# ROBUST OPTIMIZATION FOR OSPF ROUTING

DANIEL BRASIL MAGNANI

# ROBUST OPTIMIZATION FOR OSPF ROUTING

Dissertação 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 Mestre em Ciência da Computação.

Orientador: Thiago Ferreira de Noronha

Belo Horizonte

Maio de 2014

DANIEL BRASIL MAGNANI

# ROBUST OPTIMIZATION FOR OSPF ROUTING

Dissertation 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 Master in Computer Science.

Advisor: Thiago Ferreira de Noronha

Belo Horizonte

May 2014

Brasil Magnani, Daniel

M165 r Robust Optimization for OSPF Routing / Daniel Brasil Magnani. — Belo Horizonte, 2014
xxiv, 40 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação
Orientador: Thiago Ferreira de Noronha

1. Computação - Teses. 2. Heurística. 3. Pesquisa Operacional - Teses. I. Orientador. II. Titulo.

CDU 519.6*65 (043)

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

FOLHA DE APROVAÇÃO

Robust optimization for OSPF routing

**DANIEL BRASIL MAGNANI**

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Thiago F. Noronha
Prof. Thiago Ferreira de Noronha - Orientador
Departamento de Ciência da Computação - UFMG

Andréa Cynthia Santos
Profa. Andrea Cynthia Santos
Université de Tecnologie de Troyes

[illegible]
Prof. Maurício Cardoso de Souza
Departamento de Engenharia de Produção - UFMG

[illegible]
Prof. Sebastián Alberto Urrutia
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 18 de junho de 2014.

*To my grandparents Eunice, Peter, Haroldo and Elyanne.*

# Acknowledgments

This dissertation is a result of more than 2 years of work. I thank everyone that contributed during this time.

To my advisor, Thiago, for the knowledge, patience, support and comprehension, essential to the development of the work.

To Andréa and Christophe, for the opportunity to work with them at the *Université de Technologie de Troyes*.

To the colleagues and employees of the Department of Computer Science, for making the days more enjoyable.

To my family and friends for supporting and motivating me.

To my girlfriend for staying by my side at the difficult moments.

To the Brazilian National Council for Scientific and Technological Development (CNPq) and the Coordination for the Improvement of Higher Education Personnel (CAPES) for funding this research.

*"I am he as you are he as you are me and we are all together"*
(John Lennon)

# Resumo

No protocolo OSPF, dado um conjunto de pesos para cada link, os dados são roteados através do menor caminho entre o remetende e o destinatário. O problema da atribuição de pesos OSPF (do inglês OSPF weight setting problem) consiste em definir os pesos dos links de uma rede de computadores, de tal forma que o roteamento resulte na rede menos congestionada possível. A maioria dos trabalhos da literatura assumem que uma única matriz de demandas estática está disponível. Entretanto, o tráfego em redes de computadores pode variar significantemente ao longo do tempo, e não é prático para o administrador da rede mudar manualmente os pesos dos links toda vez que uma variação significante ocorrer. Estes fatores motivaram o desenvolvimento de modelos de otimização para o problema que lidam com incerteza no tráfego. Ao invés de minimizar o congestionamento médio em relação aos vários cenários, como é o caso dos trabalhos da literatura, nós propomos um novo modelo de otimizaçao, baseado em Otimização Robusta, onde o congestionamento em cada cenário é considerado individualmente. Nós argumentamos que o usuário experimenta cada cenário individualmente. Portanto, uma solução que é boa na média pode resultar numa qualidade de serviço ruim na perspectiva do usuário. Experimentos computacionais, realizados em redes realísticas e artificiais, mostram que, comparado à abordagem que minimiza o caso médio, nossa abordagem consegue reduzir o arrependimento em 25%, enquanto aumenta o congestionamento médio em apenas 0.72%, indicando que nossa abordagem pode ser uma alternativa melhor para a atribuição de pesos OSPF.

# Abstract

On OSPF protocol, given a set of weights for each link, the data are routed through the shortest paths between the sender and the receiver. The OSPF weight setting problem consists in assigning the link weights such that the respective shortest path routing results in the least congested network. Most of the works in the literature assume that a single static demand matrix is available. However, the traffic on computer networks may significantly vary in different periods of time, and it is not practical for the network operator to manually change the weights of the links each time significant variation in traffic occurs. These factors motivated the development of optimization models for OSPF weight setting that deal with traffic uncertainties. Instead of minimizing the average congestion over all scenarios as is the case of the works in the literature, we propose a new optimization models, based on Robust Optimization, where the congestion in each scenario is considered individually. We argue that the user experiences each scenario individually. Therefore, a solution that is good on average may sometimes result in a bad quality-of-service from the user point of view. Computational experiments, performed on realistic and artificial instances, show that, compared to the approach that minimizes the average case, our approach is able to reduce the congestion regret by 25%, while increasing the average congestion by only 0.72%, indicating that our approach may be a better alternative for weight setting in OSPF networks.

# List of Figures

# List of Tables

# Contents

# Chapter 1

# Introduction

The Internet is a global network connecting routers, switches, and hubs that communicate mainly through the Internet Protocol (IP). The data is transmitted in small units called packets, that contain the message being sent, the destination address, as well as other relevant information. Every subnetwork that is under the administration of a single institution is called an autonomous system (AS), as shown on Figure 1.1. The Internet can be divided into two layers of protocols: the *intra-domain* layer, implemented by the Interior Gateway Protocols (IGPs), is responsible for managing the traffic *inside* an autonomous system, while the *inter-domain* layer deals with the traffic *between* the autonomous systems.

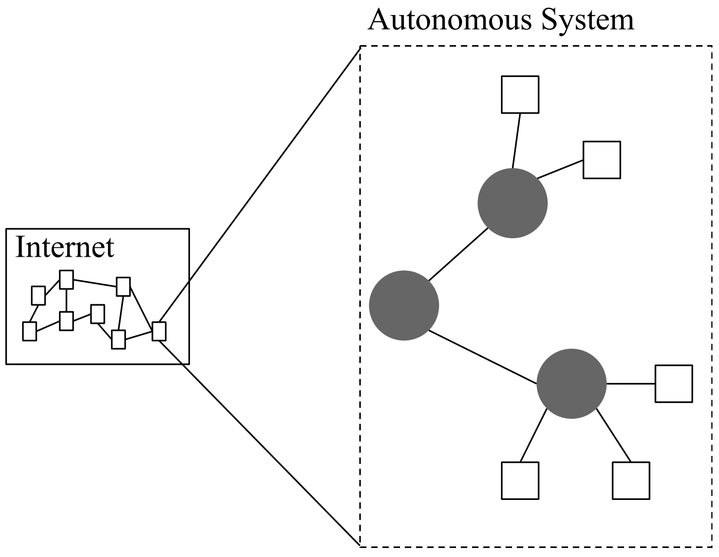


Figure 1.1: An autonomous system on the Internet

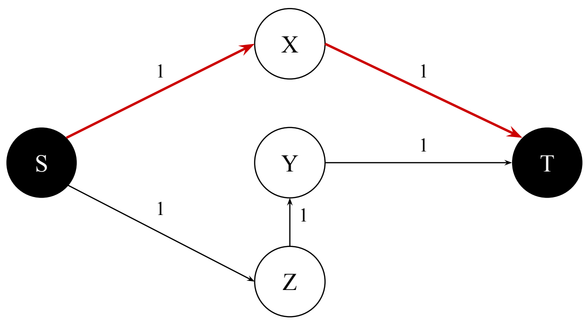
One of the responsibilities of an AS is to solve the routing problem, that consists in deciding the paths through which the network flow is going to pass. Since there are many possible routes for each demand, deciding the path of the data is crucial to avoid link overloading, in order to achieve better response times and to keep the network reliable. There are several protocols that provide intra-domain routing, such as RIP, OSPF and IS-IS. Each organization is responsible for choosing the protocol that suits best its demands. Nowadays, OSPF and IS-IS are the most common choices.

The Open Shortest Path First (OSPF) protocol is an IGP that was created by the OSPF working group of the Internet Engineering Task Force [Moy, 1998]. In OSPF routing, the network administrator assigns integer weights to each link of the network. These weights are used as lengths to calculate the shortest paths between all pairs of routers, and the data is routed through the shortest paths. In the case of multiple shortest paths, the traffic is split evenly, among all outgoing links that belong to the shortest paths. This behavior is called Equal Cost Multi-Path (ECMP) rule. Figure 1.2 shows an example of OSPF routing with the ECMP rule. In the first case ($a$), all links have the same weight, and all the 10 megabits are routed through the shortest path ($S \to X \to T$) of length 2. On the second case ($b$), there are two shortest paths of length 3: ($S \to X \to T$) and ($S \to Z \to Y \to T$). Therefore, the flow is split evenly among links $(S, X)$ and $(S, Z)$.

## 1.1 Problem Definition

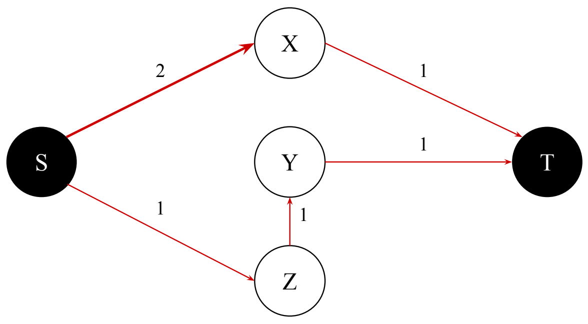
The weight setting problem (WSP) consists in assigning the link weights in order to optimize an objective function that models some network performance metric. The problem is part of a wider area, called Traffic Engineering, responsible for managing the resources of the network to satisfy the user demands. It was proven to be NP-hard in [Fortz and Thorup, 2000].

WSP was modeled by Fortz and Thorup [2000] as follows. Given a directed graph $G = (N, A)$, where $N$ is a set of nodes that represent the routers, and $A$ is a set of arcs that represent the links. The capacity of each link $a \in A$ is denoted by $c_a \in \mathbb{N}$, and $d$ is a demand matrix, where each element $d_{ij} \in \mathbb{N}$ tells how much traffic flows from $i \in N$ to $j \in N$. The decision variables $w_a \in \{1, \ldots, 65535\}$ are the weights assigned to each link $a \in A$, and the objective function aims to minimize the function $\Phi$ that models the network congestion. Given the weight of each link $a \in A$, let $l_a$ be the resulting amount of flow passing through arc $a$, and $\phi_a(l_a)$ be the monotonically increasing piecewise linear convex function that models the congestion of $a$, where



| Path | Length | Flow (Mb) |
|---|---|---|
| $(S \to X \to T)$ | 2 | 10 |
| $(S \to Z \to Y \to T)$ | 3 | 0 |

(a) In this example, all link weights are set to 1. The shortest path between $S$ and $T$ is $(S \to X \to T)$. Thus, all traffic is sent through it.



| Path | Length | Flow (Mb) |
|---|---|---|
| $(S \to X \to T)$ | 3 | 5 |
| $(S \to Z \to Y \to T)$ | 3 | 5 |

(b) In this example, the weight of link $S \to T$ is set to 2, and there are two shortest paths. Therefore, the traffic is split among them.

