A Local Search for the Robust OSPF Weight Setting Problem

Daniel Brasil Magnani, Thiago Ferreira de Noronha

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

{daniel,tfn}@dcc.ufmg.br

Resumo

O problema de atribuição de pesos OSPF consiste em determinar pesos para todos os links de uma rede de computadores. Estes pesos induzem uma rota entre todos os pares de roteadores, e consequentemente dizem como o tráfego é roteado pela rede. Neste trabalho uma versão robusta do problema de atribuição de pesos OSPF é apresentada, levando em consideração a variação nas demandas em redes de computadores. Nós mostramos formas de modelar o problema, e o resolvemos com heurísticas. Por fim, nós executamos experimentos computacionais para avaliar a performance das diferentes modelagens para o problema.
Palavras-chave: OSPF; otimização robusta; busca local; heurísticas;

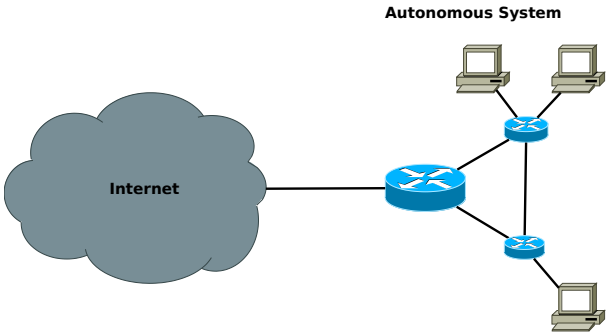
Abstract

The OSPF weight setting problem consists in assigning weights to all links in a computer network. These weights induce a route for all pairs of routers, and hence tell how the traffic is routed thought the network. In this work a robust version of the OSPF weight setting problem is presented, taking into consideration the variation on the demands of computer networks. We show ways to model the problem, and solve it with heuristics. Finally, we execute computational experiments to evaluate the performance of different modelings for the problem.
Keywords: OSPF; robust optimization; local search; heuristics;

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.

The network that has its own infrastructure and rules is called an autonomous system (AS). The Internet can be divided into two protocols layers: 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.



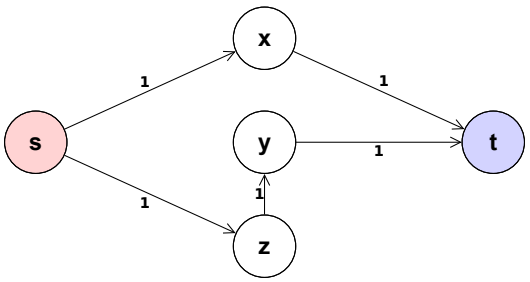
The Open Shortest Path First (OSPF) protocol is an IGP that was created by the OSPF working group of the Internet Engineering Task Force [Moy, 1989]. In OSPF the routers maintain an identical database containing information about the network topology, and regularly exchange data with other routers in the same AS to keep the database up-to-date.

The routing problem of an AS consists in defining the paths in which the network packets are going to pass inside that AS, in order to obtain a better use of the network infrastructure. As there are many possible routes for each packet, deciding the path of the information is crucial to avoid link overloading and to keep the network reliable.

In OSPF routing, the network operator assigns integer weights to each link of the network. These weights are used as lengths to calculate the shortest paths between the nodes. The packets must be routed through the shortest paths. In the case of multiple shortest paths, the traffic is split evenly, in each node, among all outgoing shortest path links. Figure 1 shows an example of OSPF routing. In the first case (a), all links have the same weight, and all the 10 packets are routed through path $(s \to x \to t)$. On the second case (b), there are two shortest paths of size 3, and the packets are split evenly among them.

