, A. (2010). A note on the selection of benders' cuts. *Mathematical Programming*, 124(1-2):175--182.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675--701.

Furini, F., Iori, M., Martello, S., and Yagiura, M. (2015). Heuristic and exact algorithms for the interval min-max regret knapsack problem. *INFORMS Journal on Computing*, 27:392–405.

Gabrel, V., Murat, C., and Wu, L. (2013). New models for the robust shortest path problem: complexity, resolution and generalization. *Annals of Operations Research*, 207:97--120.

Garcia, S. and Herrera, F. (2008). An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. *Journal of Machine Learning Research*, 9(Dec):2677--2694.

Gilbert, E. N. (1959). Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141--1144.

Gintner, V., Kliewer, N., and Suhl, L. (2005). Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. *OR Spectrum*, 27(4):507--523.

Goncalves, J. R., Carvalho, I. A., and Noronha, T. F. (2017). An experimental evaluation of the algorithm mean upper heuristic for interval data min-max regret combinatorial optimization problem. In *2017 Brazilian Conference on Intelligent Systems (BRACIS)*, pages 378--383. IEEE.

Guo, P., Cheng, W., Wang, Y., and Boysen, N. (2018). Gantry crane scheduling in intermodal rail-road container terminals. *International Journal of Production Research*, 56(16):5419--5436.

Gupta, S. K. and Rosenhead, J. (1968). Robustness in sequential investment decisions. *Management science*, 15(2):B--18.

Inuiguchi, M. and Sakawa, M. (1995). Minimax regret solution to linear programming problems with an interval objective function. *European Journal of Operational Research*, 86(3):526--536.

Kamien, M. I. and Schwartz, N. L. (2012). *Dynamic optimization: the calculus of variations and optimal control in economics and management*. Courier Corporation.

Kannan, R. (1982). Circuit-size lower bounds and non-reducibility to sparse sets. *Information and control*, 55(1-3):40--56.

Karaşan, O. E., Pinar, M. C., and Yaman, H. (2001). The robust shortest path problem with interval data. Technical report, Bilkent University, Department of Industrial Engineering.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85--103. Springer.

Kasperski, A., Makuchowski, M., and Zieliński, P. (2012). A tabu search algorithm for the minmax regret minimum spanning tree problem with interval data. *Journal of Heuristics*, 18(4):593--625.

Kasperski, A. and Zielinski, P. (2004). Minimizing maximal regret in linear assignment problems with interval data. *Instytut Matematyki PWr., Wroclaw.*

Kasperski, A. and Zieliński, P. (2006). An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97(5):177--180.

Kasperski, A. and ZielińSki, P. (2006). The robust shortest path problem in series–parallel multidigraphs with interval data. *Operations Research Letters*, 34(1):69--76.

Kasperski, A. and Zieliński, P. (2007). On the existence of an fptas for minmax regret combinatorial optimization problems with interval data. *Operations Research Letters*, 35(4):525--532.

Kasperski, A. and Zieliński, P. (2015). Complexity of the robust weighted independent set problems on interval graphs. *Optimization Letters*, 9(3):427--436.

Kasperski, A. and Zieliński, P. (2016). Robust discrete optimization under discrete and interval uncertainty: A survey. In *Robustness Analysis in Decision Aiding, Optimization, and Analytics*, pages 113--143. Springer.

Kouvelis, P. and Yu, G. (1997). *Robust discrete optimization and its applications*, volume 14 of *Nonconvex optimization and its applications*. Springer.

Lebedev, V. and Averbakh, I. (2006). Complexity of minimizing the total flow time with interval data and minmax regret criterion. *Discrete Applied Mathematics*, 154(15):2167--2177.

Li, X., Lu, Q., Dong, Y., and Tao, D. (2017). Sce: A manifold regularized set-covering method for data partitioning. *IEEE transactions on neural networks and learning systems*, 29(5):1760--1773.

Martello, S., Pisinger, D., and Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414--424.

Martinovic, J., Delorme, M., Iori, M., Scheithauer, G., and Strasdat, N. (2020). Improved flow-based formulations for the skiving stock problem. *Computers & Operations Research*, 113:104770 .

Mausser, H. E. and Laguna, M. (1998). A new mixed integer formulation for the maximum regret problem. *International Transactions in Operational Research*, 5(5):389--403.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326--329.

Minoux, M. (2009). Solving some multistage robust decision problems with huge implicitly defined scenario trees. *Algorithmic Operations Research*, 4(1):1--18.