Figure 1.2: Example of OSPF routing, and the ECMP rule

$u_a = \frac{l_a}{c_a}$ is the utilization ratio of the link $a$. The Equation 1.1 shows the derivative $\phi'_a(l_a)$ of the function $\phi_a(l_a)$. The objective function $\Phi$ is defined as the sum of $\phi_a$ for all arcs $a \in A$, as shown on Equation 1.2. A graphic representation of $\phi_a$ is given in Figure 1.3. The more the flow of an arc is close to its capacity, the more expensive it is to add flow to it. We note that in this model the amount of flow $l_a$ of the arc $a \in A$ can be larger than its capacity $c_a$. However, in this case the congestion cost of $a$ is exponentially large.

$$\phi'_a(l_a) = \begin{cases} 1 \; for & 0 \leq \frac{l_a}{c_a} < 1/3 \\ 3 \; for & 1/3 \leq \frac{l_a}{c_a} < 2/3 \\ 10 \; for & 2/3 \leq \frac{l_a}{c_a} < 9/10 \\ 70 \; for & 9/10 \leq \frac{l_a}{c_a} < 1 \\ 500 \; for & 1 \leq \frac{l_a}{c_a} < 11/10 \\ 5000 \; for & 11/10 \leq \frac{l_a}{c_a} \end{cases} \tag{1.1}$$

$$\Phi = \sum_{a \in A} \phi_a(l_a) \tag{1.2}$$

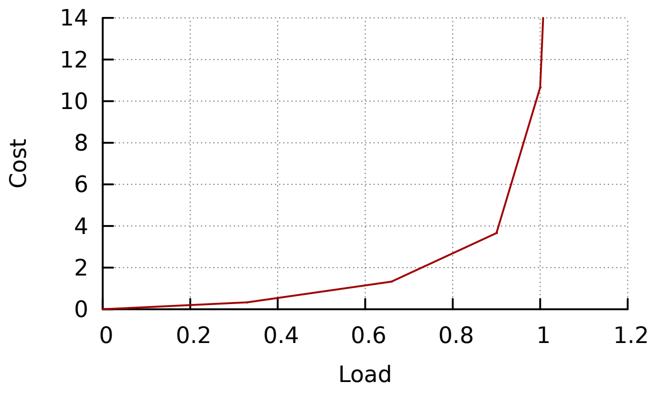


Figure 1.3: Cost function $\phi_a$ for arcs with capacity $c_a = 1$

There are several objective functions that can be used for Traffic Engineering. Balon et al. [2006] compared and evaluated nine objectives found in the literature, and concluded that, despite each objective have its own pros and cons, the objective function used by Fortz and Thorup [2000] performs well in all scenarios. Therefore, in this work we focus on the problem as modeled by Fortz and Thorup [2000], in which the function $\Phi$ is used as the objective function.

The works [Fortz and Thorup, 2000; Pióro et al., 2002; Ericsson et al., 2002; Buriol et al., 2005; Reis et al., 2011] assume that a single static demand matrix is available. However, the traffic on computer networks may significantly vary in different periods of time. Unfortunately, it is not practical for the network operator to manually change the weights of the links each time a significant variation in traffic occurs, because this might disrupt the consistency and dependability of network operations. These factors motivated the development of models for OSPF weight setting that deal with traffic uncertainties [Fortz and Thorup, 2002; Mulyana and Killat, 2005; Abrahamsson and Bjorkman, 2009; Altın et al., 2010; Altin et al., 2012].

The most prominent of these works are those of Fortz and Thorup [2002] and Altin et al. [2012]. They assume that the uncertainty in the network traffic can be approximated by a set $R$ of demand matrices, with each matrix representing a possible scenario of traffic. Given the weight of each link $a \in A$, let $\Phi_r$ be the cost of $\Phi$ for the demand matrix $r \in R$. The objective function used in these works consists in minimizing the sum of $\Phi_r$ for all demand matrices in $R$, i.e. $\Phi^S = \min \sum_{r \in R} \Phi_r$. This is equivalent to minimize the average of $\Phi_r$, for all $r \in R$, as the number of matrices in $R$ is fixed.

In this dissertation, we propose a new approach for the OSPF weight setting problem that also uses a set of demand matrices (traffic scenarios) to model the uncertainty in the network traffic. However, instead of minimizing the average value of $\Phi_r$, as is the case of [Altin et al., 2012; Fortz and Thorup, 2002], in our approach each scenario is considered individually. We argue that, the user experiences each scenario individually. Therefore, a solution that is good on average may sometimes result in a bad quality-of-service from the user point of view, during some of the scenarios.

We propose three new optimization models for weight setting in OSPF networks based on Robust Optimization. First, we propose a *minmax* model that minimizes the congestion cost of the most congested scenario. Next, as the latter is generally considered a conservative approach, we propose a *minmax regret* and a *minmax relative regret* model that minimize, for each scenario, the regret of using the given weight setting, instead of the optimal weight setting, for that scenario. As finding the cost of the optimal weight setting for one scenario is NP-Hard, our approach uses the well known lower bound to this cost proposed in [Fortz and Thorup, 2000]. As far as we know, this is the first work in the literature that uses this approach in order to deal with robust optimization problems with discrete sets of scenarios, where the classic optimization counterpart is NP-Hard.

To solve the proposed models, we extend, evaluate, and compare the best algorithms for the OSPF weight setting problem in the literature. The first algorithm is the tabu search of Fortz and Thorup [2000], and the second is the genetic algorithm of Buriol et al. [2005]. Both approaches are extended for each of the three models. Computational experiments, performed on realistic and artificial instances, show that our approach is able to reduce the congestion regret of Fortz and Thorup [2002] by 25%, while increasing the average congestion by only 0.72%, indicating that our approach may be a better alternative for weight setting in OSPF networks.

The remainder of this text is organized as follows. Chapter 2 reviews relevant works related to Traffic Engineering and Robust Optimization. Then, Chapter 3 presents the models proposed in this work and the algorithms used to solve them. Next, the computational experiments are presented in Chapter 4, and concluding remarks are drawn in the last chapter.

# Chapter 2

# Related works

Before the work of Fortz and Thorup [2000], the common approach to OSPF weight setting was to set the weight of a link to a value inversely proportional to its capacity, as recommended by the router manufacturer Cisco. Thereafter, there have been many works in the literature related to this problem. In this chapter, we review some of these papers that are related to this dissertation.

In Section 2.1, we review the works that deal with the OSPF weight setting problem and a single demand matrix. As WSP is NP-Hard, the most important methods to solve it are based on heuristics. The works [Fortz and Thorup, 2000; Fortz and Ümit, 2011] use tabu search based algorithms, while [Ericsson et al., 2002; Buriol et al., 2005; Reis et al., 2011] use genetic algorithms. Besides, a simulated annealing heuristic is proposed in [Pióro et al., 2002]. Other methods are also proposed, such as the Lagrangian relaxation of Pióro et al. [2002]; Srivastava et al. [2005] and the branch and cut of Parmar et al. [2006]. In addition, Fortz et al. [2003] and Broström and Holmberg [2006] studied a variant of WSP that seeks routes that perform well in the case of link failures.

In Section 2.2, we review works that tackled variants of the problem that aim at obtaining networks capable of dealing with uncertainty in the traffic demands. [Fortz and Thorup, 2002; Altin et al., 2012] focus on obtaining routes that minimize the average network congestion, while Mulyana and Killat [2005] try to obtain routes that minimize the link utilization ratio. Besides, Abrahamsson and Bjorkman [2009] seek routes that maximize the spare capacity on the links, in order to allow a future increasing in traffic.

In Section 2.3, we review works that deal with uncertainty in the General Routing Problem (GRP). The latter is a generalization of WSP that has no ECMP constraint. This problem can be solved in polynomial time, and the works of [Altın et al., 2010;

Altin et al., 2012; Buriol et al., 2005; Ericsson et al., 2002; Fortz and Thorup, 2000] showed that the solution to this problem is a lower bound very close to the cost of the optimal solution of WSP. Applegate and Cohen [2003, 2006] discussed the problem of finding a good route with little information about the traffic. Ben-Ameur and Kerivin [2005]; Tabatabaee et al. [2007] modeled the uncertainty in GRP with polyhedral demands, while Zhanga et al. [2005] proposed a multi-objective approach that minimizes both the average and the maximum link utilization.

## 2.1 OSPF weight setting problem

The first effort to solve WSP was made by Fortz and Thorup [2000]. They propose a tabu search heuristic called IGP-WO. First, an initial solution is generated by assigning random weights, within the range $[0, 20]$, for each link. Next, two neighborhoods are defined for the current solution $x$. The first, called *single weight change*, is defined as any solution that can be generated by changing the weight of a single link of $x$. In the second, called *evenly balanced flows*, they take a node $s$ that has flow going to target $t$, and set the weights of the arcs $(s, u) \in A$ in such a way that there is one shortest path from $s$ to $t$, passing through $u$, for each neighbor $u$ of $s$. Initially, $\delta = 20\%$ of the neighbors are evaluated at each iteration. Whenever a better solution is found on this restricted neighborhood of the current solution, $\delta$ is divided by 3, and the current solution is updated. If no improving neighbor is found, the value of $\delta$ is doubled, and the search continues from the best of the neighbors. A hash table is used to avoid cycling. This procedure also makes use of a perturbation procedure. After 300 iterations without improving the current solution, the weight of 10% of the arcs are randomly changed. The procedure stops after 5000 iterations. The results of this heuristic were improved by Fortz and Ümit [2011]. They used a different heuristic to provide the initial solution for the tabu search. The latter consists in solving the General Routing Problem for the same instance with a column generation algorithm. Then, the weight of each links is set to the value of the respective dual variable in the optimal solution of the column generation.

Pióro et al. [2002] presented a mixed-integer linear programming formulation for the variant that aims at minimizing the maximum utilization, and proposed four different methods to solve it. The first is a local search, called *weights' adjustment*. The main idea for the procedure is to increase the weight of overloaded links, and decrease the weight of underloaded links. The second method is based on *Simulated Annealing* [Johnson et al., 1989], where a neighbor is defined as any solution (weight setting) that can be generated by increasing or decreasing the weight of a single link in the current solution. The third uses *Lagrangian relaxation* [Geoffrion, 1974], and solve the dual of the GRP to obtain solutions, with the objective of maximizing the residual capacity. The fourth is a *two-phase algorithm*. On the first phase it tries to find routes for each demand, and on the second phase it tries to find a weight setting that induces these routes. Computational experiments showed that the weights' adjustment heuristic is better for small networks, while the two-phase algorithm is better for bigger networks.

Parmar et al. [2006] used the formulation proposed in [Pióro et al., 2002], and proposed heuristics to obtain feasible initial solutions, and a branch and cut algorithm to solve the problem at optimality. The latter was able to reduce the average optimality gap of CPLEX from 35% to 5% on small instances with up to 15 nodes. However, optimal solutions are obtained for only 7 out of the 18 instances tested.

Srivastava et al. [2005] tackled three variants of the problem that differ from each other by their objective functions: minimization of the overall congestion, minimization of the maximum link utilization, and a combination of both. Two scaling factors are used to prioritize one function over another. They use a Lagrangian relaxation based dual method to find solutions. The authors evaluate the quality of the solutions in terms of maximum utilization, fraction of used capacity, number of overloaded links, fraction of required extra capacity (when infeasible), and overall congestion (modeled by function $\Phi$). They conclude that the composite objective function performs well for the various measures, and can be adjusted to satisfy the objective of the network administrator.

Ericsson et al. [2002] used a Biased Random Key Genetic Algorithm (BRKGA) to solve the OSPF weight setting problem, named GAOSPF. An individual is represented by a vector of weights $w = \{w_a \in [1, w_{max}] : a \in A\}$, where $w_{max}$ is set to 20. The initial population is generated by choosing a random weight for each arc $a \in A$. The fitness function used is $\Phi$. According to the fitness of all individuals, they are separated in sets $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$. The best solutions are kept in $\mathcal{A}$, the worst in $\mathcal{C}$, and the rest in $\mathcal{B}$. The next generation is created as follows. First, all individuals in $\mathcal{A}$ are copied to the next generation. Next, each individual in set $\mathcal{B}$ is replaced by the result of a crossover between two individuals: one from set $\mathcal{A}$ and the other from $\mathcal{B} \cup \mathcal{C}$. Then the set $\mathcal{C}$ is replaced by new randomly generated solutions. With this algorithm, they found solutions as good as those of [Fortz and Thorup, 2000] in most of the topologies, but were unable to find good solutions for random graphs. They argued that the maximum running time set was not long enough to GAOSPF converge to good solutions for these instances.