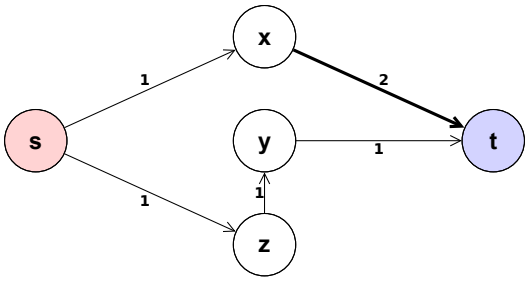
1.1 Single demand matrix model

The OSPF weight setting problem was modeled on Fortz and Thorup [2000] as follows. Given a directed graph $G = (N, A)$, where N is a set of nodes that represent the routers, A a set of arcs that represent the links, $c_a \in \mathbb{N}$ is the capacity of each link $a \in A$ and d is a demand matrix. Each element $d_{ij} \in \mathbb{N}$ tells the amount of information that must be sent between each pair $(i, j) \in N \times N$. The OSPF weight setting problem consists in assigning a weight $w_a \in \mathbb{N}$ for each $a \in A$. The weights induce a route for each pair of nodes in the network, as they are used to calculate the shortest paths between the nodes. The objective function used aims to reduce the congestion on the network, by using a linear increasing convex function Φ to calculate the network congestion. Let



(a)

Path	Length	#Packets
$(s \to x \to t)$	2	10
$(s \to z \to y \to t)$	3	0



(b)

Path	Length	#Packets
$(s \to x \to t)$	3	5
$(s \to z \to y \to t)$	3	5

Figure 1: The source node s sends 10 packets to the target node t. There are two possible paths: $(s \to x \to t)$ and $(s \to z \to y \to t)$. On (a), all arcs are set up with the same weight equal to 1. As the first path has weight 2, and the second weigh 3, all the packets from s to t go through the first path. On (b), the arc (x, t) has weight 2. In this case both paths have the same total weight equal to 3. Hence the packets are split evenly among them.

l_a be the total amount of flow passing through arc a, and $\phi_a(l_a)$ be a piecewise linear function that models the congestion cost of link a, as shown on Equation (1). A graphic representation of this function is given in Figure 2. The more an arc is congested, more expensive it is to add flow to it. Finally, the objective function Φ is the sum of ϕ_a for all arcs $a \in A$, as described on Equation (2).

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

$$\Phi = \sum_{a \in A} \phi_a(l_a) \tag{2}$$

Another objective function, used by Altin et al. [2009], aims to minimize the maximum utilization rate of an arc, as shown on Equation 3, where the utilization rate of an arc $a \in A$ is defined

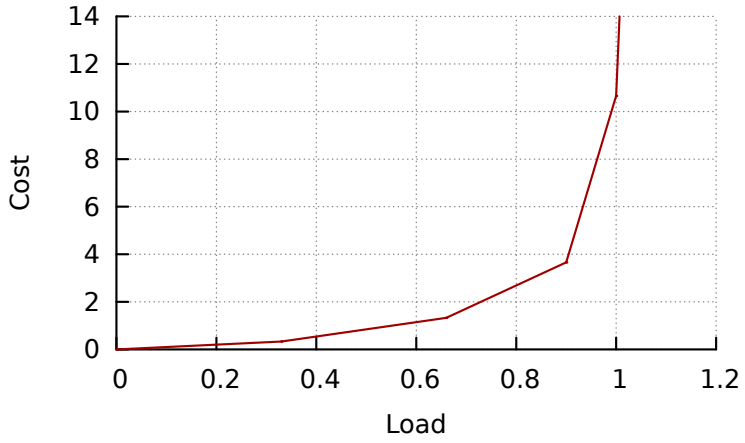


Figure 2: Cost function ϕ_a for arcs wih capacity $c_a = 1$

as $u_a = \frac{l_a}{c_a}$. Here all the effort is made to avoid the occurrence of an overloaded link. The weight setting problem in OSPF routing was proven NP-hard in [Fortz and Thorup, 2004].

$$\min \max_{a \in A} \{u_a\} \tag{3}$$

1.2 Multiple demand matrices model

The traffic on computer networks may vary through the day, due to unpredictable factors such as link failures and peaks of traffic. Unfortunately, it is not practical for the network operator to manually change the weights of the links each time a problem on traffic occurs. These factors motivated the development of algorithms for OSPF weight setting problem that deal with traffic uncertainties.