Montemanni, R. (2006). A benders decomposition approach for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 174(3):1479--1490.

Montemanni, R., Barta, J., and Gambardella, L. (2006). Heuristic and preprocessing techniques for the robust traveling salesman problem with interval data. Technical report Technical report IDSIA-01-06, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale.

Montemanni, R., Barta, J., Mastrolilli, M., and Gambardella, L. M. (2007). The robust traveling salesman problem with interval data. *Transportation Science*, 41(3):366--381.

Montemanni, R. and Gambardella, L. M. (2004). An exact algorithm for the robust shortest path problem with interval data. *Computers & Operations Research*, 31(10):1667--1680.

Montemanni, R. and Gambardella, L. M. (2005). A branch and bound algorithm for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 161(3):771--779.

Montemanni, R., Gambardella, L. M., and Donati, A. V. (2004). A branch and bound algorithm for the robust shortest path problem with interval data. *Operations Research Letters*, 32(3):225--232.

Nemenyi, P. (1962). Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263. International Biometric Society.

Neumann, M. S. (2010). *Advances in robust combinatorial optimization and linear programming*. PhD thesis, Université Libre de Bruxelles - Faculté des Sciences.

Nguyen, U. T. and Xu, J. (2007). Multicast routing in wireless mesh networks: Minimum cost trees or shortest path trees? *IEEE Communications Magazine*, 45(11).

Nikulin, Y. (2008). Simulated annealing algorithm for the robust spanning tree problem. *Journal of Heuristics*, 14(4):391--402.

Papadimitriou, C. (1994). *Computational Complexity*. Theoretical computer science. Addison-Wesley. ISBN 9780201530827.

Pereira, J. and Averbakh, I. (2011). Exact and heuristic algorithms for the interval data robust assignment problem. *Computers & Operations Research*, 38(8):1153--1163.

Pereira, J. and Averbakh, I. (2013). The robust set covering problem with interval data. *Annals of Operations Research*, pages 1--19. .

Pérez-Galarce, F., Álvarez-Miranda, E., Candia-Véjar, A., and Toth, P. (2014). On exact solutions for the minmax regret spanning tree problem. *Computers & Operations Research*, 47:114--122.

Pérez-Galarce, F., Candia-Véjar, A., Astudillo, C., and Bardeen, M. (2018). Algorithms for the minmax regret path problem with interval data. *Information Sciences*, 462:218–241.

Poss, M. (2018). Robust combinatorial optimization with knapsack uncertainty. *Discrete Optimization*, 27:88--102.

Ross, T. J. (2016). *Fuzzy logic with engineering applications*, volume 4. Wiley Online Library.

Salazar-Neumann, M. (2007). The robust minimum spanning tree problem: Compact and convex uncertainty. *Operations research letters*, 35(1):17--22.

Schaefer, M. and Umans, C. (2002). Completeness in the polynomial-time hierarchy: A compendium. *SIGACT news*, 33(3):32--49.

Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591--611.

Shelby, Z. and Bormann, C. (2011). *6LoWPAN: The wireless embedded Internet*, volume 43 of *Wiley Series in Communications Networking & Distributed Sistems*. John Wiley & Sons.

Spall, J. C. (2005). *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65 of *Wiley Series in Discrete Mathematics and Optimization*. John Wiley & Sons.

Stockmeyer, L. J. (1976). The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1--22.

Taccari, L. (2016). Integer programming formulations for the elementary shortest path problem. *European Journal of Operational Research*, 252(1):122--130.

Turhan, A. M. and Bilgen, B. (2017). Mixed integer programming based heuristics for the patient admission scheduling problem. *Computers & Operations Research*, 80:38--49.

Yaman, H., Karaşan, O. E., and Pinar, M. Ç. (2001). The robust spanning tree problem with interval data. *Operations Research Letters*, 29(1):31--40.

Yu, G. and Yang, J. (1998). On the robust shortest path problem. *Computers & Operations Research*, 25(6):457--468.

Yu, H.-I., Lin, T.-C., and Wang, B.-F. (2008). Improved algorithms for the minmax-regret 1-center and 1-median problems. *ACM Transactions on Algorithms (TALG)*, 4(3):36.

Zhang, B. and Mouftah, H. T. (2002). A destination-driven shortest path tree algorithm. In *2002 IEEE International Conference on Communications. Conference Proceedings*, volume 4, pages 2258--2262. IEEE.