Buriol et al. [2005] implemented a hybrid genetic algorithm (HGA) to improve GAOSPF. They perform a local search on each individual after the crossover operation. The local search selects the five most congested arcs, and increase their weights, by one unit at a time, to the limit of $\frac{w_{max}-w_a}{4}$. They were able to obtain better results in comparison to GAOSPF, and achieved solutions competitive with those of the tabu search of [Fortz and Thorup, 2000].

## 2.2 Uncertain demands

Fortz and Thorup [2002] proposed an optimization model for OSPF weight setting, where the congestion of the network was optimized for a set of demand matrices, representing different scenarios of traffic. They aim at finding a single arrangement of weights that generate routes that are good for the demand matrices on average. Given a set of demand matrices (scenarios of traffic) $R$, let $\Phi_r$ be the cost of the solution for a demand matrix $r \in R$. The objective function is to minimize the sum of $\Phi_r$ for all demand matrices in $R$, as shown on Equation 2.1.This is equivalent to minimizing the average of $\Phi_r$, for all $r \in R$, as the number of matrices in $R$ is fixed. They proposed an extension of the IGO-WO heuristic to solve this variant of WSP. Here, we refer to this heuristic as IGP-AVE. The latter is equal to IGP-WO, except for the objective function, that minimizes the sum of the network congestion.

$$\min \sum_{r \in R} \Phi_r \tag{2.1}$$

Mulyana and Killat [2005] considered a model based on outbound traffic constraints, where, instead of using a demand matrix, they calculate the load based on the maximum amount of traffic originating from each node. The objective is to minimize the maximum utilization. They solve the model with an heuristic approach based on simulated annealing.

Abrahamsson and Bjorkman [2009] also tackled the uncertainty on the demands, by defining a cost function that increases faster when the link flow exceeds 80% of its capacity. They argue that the 20% spare capacity may be enough to handle future variations in the traffic. The authors apply a tabu search based on the one presented on [Fortz and Thorup, 2000].

Altın et al. [2010] used a different approach to tackle the uncertainty on OSPF routing. The variation in the demands is modeled by a set of linear inequalities, with a lower and an upper bound for each demand, forming a polyhedron $\mathfrak{D}$, that contains each demand matrix $r$. Given a solution $x$, they propose a metric, called *performance ratio*, that consists of the ratio between (i) the maximum utilization of $x$ on the worst scenario of traffic and (ii) the least maximum utilization for that scenario. The objective is to find the weight setting that has the minimum performance ratio. They propose a *flow* formulation and a *tree* formulation, and solve them using a branch and price algorithm. Computational experiments showed that the algorithm does not find optimal solutions within 2 hours of running time for 25 out of 44 instances for the *tree* formulation, and for 27 out of 44 instances for the *flow* formulation. Besides, the performance gets worse as the networks get larger.

In [Altin et al., 2012], the authors proposed an heuristic for the OSPF weight setting problem with polyhedral demands. Differently from [Altın et al., 2010], the objective function is the same as [Fortz and Thorup, 2002], that is, to minimize the sum of $\Phi_r$ for all demand matrices in $\mathfrak{D}$. As the number of demand matrices in $\mathfrak{D}$ grows exponentially with the number of nodes, they propose an iterative two-step heuristic to solve this problem. It starts with a set $\tilde{\mathrm{D}}$ with a single demand matrix from $\mathfrak{D}$. Next, at each iteration, the procedure samples a new demand matrix from $\mathfrak{D}$, and then runs the IGP-WO heuristic of [Fortz and Thorup, 2002] to optimize $\tilde{\mathrm{D}}$. The procedure stops after 50 iterations. The routes given by the heuristic are compared to routes obtained for the general routing problem with uncertain demands. The authors conclude that it is possible to achieve routes on OSPF comparable to unconstrained routing.

## 2.3 General routing problem

The general routing problem is a generalization of the OSPF weight setting problem that has no ECMP constraint, i.e. the traffic does not need to be split evenly among outgoing links within shortest paths with the same length. As this problem is similar to WSP, we review below the works that deal with uncertainty for GRP.

Balon et al. [2006] compared nine versions of GRP that differ from each other by the objective functions. The *Fortz* objective [Fortz and Thorup, 2000] minimizes the convex piecewise linear function $\Phi$. The *MIRA* objective maximizes the max flow on the network. The idea is to obtain a network that has a better chance of sending more traffic without exceeding the capacity of the links. The *Blanchy* objective function is

$$\sum_{a \in A} (u_a - u_{mean})^2 + \beta \sum_{a \in A} (u_a)^2,$$

where $u_{mean}$ is the average link utilization of the network, and $\beta$ is a parameter that allows to give more importance to one or another part of the function. The former tries to provide load balancing, and the latter to reduce the network cost by decreasing the size of the shortest paths. The *MeanDelay* objective minimizes

$$\sum_{a \in A} \frac{1}{c_a - l_a},$$

and the weighted version, *WMeanDelay*, minimizes

$$\sum_{a \in A} \frac{l_a}{c_a - l_a}.$$

Both functions aim at reducing the overall load on the network. The *InvCap* objective minimizes the total utilization, i.e., $\sum_{a \in A} u_a$. The $u_{max}$ objective minimizes the maximum utilization. The *MinHop* objective minimizes the total load, i.e., $\sum_{a \in A} l_a$. Finally, the objective *Degrande* is a weighted sum of $u_{max}$ and *InvCap*. The authors solve the problem to optimality for each objective function, using instances with real and artificial networks. The objectives are compared for 6 metrics. The computation results showed that *Delay*, *Degrande* and *Fortz* are the best objectives to satisfy the requirements of traffic engineering. Among these, the most used in the literature is *Fortz*. Therefore, we focus our work on this objective function. The GRP is formulated as a Linear Programming problem as follows [Fortz and Thorup, 2000].

$$\min \Phi = \sum_{a \in A} \phi_a(l_a) \tag{2.2}$$

subject to

$$\sum_{x:(x,y)\in A} f^{(s,t)}_{(x,y)} - \sum_{z:(y,z)\in A} f^{(s,t)}_{(y,z)} = \begin{cases} -d_{s,t} & if\ y = s, \\ d_{s,t} & if\ y = t, \\ 0 & otherwise \end{cases} \qquad x, y, s, t \in N \tag{2.3}$$

$$l_a = \sum_{(s,t)\in NxN} f^{(s,t)}_a \qquad a \in A \tag{2.4}$$

$$\phi_a(l_a) \geq l_a \qquad a \in A \tag{2.5}$$

$$\phi_a(l_a) \geq 3l_a - \frac{2}{3}c_a, \qquad a \in A \tag{2.6}$$

$$\phi_a(l_a) \geq 10l_a - \frac{16}{3}c_a, \qquad a \in A \tag{2.7}$$

$$\phi_a(l_a) \geq 70l_a - \frac{178}{3}c_a, \qquad a \in A \tag{2.8}$$

$$\phi_a(l_a) \geq 500l_a - \frac{1468}{3}c_a, \qquad a \in A \tag{2.9}$$

$$\phi_a(l_a) \geq 5000l_a - \frac{16318}{3}c_a, \qquad a \in A \tag{2.10}$$

$$f^{(s,t)}_a \geq 0 \qquad a \in A; s, t \in N \tag{2.11}$$

The objective function (2.2) minimizes the total congestion, calculated as the sum of the congestion of each link of the network. Constraints (2.3) are the flow conservation constraints, that guarantee that the demands for each pair $(s,t)$ are satisfied. The load on the links is calculated by constraints (2.4). Constraints (2.5) to (2.10) define the cost function $\phi_a(l_a)$. The optimal solution of this formulation is used as a lower bound to OSPF routing in [Fortz and Thorup, 2000; Ericsson et al., 2002; Buriol et al., 2005; Reis et al., 2011; Fortz and Ümit, 2011].

There are many variants of the general routing problem that deal with uncertain demands [Applegate and Cohen, 2003, 2006; Tabatabaee et al., 2007; Ben-Ameur and Kerivin, 2005; Zhanga et al., 2005; Belotti and Mustafa, 2008]. The most important are reviewed below.

Applegate and Cohen [2003, 2006] studied the problem with little information on the demands. They assume that the traffic is proportional to the link capacity, and derive a traffic matrix. From that traffic matrix, they create others, with traffic varying within an arbitrary range. The idea is that the routing must be good for all traffic matrices inside that range. The objective, which is also used in [Altın et al., 2010], is to minimize the maximum performance ratio over all traffic matrices.

Ben-Ameur and Kerivin [2005] modeled the uncertainties on the demands as a polyhedron, as in [Altin et al., 2012]. The objective is to minimize the congestion on the network, considering the congestion as the sum of the flow in all arcs multiplied by the routing cost of the arc, which is the Euclidean distance between end nodes of the arc. The authors solve the model with a linear programming solver.

Zhanga et al. [2005] applied a multi-objective framework in order to balance the average case and the worst case scenarios. They consider a set of traffic matrices, representing the peak demands of six consecutive hours. They use two objectives. The first is a weighted sum of the average *link* cost ($P^D$) and the maximum *link* cost ($F^D$), i.e., $(1-\beta)P^D + \beta F^D$, where the cost of a link is denoted by $\frac{l_a}{c_a - l_a}$. The second is a weighted sum of the average *network* cost ($P^A$) and the maximum *network* cost ($F^D$), i.e., $(1-\beta)P^A + \beta F^A$, where the network cost is the sum of the link costs. The authors tackled the general routing problem and the OSPF weight setting problem, and solve them with linear programming and local search, respectively. For the general routing problem, they were able to achieve up to 15% of improvement on the worst case, with a reduction of only 3% on the average case. No improvement was observed for the OSPF weight setting problem.

Tabatabaee et al. [2007] also used a polyhedral representation of the demands. They use two kinds of constraints to form the polyhedron, named *Pipe* and *Hose* model constraints. On the Pipe model, each demand is represented by an upper bound. On the Hose model, there is an upper bound for the total amount of traffic emanating from each node, and another upper bound for the traffic received by each node. The objective is to minimize the maximum utilization for all traffic matrices inside the polyhedron. The authors formulate a linear programming problem, and solve it with a column generation algorithm. They test the Pipe and Hose model constraints, and also test the possibility of link failures on the network.

The Table 2.1 presents a summary of the related works described above. The first column shows the reference to the work. The second column displays the objective function of the optimization model used. The last column shows the method used to solve the problem.

Further details about intra-domain routing, shortest path routing protocols, and the general routing problem can be found in the surveys [Pióro and Medhi, 2004; Bley et al., 2010; Fortz, 2011; Altın et al., 2009].

| Work | Objective (minimize) | Method |
|---|---|---|
| **OSPF** | | |
| Fortz and Thorup [2000] | congestion | Local search |
| Pióro et al. [2002] | overloaded links<br>average link overload<br>exceeded capacity<br>congestion | Local search<br>Lagrangian relaxation<br>Simulated annealing<br>Two-phase algorithm |
| Ericsson et al. [2002] | congestion | Genetic algorithm |
| Buriol et al. [2005] | congestion | Genetic algorithm |
| Srivastava et al. [2005] | congestion<br>maximum utilization | Lagrangian relaxation |
| Parmar et al. [2006] | maximum utilization | Branch and cut |
| Fortz [2011] | congestion | Local search |
| **OSPF + Link failure** | | |
| Fortz et al. [2003] | congestion | Local search |
| Broström and Holmberg [2006] | disturbance after failure | Column generation |
| **OSPF + Uncertainty** | | |
| Fortz and Thorup [2002] | congestion | Local search |
| Mulyana and Killat [2005] | maximum utilization | Simulated annealing |
| Abrahamsson and Bjorkman [2009] | maximum utilization | Local search |
| Altın et al. [2010] | performance ratio | Branch and price |
| Altin et al. [2012] | congestion | Local search |
| **GRP + Uncertainty** | | |
| Applegate and Cohen [2003, 2006] | maximum utilization<br>performance ratio | Linear programming |
| Ben-Ameur and Kerivin [2005] | congestion | Linear programming |
| Zhanga et al. [2005] | maximum utilization<br>congestion | Linear programming<br>Local search |
| Tabatabaee et al. [2007] | maximum utilization | Linear programming |

Table 2.1: Summary of the related works.