Fortz and Thorup [2002] introduced the idea of optimizing OSPF weights based on a set of demand matrices, representing scenarios of traffic. They tackle the problem by trying to find a single arrangement of weights that generates a route that is good for all scenarios. Given a set of scenarios (demand matrices) R and Φ_r, the cost of the solution for a scenario $r \in R$, the objective function is given by:

$$\min \sum_{r \in R} \Phi_r.$$

Altin et al. [2009] modeled the demand uncertainties by a system of linear inequalities that form a polyhedron $\mathfrak{D}$, where each solution inside $\mathfrak{D}$ is a possible demand matrix. This approach has the advantage of defining a range of values for the demand matrix, instead of a fixed set of matrices. The limits of $\mathfrak{D}$ can be easily obtained by measuring the maximum and minimum traffic of each link, avoiding errors that could happen on the estimation of the demands.

The Internet has grown from 5000 Autonomous Systems in 1999 to 42000 in 2012, each one responsible for routing huge amounts of information. Although alternatives to shortest path routing protocols emerged, such as Multi Label Switching Protocol, they did not manage to overcome OSPF [Altın et al., 2013]. The subject is still object of study and recent papers show that there is room for improvement in OSPF routing [Altın et al., 2013; Altin et al., 2012; Fortz and Ümit, 2011; Fortz, 2011].

The remainder of this text is organized as follows. Firstly, Section 2 overviews the related work to OSPF routing. Section 3 presents the formulations and algorithms proposed in this work to solve the problem. Finally, Section 4 presents the computational experiments and results obtained.

2 Related works

Before the works of Fortz and Thorup [2000], the common approach to set the weighs was assigning their values inversely proportional to the link bandwidth, as recommended by the major router vendor Cisco. Fortz and Thorup [2000] introduced the idea of optimizing the assignment of link weights, with the objective of inducing better routes for the flow on the network.

The first effort to automatically solve the OSPF weight setting problem was made by Fortz and Thorup [2000]. They proposed a local search heuristic that uses two kinds of neighborhoods. The first one consists of changing the weight of a single link, by choosing a random arc $a \in A$ and setting a new weight $w'_a \neq w_a$. The second tries to split the traffic as evenly as possible. More precisely, we take a node u that has flow going to target t, and set the weights of the arcs leaving u in a way that all paths from u to t have the same length. In order to promote diversification and reach better local minimums, non-improving moves are made, using a hash structure to avoid cycling between solutions. To evaluate the cost of a solution, the shortest paths between all pairs of sources and targets must be obtained. Fortz and Thorup [2000, 2004] argued that this procedure can be a bottleneck for the local search. They solve this problem by noticing that the difference between the shortest paths from one neighbor to another is small. Therefore, they recalculate the cost of the new solution using information from the previous, without having to compute all the shortest paths again.

Fortz [2011] improved the above local search by using an initial solution obtained from solving an easier problem. They relax the assumption of an evenly distributed flow between shortest paths, letting the flow be routed unconstrained, and use column generation to calculate a solution. They take advantage of the dual variables from the column generation procedure, using them as weights for OSPF weight setting problem.

Ericsson et al. [2002] used a genetic algorithm (GA) to solve the OSPF weight setting problem, named GAOSPF. It assigns fitness to each individual, and uses crossover and mutation operations to increase the population, maintaining the fittest individuals and discarding the weak. An individual is represented by a solution, i.e., a set of weights $w = \{w_a \in [1, 65535]; a \in A\}$. The initial population can be generated by choosing a random weight in $[1, 65535]$ for each arc. The fitness function used is the same used by Fortz and Thorup [2000] to calculate the cost Φ of a given solution. Knowing 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 of the population is created as follows: all individuals of $\mathcal{A}$ remain. $\mathcal{B}$ is replaced by the result of crossovers between $\mathcal{A}$ and $\mathcal{B} \cup \mathcal{C}$, and $\mathcal{C}$ is replaced by new random solutions.

Buriol et al. [2003] implemented a hybrid genetic algorithm (HGA), adding local searches in small neighborhoods to improve the genetic algorithm performance after the crossover operation. They were able to obtain better results in comparison to GAOSPF, and achieved solutions comparable to the local search of Fortz and Thorup [2000]. Later, Reis et al. [2011] introduced the concept of biased random-key genetic algorithm (BRKGA), and applied it to the OSPF weight setting problem.