# Chapter 3

# Robust optimization approach

In this chapter, we propose three robust optimization models for the OSPF weight setting problem. Here, the term Robust Optimization (RO) refers to the framework discussed in [Kouvelis and Yu, 1997], and must not be confused with robustness to link failures, and polyhedral demand uncertainty.

There are three common approaches to deal with uncertainty on decision making. The first is to estimate the data, and assume that a given scenario is expected in the future. Thereafter, the problem can be solved as a regular deterministic problem. Naturally, this method has the disadvantage of having to forecast the future, which, depending on the problem, can be challenging. Also, it is only suitable when the decision maker is interested only in the most likely scenario, and not in the whole set of possible scenarios.

Another approach, is to use Stochastic Programming, where the uncertainty is modeled by a set of random variables with a known probability distribution. The model is solved based on the expected outcome of the variables. Differently from the previous, this method recognizes the possibility of multiple scenarios in the future. However, it also relies on estimating the data, as the probabilities have to be known in advance.

Robust Optimization [Kouvelis and Yu, 1997] is a way to deal with uncertainty in decision making, where the variability of the data is represented by deterministic values. It is specially useful for when the decision maker is interested on 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 scenarios.

The OSPF weight setting problem fits into both categories. The weights cannot be changed frequently, as it could disrupt the consistency and dependability of the network. Furthermore, it may not be interesting for the network to have a low average congestion, with a high congestion in some specific scenarios, since the user experiences each scenario individually.

One of the first works on robust optimization was [Soyster, 1973], but only in the late 90s that the term robust optimization was consolidated by the works of [Ben-Tal and Nemirovski, 1998, 1999; El Ghaoui and Lebret, 1997; El Ghaoui et al., 1998; Kouvelis and Yu, 1997]. Many classic problems have robust versions, such as the shortest path problem [Coco et al., 2014], the minimum spanning tree problem, knapsack problem, resource allocation problem and the assignment problem [Kasperski et al., 2005]. Recent works have shown that robust optimization has been successfully used in many optimization problems [Gabrel et al., 2013]. We refer to the book [Kouvelis and Yu, 1997] for an overview on robust optimization and related problems.

The robust optimization framework presented on [Kouvelis and Yu, 1997] defines three critical steps to build a robust model. The first is the structuring the uncertain data. The most common approaches use a discrete set of scenarios, each one with a single value for each parameter, or an interval of values for each parameter, which leads to an infinite number of scenarios. The second step is to choose the appropriate robust criterion. [Kouvelis and Yu, 1997] highlight the *minmax* (or absolute robust), *minmax regret* (or robust deviation) and *minmax relative regret* (or relative robustness). The third and last step is to join the previous steps in a complete robust model. Given the robust model, a solution is said to be robust if it has the smallest value for the robust criterion, among all feasible solutions. A robust optimization problem consists in finding a robust solution for a given robust model.

In this work, we focus on robust optimization models where the uncertainty is modeled by a set of discrete scenarios. The scenarios can be retrieved by capturing snapshots of the network traffic in different moments of time. The optimization criteria for robust optimization determine how conservative is the robust model [Kouvelis and Yu, 1997]. Our models differ from each other by the robust optimization criteria used in the objective function. We use three criteria on our models: *minmax*, *minmax regret* and *minmax relative regret*.

Let $R$ be the set of scenarios for a given problem, and $X$ be the set of feasible solutions for this problem. Let also $\overline{x}_r$ be the cost of solution $x$ for the scenario $r$. The *minmax* criterion is defined as

$$\min_{x \in X} \max_{r \in R} \{\overline{x}_r\}, \tag{3.1}$$

i.e., the robust solution is the one that minimizes the maximum value of $\overline{x}_r$ over all scenarios.

Given $R$, $X$, and $\overline{x}_r$ as defined above, the *regret* of a solution $x \in X$ for a scenario $r \in R$ is defined as the difference between $\overline{x}_r$ and the cost $\overline{x}^*_r$ of the optimal solution $x^*_r$ for the scenario $r$, i.e. the regret of using $x$ instead of $x^*_r$ if scenario $r$ occurs. The *minmax regret* criterion is defined as

$$\min_{x \in X} \max_{r \in R} \{\overline{x}_r - \overline{x}^*_r\}, \tag{3.2}$$

i.e., the robust solution is the one that minimizes the maximum regret over all scenarios.

Analogously, the *relative regret* of a solution $x \in X$ for a scenario $r \in R$ is the regret of using $x$ instead of $x^*_r$ relative to $\overline{x}_r$, i.e. $(\overline{x}_r - \overline{x}^*_r)/\overline{x}^*_r$. The *minmax relative regret* criterion is

$$\min_{x \in X} \max_{r \in R} \{\frac{\overline{x}_r - \overline{x}^*_r}{\overline{x}^*_r}\}. \tag{3.3}$$

The *relative regret* may be a better metric than *regret*, because in the former the regret is normalized by the value of $x^*_r$, which can be very different for each scenario. For instance, let two solutions $x'$ and $x''$, as well as two scenarios $r'$ and $r''$ such that $\overline{x}'_{r'} = 11$, $\overline{x}^*_{r'} = 1$, $\overline{x}''_{r''} = 100$, and $\overline{x}^*_{r''} = 90$. The *regret* of $x'$ in $r'$ $(11 - 1 = 10)$ is the same as that of $x''$ in $r''$ $(100 - 90 = 10)$. However, one can see that the cost of $x'$ in $r'$ is tenfold that of $x^*_{r'}$, while the cost of $x''$ in $r''$ is only 11% larger than that of $x^*_{r''}$.

## 3.1 Robust optimization models

The robust models for OSPF are defined as follows. As in [Fortz and Thorup, 2002], we are given a directed graph $G = (N, A)$, where $N$ is a set of nodes that represent the routers, and $A$ is a set of arcs that represent the links. The capacity of each link $a \in A$ is denoted by $c_a \in \mathbb{N}$, and $R$ is a set of demand matrices, where each element $d^r_{ij} \geq 0$ tells how much traffic flow is sent from $i \in N$ to $j \in N$ in the demand matrix (scenario) $r \in R$. The decision variables $w_a \in \{1, \ldots, 65535\}$ are the weights assigned to each link $a \in A$.

The *minmax OSPF* weight setting problem aims at finding a set of weights that minimizes the congestion in the most congested scenario. The network congestion is modeled with the cost function $\Phi$ (Equation 1.1). Let $\Phi_r(x)$ be the cost of a solution $x$ for the scenario $r$, the objective is defined as

$$\min_{x \in X} \max_{r \in R} \{\Phi_r(x)\}, \tag{3.4}$$

i.e., to find the solution that minimizes the maximum congestion in the network over all scenarios. Besides being possibly too conservative, this model might result in solutions whose weights are over tunned to the scenario where the traffic in the network is the largest, while the congestion in the other scenarios might be much larger than the minimum possible congestion for that scenario. To overcome these drawbacks and to obtain a set of weights that is good for all scenarios simultaneously, we propose two other models for the OSPF weight setting problem.

The *minmax regret OSPF* weight setting problem aims at finding a set of weights $x$ that minimizes the maximum difference in the network congestion obtained by using $x$ instead of using the best set of weights for each scenario. As finding the best set of weights for a single demand matrix is NP-Hard, we used a lower bound to this value, proposed in [Fortz and Thorup, 2000]. This lower bound is obtained by relaxing the constraint imposing that the traffic must be split equally between outgoing links of the shortest paths. The resulting problem is the general routing problem, which can be formulated as a linear programming problem and solved in polynomial time, as seen on Section 2.3. The works of [Altın et al., 2010; Altin et al., 2012; Buriol et al., 2005; Ericsson et al., 2002; Fortz and Thorup, 2000] showed that this lower bound is very close to the cost of the optimal solution for the single matrix problem. Given $\Phi_r(x)$ as defined above, the objective function is

$$\min_{x \in X} \max_{r \in R} \{\Phi_r(x) - lb_r\}, \tag{3.5}$$

where $lb_r$ is the lower bound to the minimum congestion for the scenario $r$. Analogously, the *minmax relative regret OSPF*, evaluates the relative deviation between the solution and the best solution possible for each scenario. The objective function is

$$\min_{x \in X} \max_{r \in R} \{\frac{\Phi_r(x) - lb_r}{lb_r}\}. \tag{3.6}$$

## 3.2 Heuristics for the robust models

To solve the three optimization problems defined above, we extended two important heuristics used in the literature. The first is the tabu search proposed in [Fortz and Thorup, 2000] and the second is the hybrid genetic algorithm of [Buriol et al., 2005]. We chose them because both are frequently used in the literature [Fortz and Thorup, 2000, 2002; Fortz et al., 2003; Ericsson et al., 2002; Buriol et al., 2005; Reis et al., 2011; Altin et al., 2012], and give near-optimal solutions for the OSPF weight setting problem.

### 3.2.1 Tabu search

Algorithm 1 shows the pseudo-code of the tabu search *IGP-RO*. It starts with a random solution, where each link is assigned a weight in the range $[1, 20]$ (Line 1). The neighborhood is any solution achievable by changing the weight of one link. The neighbors are chosen by picking a random arc, and then a random weight, different from the current one. At each iteration of the search, a percentage $\delta$ of the neighborhood is searched (Lines 4 to 7). If an improving solution is found, $\delta$ is divided by three. Otherwise, $\delta$ is doubled (Lines 8 to 13). The initial value of $\delta$ is 10% (Line 2), the minimum is 1%, and the maximum 40%. If a solution is not improved after 10 iterations, a perturbation occurs (Lines 14 and 15). The perturbation consists in adding a random integer between $[-2, 2]$ to the weight of 10% of the arcs . The procedure stops after the stopping condition is met, or when the current solution has the same cost of the relaxed solution. To avoid evaluating the same solution more than once, and to avoid cycling between solutions, a hash table is used to store all the solutions that have been evaluated.

**Algorithm 1** : Tabu search

```
1: Initialization: find an initial solution $x$ by assigning random weights to each link
2: $\delta = 10\%$
3: Search: while stopping criterion is not met:
4:    while $\delta$ of the neighborhood was not covered
5:        sample neighbor
6:        if neighbor was not evaluated:
7:            evaluate neighbor
8:    if better solution was found:
9:        move to better solution
10:       $\delta = \delta/3$
11:   else:
12:       move to solution with same cost
13:       $\delta = \delta \cdot 2$
14:       if solution was not improved for 10 iterations:
15:           perturb solution
16: End: return best solution found
```

### 3.2.2 Genetic Algorithm

The genetic algorithm *BRKGA-RO* starts with a population of 50 individuals (solutions), randomly built. At each generation (iteration of the algorithm), the individuals are separated into three groups, based on their fitness. The less the cost, the better the fitness is. Figure 3.1 shows how the evolution takes place in the algorithm. The 20% fittest individuals are put on group $\mathcal{A}$. The 10% worst are put on group $\mathcal{C}$, and the other 70% on group $\mathcal{B}$. The individuals of the group $\mathcal{A}$ are maintained. The individuals of group $\mathcal{C}$ are discarded, and new random individuals are put in their places. The individuals of group $\mathcal{B}$ are enrolled on a process called crossover, where new individuals are created by mixing the individuals of group $\mathcal{A}$ and $\mathcal{B} \cup \mathcal{C}$. An individual from $\mathcal{A}$ and an individual from $\mathcal{B} \cup \mathcal{C}$ are selected randomly to be the parents. The weights of each link of the new individual are chosen with a 70% probability from parent $\mathcal{A}$ and 30% from parent $\mathcal{B} \cup \mathcal{C}$. The algorithm stops after reaching a time limit.

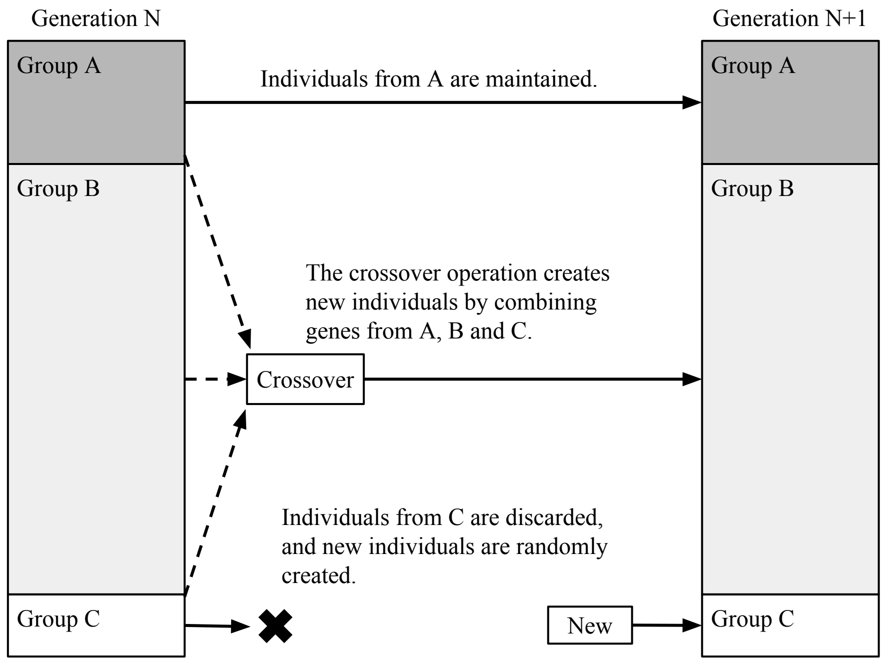

Generation N
Generation N+1
Group A
Individuals from A are maintained.
Group A
Group B
Group B
The crossover operation creates new individuals by combining genes from A, B and C.
Crossover
Individuals from C are discarded, and new individuals are randomly created.
Group C
New
Group C

Figure 3.1: Evolution of the population.

As [Buriol et al., 2005], we apply a local search on each individual. This local search consists in sorting the links in terms of link utilization, selecting the 5 most utilized links, and raising their weights, one by one, from $w_a$ to $\lceil w_a + \frac{w_{max} - w_a}{4} \rceil$. For our algorithm, we define the *most utilized links* as the links in which the utilization is the highest, taken all scenarios together.

### 3.2.3 Implementation Issues

The execution time bottleneck of both heuristics is the calculation of the cost of a given solution. To obtain the cost, firstly, we must know the shortest paths between all pair of nodes on the graph. The shortest paths can be obtained by reversing the direction of the links and running Dijkstra's algorithm for each target node. Then we must calculate the flow on the graph. This is done by computing, for each demand, how much flow will pass on each arc. Then, the value of $\Phi$ is calculated based on the loads of the arcs. These steps are repeated for each scenario.

To reduce this bottleneck, we use the algorithms and data structures of [Buriol et al., 2008]. With them, at each iteration, we only need to recalculate the shortest paths and flows for the nodes that had their shortest paths changed.

For a given solution, all shortest paths between each pair of nodes, and the flow passing on each arc are stored. The first step is finding the set $Q$ of nodes that had their shortest paths affected by changing the weight $w_a$ of an arc $a$. The set is stored on a Heap structure, where the key is the distance to the target. This way, we can retrieve the most distant node of the set in constant time.

There are four situations in which $Q$ is empty, and we can stop the procedure on the beginning. If $w_a$ is increasing, and is not on the shortest path. If $a$ is increasing and is in the shortest path, but there is an alternative path to the target, we just need to remove the path that contains $a$ from the list of shortest paths. If $w_a$ is decreasing, and the minimum distance between the tail node of $a$ and the target does not change. If $w_a$ is decreasing and $a$ enters the shortest path, we just need to insert the path that contains $a$ in the list of shortest paths.

If none of the above situations occur, we insert all nodes that had their shortest paths affected by the weight change in $Q$, and then run Dijkstra's algorithm only for this subset of nodes.

With the shortest paths, the next step is to calculate the loads on the arcs. We already have the subset $Q$ of affected nodes, therefore we only need to recalculate the loads for the arcs that connect these nodes. The others will remain unaffected. This only works if the calculus starts from the most distant node to the target, since the load on an arc depends on the load arriving from the previous nodes. As $Q$ is a Heap, finding the most distant node is done in constant time.

The neighborhoods of our local searches change only one arc at a time, thus the number of nodes that have the shortest paths modified is small, effectively improving the performance of the algorithms.

# Chapter 4

# Computational experiments

In this chapter we present the computational experiments used to test the models and algorithms presented in the previous sections. In Section 4.1 we describe the test environment, the set of test instances and the experiments. Then, in Section 4.2 we show the results and analyze them.

We define an approach as a combination of an optimization model and the algorithm used to solve the respective optimization problem. Five approaches are evaluated in the computational experiments. The first approach, called IGP-WO, is based on the tabu search of [Fortz and Thorup, 2000], and the second approach, BRKGA, is based on the genetic algorithm of [Buriol et al., 2005]. As these approaches work with a single demand matrix model, they are run for the so called *peak matrix*, as suggested by Fortz and Thorup [2002]. The peak matrix contains the maximum value of traffic for each demand. Fortz and Thorup [2002] argue that if a weight setting is good for this matrix, it has a good chance of being good for all scenarios. The third approach, referred here as IGP-AVE, is based on the model of [Fortz and Thorup, 2002], that optimizes the sum of the congestion of multiple demand matrices using a tabu search. This is the same as optimizing the average congestion, as the number of demand matrices is fixed. The last two approaches, IGP-RO and BRKGA-RO, are the extensions proposed in Section 3.2 that optimize each of the three robust criteria described in the previous chapter.

Four hypotheses are investigated in this chapter. The first is that the approaches that optimize the peak matrix (IGP-WO and BRKGA) perform much worse than IGP-AVE for the robust models. The second is that IGP-RO and BRKGA-RO can perform better than IGP-AVE for the robust models, since solutions that have a good performance on average can have a bad performance on a specific scenario. The third hypothesis is that the performance of IGP-RO and BRKGA-RO over IGP-AVE increases as the degree of variability on the traffic increases. The fourth hypothesis is that IGP-RO and BRKGA-RO can also provide good results for the average congestion, since optimizing the worst case scenario may also optimize the average case. To evaluate these hypotheses we execute two experiments. The first compares the approaches that use a single demand matrix (IGP-WO and BRKGA) with IGP-AVE, that considers multiple demand matrices. The second experiment compares IGP-AVE, that optimizes the average case, with IGP-RO and BRKGA-RO, that optimize the robust objectives.

## 4.1 Environment

The experiments were carried out on a single core of a 2.4 GHz Intel Xeon, with 32 GB of RAM memory, running GNU/Linux operating system. The heuristics IGP-WO, BRKGA, IGP-AVE, IGP-RO and BRKGA-RO were implemented from scratch in C++ and compiled with GNU/GCC version 4.6.3. The random numbers used in all heuristics were generated using the Mersenne Twister generator [Matsumoto and Nishimura, 1998]. CPLEX version 12.5 was used to calculate the lower bound to the optimal solution of each scenario.

We use the same parameters of [Fortz and Thorup, 2000] for the IGP-WO, IGP-AVE and IGP-RO heuristics. The initial value of $\delta$ is set to 10%, $w_{max}$ is set to 20, and the stop condition is 10 minutes of running time. For the BRKGA and BRKGA-RO heuristics, the population size is 50, partitioned into 20% for $\mathcal{A}$, 10% for $\mathcal{B}$ and the rest for $\mathcal{C}$. The probability of one gene be inherited from $\mathcal{A}$ is 70%. The value of $w_{max}$ is set to 20, and the stop condition is 10 minutes of running time, as is the case of IGP-WO, IGP-AVE and IGP-RO.

Two sets of instances were used. The first set was downloaded from SNDLib [Orlowski et al., 2010]. There are 13 instances based on 4 realistic network topologies, namely `abilene, geant, nobel` and `germany`. Each instance has 24 scenarios, one for the traffic of each hour of the day. There are 5 instances respective to the `abilene` network and 6 respective to the `geant` network, because more than one day of traffic was available for these networks. However, there is only one instance respective to the `nobel` network and another respective to the `germany` network, because only one day of traffic was available for these networks.

The second set of instances, based on artificial networks, was generated using the GT-ITM generator of [Zegura et al., 1996], also used in [Fortz and Thorup, 2000; Ericsson et al., 2002; Buriol et al., 2005]. A total of 10 instances, named `hier`, were generated using a 2-level hierarchical topology, because it is the topology that best represents the networks of ISPs [Fortz and Thorup, 2000]. There are two kinds of arcs in this topology. Local arcs, that have their capacity set to 200, and long distance arcs, that have their capacity set to 1000. We generated the demands as in [Fortz and Thorup, 2000], which assigns more traffic between closer nodes. For each node, two random numbers $X_i, Y_j \in [0, 1]$ are generated. Further, a random number $Z_{ij} \in [0, 1]$ is generated for each demand. Then, the demand $d_{ij}$ between nodes $i$ and $j$ is set to

$$d_{ij} = \alpha \cdot X_i \cdot Y_j \cdot Z_{ij} \cdot e^{-dist_{ij}/2\Delta},$$

where $dist_{ij}$ is the Euclidean distance between nodes $i$ and $j$, and $\Delta$ is the maximum Euclidean distance between any pair of nodes. The parameter $\alpha$ is used to generate instances with different degrees of variability on the traffic. Let $p \in \{1, 2, ..., 10\}$ be the number that identifies the instance, and $\mathcal{N}(m, v)$ be a function that returns a random number respecting the Gaussian distribution, with mean $m$, and variance $v$. The value of $\alpha$ is given by

$$\alpha = \mathcal{N}(20, p).$$

Therefore, each instance has a different variance, and thus a different degree of variability on the demands.

Table 4.1 summarizes the characteristics of each instance set. The first column gives the name of the network used. The second and third columns show respectively the number of nodes and arcs. The number of scenarios in each instance is displayed in the fourth column. The last column shows the number of instances for that network.

| | | $\|N\|$ | $\|A\|$ | $\|R\|$ | Instances |
|---|---|---|---|---|---|
| Realistic | `abilene` | 12 | 15 | 24 | 5 |
| | `geant` | 22 | 36 | 24 | 6 |
| | `nobel` | 17 | 26 | 24 | 1 |
| | `germany` | 50 | 88 | 24 | 1 |
| Artificial | `hier` | 20 | 58 | 24 | 10 |

Table 4.1: Characteristics of the networks used in the experiments.