There are two works in the literature that deals with multiple demand matrices. In the first, Fortz and Thorup [2002] modified their local search of [Fortz and Thorup, 2000] to minimize an objective function that takes into consideration a set of traffic matrices, instead of just one. The goal was minimizing the sum of the costs Φ for all matrices at once.

In the second work, Altin et al. [2009] and Altin et al. [2012] used the concept of polyhedral demands to the problem. The demand matrices are represented by a set of linear inequalities, forming a polyhedron $\mathfrak{D} = \{d \in \mathbb{R}; Bd \leq \alpha, d \geq 0\}$, with $B, \alpha \in \mathbb{R}$. These inequalities can be derived from the total bandwidth capacity of each link, or lower and upper bounds of demands. The goal is to find the weight setting that has the lower cost Φ, in the worst case scenario. They tackle the problem with a two-step algorithm. The first step aims at finding the worst case scenario for a

given route, i.e., the demand matrix that causes the cost to be maximum for that route. The second step is the execution of the local search from [Fortz and Thorup, 2002] to optimize the weights over the set of matrices found on the first step. They stop the algorithm whenever it does 50 iterations, or no traffic matrix is found on the first step.

3 Robust Optimization

Robust optimization is a branch of optimization that aims to solve problems that contain data uncertainty. In robust optimization, uncertainties are represented by discrete scenarios or continuous sets of values. It is mostly used when the probability distribution of the uncertain data is not known or when the problem is sensible to the worst case scenario. One of the first works on robust optimization was made Soyster [1973]. But it was 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 [1996]. Many classic problems have robust versions, such as Shortest Path, Minimum Spanning Tree, Linear Optimization, and Assignment Problem [Kasperski et al., 2005].

There are several optimization criteria for robust optimization. We discuss the most studied ones. Let R be the set of all possible scenarios for a problem, X be the set of feasible solutions for the problem, and $x \in X$ a solution for scenario $r \in R$. The *minmax* criterion consists in minimizing the worst case possible for all scenarios, as shown on Equation 4. It is used for problems in which the worst case must be avoided, such as nuclear accidents prevention, public health and construction of high voltage lines [Aissi et al., 2009].

$$\min_{x \in X} \max_{r \in R} \{x\}. \tag{4}$$

The *minmax regret* criterion is a less conservative approach. Let x_r^* be the optimal solution for scenario r. We define the regret (or robust deviation) of a solution $x \in X$ for a scenario r$\in R$, as the difference between the cost of x and x_r^*, i.e, the deviation of a given solution from optimum. The objective here is to minimize the biggest regret possible for all scenarios. Equation 5 describes *minmax regret* criterion.

$$\min_{x \in X} \max_{r \in R} \{x - x_r^*\}. \tag{5}$$

Another robust criterion is called *minmax relative regret*. Differently from the previous criterion, the regret is normalized by the optimal solution for each scenario, as seen on Equation 6. Hence the *minmax relative regret* is based on the percentage of deviation, while the *minmax regret* criterion is based on the absolute deviation.

$$\min_{x \in X} \max_{r \in R} \{\frac{x}{x_r^*} - 1\}. \tag{6}$$

In the present work we propose a robust version of the OSPF weigh setting problem, aiming to have a model that represents better the reality of networks. Instead of assuming one matrix of estimated demands, we use many matrices, representing different states of the network. This way, OSPF weights are optimized considering various scenarios at the same time. To the best of our knowledge, this is the first work that solves the OSPF weight setting problem with robust optimization. We compare the most important models for the problem, and observe which one gives the most reliable routes for the network.

3.1 Robust optimization model for OSPF weight setting problem

In this section we describe our approach to OSPF weight setting problem, called robust optimization model (ROM). Differently from the previous models, we do not assume a fixed demand matrix, but a set of matrices. Instead of optimizing a specific scenario of demands, e.g. the average scenario, we optimize the network for many possible scenarios at the same time. This approach provides a better representation of the reality of network usage, letting the network more reliable in case of changes on the traffic.

Our model makes use of the same constants and variables from the version defined on Section 1. The demand is represented by a set of matrices R, each matrix containing the flow d_{ij} from all pairs $(i, j) \in N \times N$. The goal is to find weights w_a for each link $a \in A$ that optimize a robust objective function.