## 4.2 Results

In the first experiment, IGP-WO, BRKGA, and IGP-AVE are executed for 10 minutes, and the values shown are the average of 20 runs, with different seeds for the pseudo-random number generator. The results for *minmax OSPF*, *minmax regret OSPF* and *minmax relative regret OSPF* are presented in Tables 4.2, 4.3 and 4.4, respectively. The first column displays the name of the instance. The next three columns show the results for IGP-WO. The second column gives the relative improvement of IGP-WO over IGP-AVE (i.e. (IGP-AVE − IGP-WO) / IGP-AVE). The third column displays the coefficient of variation of the executions, i.e., the standard variation divided by the mean. The fourth columns shows the number of iterations performed by the algorithm. The last three columns show the same values for BRKGA.

| | IGP-WO | | | BRKGA | | |
|---|---|---|---|---|---|---|
| | Improvement | CV | Iterations | Improvement | CV | Generations |
| abilene1 | -1.05% | 0.71% | 7,110.48 | -0.61% | 0.58% | 3,971.90 |
| abilene2 | -0.31% | 0.00% | 6,887.76 | -0.32% | 0.03% | 3,474.24 |
| abilene3 | -2.09% | 0.77% | 6,698.48 | -1.81% | 0.93% | 3,872.76 |
| abilene4 | 0.66% | 0.05% | 23,622.29 | 0.64% | 0.11% | 3,193.81 |
| abilene5 | -2.63% | 0.54% | 7,259.24 | -0.54% | 0.93% | 3,974.43 |
| geant1 | -0.23% | 0.43% | 875.67 | 0.07% | 0.36% | 642.33 |
| geant2 | -0.44% | 0.31% | 725.71 | -0.23% | 0.27% | 772.95 |
| geant3 | -1.82% | 0.22% | 781.48 | -1.58% | 0.30% | 828.24 |
| geant4 | -0.99% | 0.11% | 829.95 | -0.57% | 0.23% | 618.67 |
| geant5 | 1.07% | 0.33% | 766.57 | 1.31% | 0.29% | 626.71 |
| geant6 | 1.28% | 0.35% | 782.19 | 1.26% | 0.28% | 592.67 |
| nobel | -1143.49% | 20.41% | 1,734.19 | -1198.80% | 0.03% | 1,680.19 |
| germany | -13.20% | 77.97% | 337.00 | -1.22% | 5.43% | 38.24 |
| hier1 | 0.02% | 0.23% | 1,126.43 | -0.83% | 2.34% | 649.00 |
| hier2 | -0.03% | 0.30% | 1,162.95 | -0.65% | 0.95% | 687.14 |
| hier3 | -0.06% | 0.30% | 1,146.95 | -0.83% | 1.40% | 642.05 |
| hier4 | -0.05% | 0.38% | 1,429.57 | -0.76% | 0.65% | 666.81 |
| hier5 | -0.11% | 0.36% | 1,105.62 | -0.77% | 0.88% | 650.48 |
| hier6 | -0.04% | 0.35% | 1,180.90 | -0.62% | 0.77% | 674.43 |
| hier7 | 0.16% | 0.30% | 1,164.76 | -0.79% | 0.97% | 646.71 |
| hier8 | 0.03% | 0.34% | 1,220.00 | -0.49% | 0.72% | 632.71 |
| hier9 | -0.04% | 0.40% | 1,223.19 | -0.54% | 1.30% | 637.67 |
| hier10 | -0.18% | 0.42% | 1,190.95 | -0.59% | 0.88% | 686.75 |
| Average | -50.59% | 4.59% | 3,059.23 | -52.58% | 0.90% | 1,341.78 |

Table 4.2: Results of the algorithms of the literature for *minmax OSPF*

It can be observed in Table 4.2 that, for *minmax OSPF*, IGP-WO is better than IGP-AVE in 6 out of the 23 instances, and BRKGA in 4 out of the 23. Apart from instances `nobel` and `germany`, IGP-WO is at most 2.63% worse than IGP-AVE, and BRKGA is at most 1.51%. On average, IGP-WO is 50.59% worse than IGP-AVE, and

BRKGA is 52.58% worse. Comparing IGP-WO and BRKGA, IGP-WO have better results on 14 of 23 instances. Therefore, IGP-AVE is the best choice between the three algorithms for *minmax OSPF*. It performs better because it considers multiple demand matrices, and not only a single peak matrix, avoiding the oversimplification of the traffic.

| | IGP-WO | | | BRKGA | | |
|---|---|---|---|---|---|---|
| | Improvement | CV | Iterations | Improvement | CV | Generations |
| abilene1 | -7.23% | 3.31% | 7,110.48 | -4.59% | 2.89% | 3,971.90 |
| abilene2 | -16.92% | 0.00% | 6,887.76 | -17.17% | 0.53% | 3,474.24 |
| abilene3 | -682.16% | 20.25% | 6,698.48 | -620.37% | 28.02% | 3,872.76 |
| abilene4 | 6.58% | 0.51% | 23,622.29 | 6.38% | 1.16% | 3,193.81 |
| abilene5 | -72.74% | 8.89% | 7,259.24 | -14.92% | 22.49% | 3,974.43 |
| geant1 | -92.75% | 9.22% | 875.67 | -81.30% | 4.94% | 642.33 |
| geant2 | -138.68% | 9.20% | 725.71 | -115.06% | 9.33% | 772.95 |
| geant3 | -167.46% | 5.16% | 781.48 | -154.42% | 6.00% | 828.24 |
| geant4 | -73.22% | 3.68% | 829.95 | -55.39% | 4.54% | 618.67 |
| geant5 | -42.98% | 8.71% | 766.57 | -43.51% | 3.03% | 626.71 |
| geant6 | -19.88% | 7.23% | 782.19 | -19.98% | 5.22% | 592.67 |
| nobel | -4497.36% | 21.95% | 1,734.19 | -4717.27% | 0.03% | 1,680.19 |
| germany | -28.64% | 261.46% | 337.00 | 18.61% | 24.41% | 38.24 |
| hier1 | -41.59% | 13.73% | 1,126.43 | -75.09% | 86.22% | 649.00 |
| hier2 | -29.86% | 10.93% | 1,162.95 | -59.16% | 32.32% | 687.14 |
| hier3 | -28.23% | 12.41% | 1,146.95 | -63.34% | 44.29% | 642.05 |
| hier4 | -22.92% | 12.79% | 1,429.57 | -54.22% | 20.46% | 666.81 |
| hier5 | -11.83% | 14.79% | 1,105.62 | -52.15% | 33.81% | 650.48 |
| hier6 | -6.10% | 12.48% | 1,180.90 | -32.21% | 21.23% | 674.43 |
| hier7 | 1.58% | 11.24% | 1,164.76 | -40.85% | 24.31% | 646.71 |
| hier8 | -2.78% | 10.10% | 1,220.00 | -34.04% | 22.13% | 632.71 |
| hier9 | -6.21% | 10.87% | 1,223.19 | -31.47% | 30.91% | 637.67 |
| hier10 | -17.15% | 13.51% | 1,190.95 | -30.65% | 17.26% | 686.75 |
| Average | -260.81% | 20.97% | 3,059.23 | -273.57% | 19.37% | 1,341.78 |

Table 4.3: Results of the algorithms of the literature for *minmax regret OSPF*

The Table 4.3 shows that, for *minmax regret OSPF*, IGP-WO and BRKGA were, on average, 260.81% and 273.53% worse than IGP-AVE, respectively. Both heuristics had better results than IGP-AVE in 2 out of the 23 instances. IGP-WO was better than BRKGA in 15 out of the 23 instances. Table 4.4 presents similar results. IGP-WO and BRKGA are worse than IGP-AVE, and IGP-WO is slightly better than BRKGA. Analogously to the *minmax OSPF*, IGP-AVE performs better on *minmax regret OSPF* and *minmax relative regret OSPF* because it considers multiple demand matrices.

In the second experiment, we compare IGP-AVE approach, that optimizes the average congestion, with the two approaches proposed in this dissertation, that optimize each of the three robust criteria. Each algorithm was executed for 10 minutes, and

| | IGP-WO | | | BRKGA | | |
|---|---|---|---|---|---|---|
| | Improvement | CV | Iterations | Improvement | CV | Generations |
| abilene1 | -24.46% | 8.45% | 7,110.48 | -10.94% | 9.59% | 3,971.90 |
| abilene2 | -22.50% | 0.00% | 6,887.76 | -22.86% | 0.73% | 3,474.24 |
| abilene3 | -1023.81% | 10.78% | 6,698.48 | -987.30% | 16.80% | 3,872.76 |
| abilene4 | -3.34% | 0.39% | 23,622.29 | -3.22% | 0.31% | 3,193.81 |
| abilene5 | -70.43% | 8.76% | 7,259.24 | -13.41% | 22.46% | 3,974.43 |
| geant1 | -81.40% | 8.76% | 875.67 | -71.24% | 3.78% | 642.33 |
| geant2 | -196.00% | 7.54% | 725.71 | -176.86% | 7.79% | 772.95 |
| geant3 | -161.88% | 5.56% | 781.48 | -150.67% | 6.80% | 828.24 |
| geant4 | -96.46% | 3.03% | 829.95 | -89.76% | 1.81% | 618.67 |
| geant5 | -34.40% | 8.81% | 766.57 | -34.95% | 3.12% | 626.71 |
| geant6 | -96.73% | 4.50% | 782.19 | -96.43% | 2.91% | 592.67 |
| nobel | -4536.46% | 22.12% | 1,734.19 | -4759.92% | 0.03% | 1,680.19 |
| germany | -18.19% | 288.25% | 337.00 | 37.69% | 18.79% | 38.24 |
| hier1 | -127.27% | 14.66% | 1,126.43 | -145.13% | 59.85% | 649.00 |
| hier2 | -107.25% | 12.55% | 1,162.95 | -131.42% | 24.94% | 687.14 |
| hier3 | -102.45% | 15.21% | 1,146.95 | -115.53% | 30.51% | 642.05 |
| hier4 | -92.71% | 13.83% | 1,429.57 | -116.08% | 21.21% | 666.81 |
| hier5 | -73.23% | 13.61% | 1,105.62 | -103.66% | 29.21% | 650.48 |
| hier6 | -60.04% | 12.94% | 1,180.90 | -80.54% | 15.81% | 674.43 |
| hier7 | -47.48% | 12.75% | 1,164.76 | -89.91% | 18.74% | 646.71 |
| hier8 | -51.29% | 11.11% | 1,220.00 | -87.44% | 25.58% | 632.71 |
| hier9 | -60.70% | 8.74% | 1,223.19 | -83.78% | 24.12% | 637.67 |
| hier10 | -62.06% | 10.33% | 1,190.95 | -69.94% | 11.96% | 686.75 |
| Average | -310.89% | 21.86% | 3,059.23 | -321.88% | 15.52% | 1,341.78 |

Table 4.4: Results of the algorithms of the literature for *minmax relative regret OSPF*

the values shown are the average of 20 runs with different seeds for the pseudo-random number generator. The results for *minmax OSPF*, *minmax regret OSPF* and *minmax relative regret OSPF* are presented in Tables 4.5, 4.6 and 4.7 respectively. The first column displays the name of the instance. The next three columns show the results for IGP-RO. The second column gives the relative improvement of IGP-RO over IGP-AVE (i.e. (IGP-AVE – IGP-RO) / IGP-AVE). The third column displays the coefficient of variation of the executions, i.e., the standard variation divided by the mean. The fourth columns shows the number of iterations performed by the algorithm. The last three columns show the same values for BRKGA-RO.

Table 4.5 shows that IGP-RO was worse than IGP-AVE only on the `germany` instance, and obtained an average improvement of 1.29% over IGP-AVE. BRKGA-RO was better than IGP-AVE in 12 out of the 13 realistic instances, but was worse on all 10 artificial instances. The inferior performance of BRKGA-RO can be explained by the size of the instances. Networks `abilene`, `geant` and `nobel` have up to 36 arcs, while `germany` and `hier` have 58 and 88 arcs, respectively. With larger networks, the time

| | IGP-RO | | | BRKGA-RO | | |
|---|---|---|---|---|---|---|
| | Improvement | CV | Iterations | Improvement | CV | Generations |
| abilene1 | 0.12% | 0.05% | 2,686.19 | 0.07% | 0.04% | 406.24 |
| abilene2 | 0.43% | 0.05% | 2,573.05 | 0.23% | 0.23% | 364.38 |
| abilene3 | 0.29% | 0.00% | 2,346.62 | 0.29% | 0.00% | 391.00 |
| abilene4 | 0.79% | 0.27% | 5,273.24 | 0.49% | 0.38% | 336.67 |
| abilene5 | 0.42% | 0.01% | 2,744.43 | 0.39% | 0.04% | 385.10 |
| geant1 | 1.16% | 0.07% | 199.81 | 1.00% | 0.18% | 72.29 |
| geant2 | 1.06% | 0.13% | 250.33 | 1.09% | 0.13% | 79.76 |
| geant3 | 0.17% | 0.37% | 182.67 | 0.28% | 0.05% | 82.10 |
| geant4 | 0.87% | 0.11% | 285.43 | 0.69% | 0.21% | 75.10 |
| geant5 | 1.57% | 0.20% | 217.38 | 1.56% | 0.14% | 68.81 |
| geant6 | 1.88% | 0.34% | 229.95 | 1.89% | 0.28% | 70.57 |
| nobel | 2.83% | 1.27% | 893.38 | 2.06% | 0.97% | 178.10 |
| germany | -2.48% | 27.92% | 118.33 | -15.87% | 8.66% | 6.00 |
| hier1 | 1.37% | 0.13% | 464.29 | -1.90% | 2.18% | 79.33 |
| hier2 | 1.38% | 0.43% | 467.38 | -2.02% | 1.62% | 76.29 |
| hier3 | 1.60% | 0.13% | 415.29 | -1.09% | 1.33% | 78.62 |
| hier4 | 1.70% | 0.31% | 360.71 | -1.51% | 1.28% | 76.05 |
| hier5 | 1.95% | 0.18% | 370.57 | -2.24% | 2.24% | 77.90 |
| hier6 | 2.20% | 0.15% | 371.33 | -2.12% | 2.17% | 78.24 |
| hier7 | 2.43% | 0.12% | 378.38 | -0.71% | 1.49% | 81.48 |
| hier8 | 2.41% | 0.67% | 374.10 | -0.82% | 1.51% | 81.48 |
| hier9 | 2.74% | 0.16% | 370.10 | -1.23% | 1.46% | 80.29 |
| hier10 | 2.68% | 0.12% | 371.81 | -1.76% | 2.41% | 81.38 |
| Average | 1.29% | 1.44% | 954.12 | -0.92% | 1.26% | 143.79 |

Table 4.5: Results of the robust algorithms for *minmax OSPF*

required to calculate the cost of a solution increases. On network `germany`, BRKGA-RO was able to perform only 6 generations in 10 minutes, while IGP-RO executed about 118.33 iterations on average. Therefore, the time for BRKGA-RO to converge to good solutions on large sized instances is much higher than that of IGP-RO.

The results for *minmax regret OSPF* and *minmax relative regret OSPF* are displayed in Tables 4.6 and 4.7. They show that IGP-RO was better than IGP-AVE and BRKGA-RO for most of the instances. IGP-RO had an average improvement of 24.93% over IGP-AVE on *minmax regret OSPF*, and 16.10% on *minmax relative regret OSPF*. Again, BRKGA-RO had worse results than IGP-RO for larger networks.

Figure 4.1 shows the improvement of IGP-RO over IGP-AVE on the 10 instances with artificial networks. Recall that each of these instances have a different level of variation on the traffic, with 1 being the most uniform, and 10 being the most varied. For *minmax OSPF* the improvement starts in 1.37% and goes up to 2.74% on the ninth instance. For *minmax regret OSPF*, the impact of the variation of the traffic on the improvement is more clear. On instance 1 it is 39.31%, and reaches 57.21% toward

| | IGP-RO | | | BRKGA-RO | | |
|---|---|---|---|---|---|---|
| | Improvement | CV | Iterations | Improvement | CV | Generations |
| abilene1 | 0.94% | 0.26% | 2,705.62 | 0.46% | 1.13% | 398.52 |
| abilene2 | 9.56% | 4.75% | 2,552.67 | 2.42% | 5.85% | 383.14 |
| abilene3 | 0.00% | 0.00% | 2,226.71 | 0.00% | 0.00% | 393.90 |
| abilene4 | 8.51% | 0.92% | 3,981.14 | 2.34% | 3.49% | 342.24 |
| abilene5 | 11.56% | 0.21% | 2,743.19 | 10.82% | 1.35% | 388.24 |
| geant1 | -2.32% | 9.36% | 194.10 | 0.19% | 11.20% | 74.52 |
| geant2 | -4.55% | 15.59% | 246.57 | -10.83% | 13.37% | 77.43 |
| geant3 | -9.34% | 18.87% | 186.00 | -10.05% | 8.14% | 84.38 |
| geant4 | 7.75% | 11.72% | 271.29 | 7.47% | 7.28% | 71.57 |
| geant5 | -0.30% | 11.07% | 208.62 | 6.90% | 6.91% | 69.29 |
| geant6 | 21.85% | 17.63% | 206.24 | 32.21% | 6.71% | 71.81 |
| nobel | 35.11% | 8.91% | 1,603.00 | 24.89% | 16.87% | 180.62 |
| germany | 0.81% | 119.00% | 117.00 | -42.62% | 25.81% | 6.05 |
| hier1 | 39.31% | 23.70% | 420.67 | -286.41% | 35.86% | 78.05 |
| hier2 | 41.78% | 13.25% | 428.90 | -319.41% | 60.08% | 76.19 |
| hier3 | 48.43% | 11.78% | 423.95 | -188.94% | 49.85% | 80.43 |
| hier4 | 45.73% | 15.18% | 425.29 | -177.82% | 41.20% | 76.86 |
| hier5 | 49.72% | 12.49% | 427.43 | -186.88% | 44.62% | 79.19 |
| hier6 | 51.79% | 11.71% | 438.29 | -157.23% | 41.83% | 77.67 |
| hier7 | 53.56% | 10.27% | 415.90 | -106.16% | 34.03% | 81.05 |
| hier8 | 56.07% | 9.43% | 449.05 | -106.59% | 35.09% | 79.00 |
| hier9 | 57.21% | 8.61% | 437.67 | -117.80% | 44.48% | 81.57 |
| hier10 | 50.15% | 11.94% | 446.43 | -129.51% | 38.43% | 81.71 |
| Average | 24.93% | 15.07% | 937.20 | -76.20% | 23.20% | 144.93 |

Table 4.6: Results of the robust algorithms for *minmax regret OSPF*

the last instances. The same is seen on *minmax relative regret OSPF*. It starts with 24.03% of improvement, and reaches 43.89%. This happens because, as the level of variation increases, the regret of the solutions also increase. Therefore, the approaches that optimize the regret perform better.

The last analysis is presented on Table 4.8, comparing IGP-WO, BRKGA, IGP-RO and IGP-AVE for the average network congestion objective, used in [Fortz and Thorup, 2002]. The first column shows the instance. The second and third columns show the improvement on the network congestion of IGP-WO and BRKGA over IGP-AVE. The next three columns show the improvement on the network congestion of IGP-RO over IGP-AVE using the minmax, minmax regret and minmax relative regret objectives, respectively. Using a minmax objective results in the biggest average loss, of 1.55%. This corroborate with the fact that minmax is the most conservative objective. Comparing with Tables 4.5, 4.6 and 4.7, using *minmax OSPF* results in an improvement of 1.29% on its objective, while reducing 1.55% on the total congestion objective. On the other hand, with *minmax regret OSPF*, the loss is on average 0.72%, but the

| | IGP-RO | | | BRKGA-RO | | |
|---|---|---|---|---|---|---|
| | Improvement | CV | Iterations | Improvement | CV | Generations |
| abilene1 | 1.54% | 0.25% | 2,682.43 | 0.88% | 0.24% | 413.86 |
| abilene2 | 12.14% | 1.86% | 2,517.81 | -0.83% | 6.43% | 392.67 |
| abilene3 | 0.00% | 0.00% | 2,207.38 | 0.00% | 0.00% | 390.10 |
| abilene4 | 1.72% | 0.38% | 2,865.76 | 0.00% | 1.23% | 351.86 |
| abilene5 | 5.14% | 0.61% | 2,866.62 | 4.76% | 1.88% | 399.67 |
| geant1 | 23.88% | 11.37% | 212.24 | 20.84% | 7.05% | 74.95 |
| geant2 | -20.57% | 24.78% | 255.00 | -20.57% | 16.87% | 80.00 |
| geant3 | 0.22% | 22.71% | 188.95 | 7.85% | 10.79% | 80.48 |
| geant4 | 14.27% | 18.06% | 268.71 | 12.68% | 10.20% | 78.38 |
| geant5 | 21.61% | 13.93% | 202.52 | 32.19% | 4.06% | 70.86 |
| geant6 | 3.27% | 15.03% | 214.05 | 8.77% | 4.03% | 70.71 |
| nobel | 35.24% | 7.79% | 2,790.67 | 27.46% | 14.46% | 178.90 |
| germany | -74.93% | 130.57% | 117.57 | -49.24% | 84.66% | 6.00 |
| hier1 | 24.03% | 10.04% | 415.86 | -361.69% | 44.00% | 78.24 |
| hier2 | 24.77% | 12.89% | 415.43 | -373.11% | 26.71% | 74.43 |
| hier3 | 31.34% | 10.43% | 413.10 | -292.92% | 48.35% | 77.38 |
| hier4 | 29.90% | 16.36% | 415.38 | -212.31% | 44.93% | 79.52 |
| hier5 | 33.87% | 7.93% | 418.90 | -286.50% | 57.67% | 76.19 |
| hier6 | 38.08% | 12.71% | 438.48 | -233.05% | 54.95% | 79.67 |
| hier7 | 42.24% | 9.03% | 429.67 | -163.18% | 48.27% | 81.43 |
| hier8 | 43.89% | 9.57% | 429.00 | -204.48% | 29.98% | 79.48 |
| hier9 | 39.13% | 18.14% | 430.71 | -190.64% | 29.09% | 80.86 |
| hier10 | 39.55% | 9.23% | 481.67 | -185.21% | 39.33% | 114.86 |
| Average | 16.10% | 15.81% | 942.52 | -106.88% | 25.44% | 148.28 |

Table 4.7: Results of the robust algorithms for *minmax relative Regret OSPF*

improvement on the regret is 24.93%. Similarly, with minmax relative regret OSPF, the loss on the total congestion objective is 1.05%, while the improvement on the relative regret is 16.10%. These results show that by using an objective function based on the regret, one can sharply increase the performance of the network on the worst cases, without losing performance on the average case.