The objective function of the ROM is based on the *minmax* criterion. Let Φ_r be the cost of solution x on scenario r. The objective is:

$$\min_{x \in X} \max_{r \in R} \{\Phi_r\}$$

This problem is a generalization of the OSPF weight setting problem, therefore it is also NP-Hard.

4 Computational Experiments

This section describes the computational experiments executed to evaluate the models. Three models for OSPF weight setting problem are compared. The first, referred as the *single demand matrix model (SDMM)* uses the objective function of [Fortz and Thorup, 2000], and considers a single demand matrix. The second, called *multiple demand matrices model (MDMM)*, is that proposed in [Fortz and Thorup, 2002]. The third is the *robust optimization model (ROM)*, proposed in this work.

4.1 Instances

The instances used for this experiment are taken from SNDLib [Orlowski et al., 2007]. Table 1 describes the characteristics of each instance. There are 4 network topologies, namely *abilene, geant, nobel* and *germany50*. The first four instances contain scenarios divided by days. The other four are divided by hour. For the SDMM, the demand matrices were merged into one, containing the average demand of all scenarios.

Instance	#Nodes	#Arcs	#Scenarios
abilene-day	12	15	35
geant-day	22	36	31
nobel-day	17	26	28
germany50-day	50	88	28
abilene-hour	12	15	24
geant-hour	22	36	24
nobel-hour	17	26	24
germany50-hour	50	88	24

Table 1: Characteristics of the instances used in the experiments. The first column shows the name of the instance, the second shows the number of nodes, the third shows the number of links, and the last column shows the number of scenarios.

The experiments were carried on an Intel Xeon CPU at 2.27GHz, with 8 cores, 8MB of cache and 16GB of RAM. The operating system used was Ubuntu version 10.04. The algorithms were coded in C++ and compiled with GCC version 4.6.3.

4.2 Heuristic

A version of the local search proposed in [Fortz and Thorup, 2002] was used to solve each of the optimization problems. Algorithm 1 shows the pseudo-code of this heuristic. An initial solution is created by assigning random weights, within the range $[0, 20]$, for each link. The local search is applied using the same neighborhood as [Fortz and Thorup, 2002], where a neighbor is any solution that can be reached by changing the weight of a single link. To select a neighbor, a random weight is assigned to a random link. Initially, 10% of the neighborhood is searched. Whenever a better solution is found on the neighborhood, the sampling is divided by 3, on the other case, when an better solution is not found, the sampling is doubled. The local search moves to the best solution found inside the sampling area. If there is no improving solution, it moves to one with the same cost. Similar to [Fortz and Thorup, 2002], a hash structure is used to avoid evaluating solutions that were already visited. The procedure stops after 100 iterations.

Algorithm 1 : Local search

Initialization: find an initial solution x by assigning random weights to each link
$\delta = 10\%$
Search: while stopping criterion is not met:
    calculate the cost Φ of solution
    evaluate δ of the neighborhood
    if better solution was found:
        move to better solution
        $\delta = \delta/3$
    else:
        move to solution with same cost
        $\delta = \delta \cdot 2$
End: return best solution found

4.3 Results

Two measures are compared for each solution. The first is the cost of the scenario with the highest cost, or $\max_{r \in R}\{\Phi_r\}$, to see how the solution behaves under the worst case scenario. The second is the average cost of the scenarios, or $\frac{1}{|R|}\sum_{r \in R}\Phi_r$, to see the overall performance of the solution. The results of the experiments are shown on Table 2.

Analyzing the first measure, the *SDDM* overcame the other two only in one instance, and had the worst performance on 5 of 8 instances. It is the least preferable for networks sensible to the worst case. This can be explained by the assumption of a single demand matrix, that ignores the fact that some scenarios are much more difficult than others. The *ROM* have the best result for 5 of 8 instances. It is natural, since the goal of this objective function is minimizing the cost of worst case scenario. For the second measure, again the *SDMM* has the worst performance, having the best results for only 2 of 8 instances. The *MDMM* performed better on 5 of 8 instances, and the *ROM* on 3 of 8. Comparing the first and second set of instances, on the former the *MDMM* won on 6 of 8 instances, and on the latter, *ROM* objective performed better on 6 of 8 instances. This can be explained by the fact that, on the second set of instances, since each scenario represents an hour, the variation between each scenario is bigger, causing the average solution to be less preferable.

Instance	$\max_{r\in R}\{\Phi_r\}$			$\frac{1}{\lvert R\rvert}\sum_{r\in R}\Phi_r$		
	MDMM	*ROM*	*SDMM*	*MDMM*	*ROM*	*SDMM*
abilene-day	1.869	1.869	**1.863**	1.015	1.014	**1.009**
geant-day	**2.908**	**2.908**	2.909	**2.15**	**2.15**	**2.15**
nobel-day	**88843.962**	89226.94	117085.391	**17125.955**	17385.665	21309.589
germany50-day	**8.507**	17.99	17.069	**4.319**	5.629	5.229
abilene-hour	13.619	**13.343**	14.696	7.886	**7.737**	7.801
geant-hour	7.428	**7.37**	7.732	**5.866**	5.968	5.927
nobel-hour	21265.933	**21263.35**	27260.172	6041.295	**5380.226**	5553.071
germany50-hour	6.117	**6.11**	6.273	**3.893**	3.91	3.906

Table 2: The table presents the results of the computational experiments. The first column shows the name of the instance. The next three columns show the value of $\max_{r\in R}\{\Phi_r\}$ for the three models tested. The last three show the value of $\frac{1}{|R|}\sum_{r\in R}\Phi_r$ for the models. The best results for each instance are highlighted.

5 Concluding remarks

This work presented a robust version of the OSPF weight setting problem, taking into consideration the variation on the demands of computer networks. We showed a way to model the robust optimization problem, and solved it with heuristics of the literature. The computational experiments showed that a robust approach for the problem is efficient to achieve solutions that are good on the worst case scenario. Besides, they show that minimizing the average cost of all scenarios may not be the best objective for a robust model, since it can generate solutions that do not perform well for the worst case scenario.

References

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

Ayşegül Altin, Pietro Belotti, and Mustafa Ç. Pınar. OSPF routing with optimal oblivious performance ratio under polyhedral demand uncertainty. *Optimization and Engineering*, 11(3):395–422, November 2009.

Ayşegül Altin, B. Fortz, and Hakan Ümit. Oblivious OSPF routing with weight optimization under polyhedral demand uncertainty. *Networks*, 60(2):132–139, April 2012.

Ayşegül Altın, Bernard Fortz, Mikkel Thorup, and Hakan Ümit. Intra-domain traffic engineering with shortest path routing protocols. *Annals of Operations Research*, 204(1):65–95, 2013.

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

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

L. S. Buriol, L. S Buriol, M. G. C Resende, C. C Ribeiro, and M Thorup. A Hybrid Genetic Algorithm For The Weight Setting Problem In OSPF/IS-IS Routing. *Journal of Combinatorial Optimization*, 6:299 – 333, 2003.

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

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

M. Ericsson, M. G. C Resende, and P. M Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. *JOURNAL OF COMBINATORIAL OPTIMIZATION*, 6:299 – 333, 2002.

B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 2, pages 519–528. IEEE, 2000.

B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, May 2002.

Bernard Fortz. Applications of meta-heuristics to traffic engineering in IP networks. *International Transactions in Operational Research*, 18(2):131–147, March 2011.

Bernard Fortz and Mikkel Thorup. Increasing Internet Capacity Using Local Search. *Computational Optimization and Applications*, 29(1):13–48, October 2004.

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

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

P. Kouvelis and G. Yu. *Robust discrete optimization and its applications*. Springer, 1996.

J. Moy. OSPF specification. RFC 1131, Internet Engineering Task Force, 1989.

S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. In *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007. http://sndlib.zib.de, extended version accepted in Networks, 2009.

Roger Reis, Marcus Ritt, Luciana S. Buriol, and Mauricio G. C. Resende. A biased random-key genetic algorithm for OSPF and DEFT routing to minimize network congestion. *International Transactions in Operational Research*, 18(3):401–423, May 2011.

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