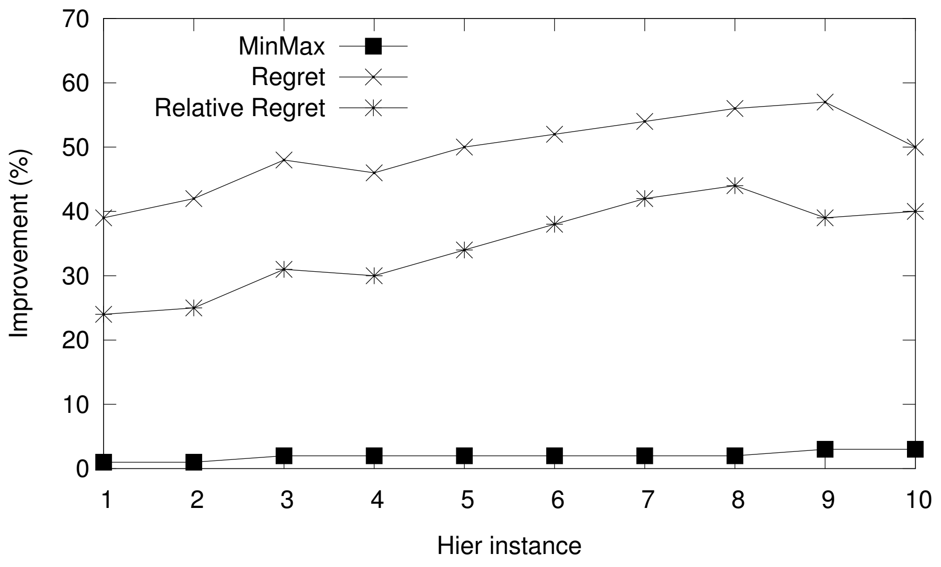


Figure 4.1: Improvement of IGP-RO for the artificial instances.

| | | | IGP-RO | | |
|---|---|---|---|---|---|
| | IGP-WO | BRKGA | MinMax | Regret | Relative Regret |
| abilene1 | -4.79% | -2.67% | -0.52% | -0.53% | -0.63% |
| abilene2 | -1.90% | -1.92% | -0.85% | -0.42% | -0.40% |
| abilene3 | -2.71% | -2.61% | -0.50% | 0.00% | 0.00% |
| abilene4 | -0.44% | -0.42% | -0.06% | -0.05% | 0.09% |
| abilene5 | -1.46% | -0.63% | -1.00% | -1.00% | -0.46% |
| geant1 | -2.20% | -1.90% | -0.97% | -0.42% | -0.41% |
| geant2 | -2.03% | -1.79% | -0.63% | -0.30% | -0.32% |
| geant3 | -2.07% | -1.96% | -0.70% | -0.25% | -0.36% |
| geant4 | -2.05% | -1.57% | -0.59% | -0.33% | -0.70% |
| geant5 | -1.04% | -0.91% | -0.93% | -0.37% | -0.56% |
| geant6 | -1.67% | -1.68% | -0.95% | -0.52% | -0.27% |
| nobel | -98.02% | -102.61% | -1.50% | -2.23% | -2.47% |
| germany | -1.89% | -6.13% | -6.05% | -5.08% | -13.54% |
| hier1 | -1.37% | -1.74% | -1.81% | -0.37% | -0.27% |
| hier2 | -1.41% | -1.80% | -1.79% | -0.44% | -0.28% |
| hier3 | -1.55% | -1.97% | -1.99% | -0.42% | -0.33% |
| hier4 | -1.60% | -2.11% | -1.88% | -0.49% | -0.40% |
| hier5 | -1.58% | -2.13% | -2.18% | -0.53% | -0.40% |
| hier6 | -1.72% | -2.00% | -2.08% | -0.51% | -0.44% |
| hier7 | -1.78% | -2.49% | -2.06% | -0.56% | -0.46% |
| hier8 | -2.02% | -2.46% | -2.27% | -0.54% | -0.42% |
| hier9 | -2.16% | -2.58% | -2.12% | -0.54% | -0.50% |
| hier10 | -2.26% | -2.44% | -2.26% | -0.61% | -0.54% |
| Average | -6.08% | -6.46% | -1.55% | -0.72% | -1.05% |

Table 4.8: Results of IGP-WO, BRKGA and IGP-RO for total network congestion.

# Chapter 5

# Conclusion

The OSPF weight setting problem consists in defining the routes of the data on a computer network ruled by OSPF protocol, to satisfy an objective function. In this work, we proposed three optimization models for the problem, based on the Robust Optimization framework of [Kouvelis and Yu, 1997]. Instead of minimizing the average network congestion as is the case of the other works in the literature, we considered each scenario individually, since a solution that is good on average may sometimes result in a bad quality-of-service in some of the scenarios. The *minmax OSPF* is the most conservative model, while *minmax regret OSPF* and *minmax relative regret OSPF* are less conservative.

Three approaches of the literature were applied to the robust models, on realistic and artificial networks. Comparing the approaches, we conclude that IGP-AVE provides better results than IGP-WO and BRKGA, as IGP-AVE considers multiple demand matrices instead of a peak matrix. Further, we introduced extensions of algorithms of the literature, and performed computational experiments with them. The results pointed out that the classic approach of minimizing the average congestion may result in the network being over congested in some of the scenarios, resulting in a potential bad quality of-service from the user point-of-view. With our approaches, we are able to have networks with 24.93% less regret, while increasing in only 0.72% the average congestion, and 16.10% less relative regret, with 1.02% increase in the average congestion. This indicates that our optimization models may be a better alternative for weight setting in OSPF networks.

## 5.1 Future works

IGP-RO can replace IGP-WO in the iterative two-step heuristic of Altin et al. [2012], in order to efficiently optimize an exponentially large number of scenarios. Future works may also develop new exact or heuristic algorithms for the optimization problems presented here, or propose new lower bounds for the cost of the optimal solutions for these problems. Besides, the optimization models proposed here may be extended for other IGP protocols, such as IS-IS Callon [1990] and DEFT Xu et al. [2007].

# Bibliography

Abrahamsson, H. and Bjorkman, M. (2009). Robust traffic engineering using l-balanced weight-settings in OSPF/IS-IS. In *Broadband Communications, Networks, and Systems, 2009. BROADNETS 2009. Sixth International Conference on*, pages 1--8. IEEE.

Altın, A., Belotti, P., and Pınar, M. Ç. (2010). OSPF routing with optimal oblivious performance ratio under polyhedral demand uncertainty. *Optimization and Engineering*, 11(3):395--422.

Altın, A., Fortz, B., Thorup, M., and Ümit, H. (2009). Intra-domain traffic engineering with shortest path routing protocols. *4OR*, 7(4):301--335.

Altin, A., Fortz, B., and Ümit, H. (2012). Oblivious OSPF routing with weight optimization under polyhedral demand uncertainty. *Networks*, 60(2):132--139.

Applegate, D. and Cohen, E. (2003). Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 313--324. ACM.

Applegate, D. and Cohen, E. (2006). Making routing robust to changing traffic demands: algorithms and evaluation. *IEEE/ACM Transactions on Networking (TON)*, 14(6):1193--1206.

Balon, S., Skivée, F., and Leduc, G. (2006). How well do traffic engineering objective functions meet TE requirements? In *NETWORKING 2006. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, pages 75--86. Springer.

Belotti, P. and Mustafa, C. (2008). Optimal oblivious routing under statistical uncertainty. *Optimization and Engineering*, pages 257--2271.

Ben-Ameur, W. and Kerivin, H. (2005). Routing of uncertain traffic demands. *Optimization and Engineering*, 6(3):283--313.

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.

Bley, A., Fortz, B., Gourdin, E., Holmberg, K., Klopfenstein, O., Pióro, M., Tomaszewski, A., and Ümit, H. (2010). Optimization of OSPF routing in IP networks. In *Graphs and Algorithms in Communication Networks*, pages 199--240. Springer.

Broström, P. and Holmberg, K. (2006). Multiobjective design of survivable IP networks. *Annals of Operations Research*, 147(1):235--253.

Buriol, L. S., Resende, M. G., Ribeiro, C. C., and Thorup, M. (2005). A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks*, 46(1):36--56.

Buriol, L. S., Resende, M. G., and Thorup, M. (2008). Speeding up dynamic shortest-path algorithms. *INFORMS Journal on Computing*, 20(2):191--204.

Callon, R. W. (1990). Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. RFC 1195, Internet Engineering Task Force.

Coco, A. A., Júnior, J. a. C. A., Noronha, T. F., and Santos, A. C. (2014). An integer linear programming formulation and heuristics for the minmax relative regret robust shortest path problem. *Journal of Global Optimization*, pages 1--23.

El Ghaoui, L. and Lebret, H. (1997). Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035--1064.

El Ghaoui, L., Oustry, F., and Lebret, H. (1998). Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9(1):33--52.

Ericsson, M., Resende, M. G. C., and Pardalos, P. M. (2002). A genetic algorithm for the weight setting problem in OSPF routing. *Journal of combinatorial optimization*, 6(3):299--333.

Fortz, B. (2011). Applications of meta-heuristics to traffic engineering in IP networks. *International transactions in operational research*, 18(2):131--147.

Fortz, B. and Thorup, M. (2000). Internet traffic engineering by optimizing OSPF weights. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 519--528. IEEE.

Fortz, B. and Thorup, M. (2002). Optimizing OSPF/IS-IS weights in a changing world. *IEEE journal on selected areas in communications*, 20(4).

Fortz, B., Thorup, M., et al. (2003). Robust optimization of OSPF/IS-IS weights. In *Proc. INOC*, pages 225--230.

Fortz, B. and Ümit, H. (2011). Efficient techniques and tools for intra-domain traffic engineering. *International transactions in operational research*, 18(3):359--376.

Gabrel, V., Murat, C., and Thiele, A. (2013). Recent advances in robust optimization: An overview. *European Journal of Operational Research.*

Geoffrion, A. M. (1974). *Lagrangean relaxation for integer programming.* Springer.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. (1989). Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations research*, 37(6):865--892.

Kasperski, A., Kobylanski, P., Kulej, M., and Zielinski, P. (2005). Minimizing maximal regret in discrete optimization problems with interval data. *Issues in Soft Computing Decisions and Operations Research*, pages 193--208.

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

Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3--30.

Moy, J. T. (1998). *OSPF: anatomy of an Internet routing protocol.* Addison-Wesley Professional.

Mulyana, E. and Killat, U. (2005). Optimizing IP networks for uncertain demands using outbound traffic constraints. In *Proceedings of INOC*, volume 2005, pages 695--701. Citeseer.

Orlowski, S., Wessäly, R., Pióro, M., and Tomaszewski, A. (2010). SNDlib 1.0—survivable network design library. *Networks*, 55(3):276--286.

Parmar, A., Ahmed, S., and Sokol, J. (2006). An integer programming approach to the OSPF weight setting problem. *Optimization Online*.

Pióro, M. and Medhi, D. (2004). *Routing, flow, and capacity design in communication and computer networks*. Elsevier.

Pióro, M., Szentesi, A., Harmatos, J., Jüttner, A., Gajowniczek, P., and Kozdrowski, S. (2002). On open shortest path first related network optimisation problems. *Performance evaluation*, 48(1):201--223.

Reis, R., Ritt, M., Buriol, L. S., and Resende, M. G. (2011). A biased random-key genetic algorithm for OSPF and DEFT routing to minimize network congestion. *International Transactions in Operational Research*, 18(3):401--423.

Soyster, A. (1973). Technical note—convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations research*, 21(5):1154--1157.

Srivastava, S., Agrawal, G., Pioro, M., and Medhi, D. (2005). Determining link weight system under various objectives for OSPF networks using a lagrangian relaxation-based approach. *Network and Service Management, IEEE Transactions on*, 2(1):9--18.

Tabatabaee, V., Kashyap, A., Bhattacharjee, S., La, R. J., and Shayman, M. A. (2007). Robust routing with unknown traffic matrices. In *INFOCOM*, pages 2436--2440.

Xu, D., Chiang, M., and Rexford, J. (2007). DEFT: Distributed exponentially-weighted flow splitting. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 71--79. IEEE.

Zegura, E. W., Calvert, K. L., Bhattacharjee, S., et al. (1996). How to model an internetwork. In *IEEE infocom*, volume 96, pages 594--602. IEEE.

Zhanga, C., Kurose, J., Towsley, D., Ge, Z., and Liu, Y. (2005). Optimal routing with multiple traffic matrices: Tradeoff between average and worst case performance. In *Network Protocols, 2005. ICNP 2005. 13th IEEE International Conference on*, pages 10 pp.–.