Efficient OSPF Weight Allocation for Intra-domain QoS Optimization*

Pedro Sousa[◇], Miguel Rocha[◇], Miguel Rio[▽], and Paulo Cortez[+]

[◇]Department of Informatics/CCTC, University of Minho, Portugal, email: {pns,mrocha}@di.uminho.pt
[▽]Department of Electronic and Electrical Engineering, University College London, UK, email: m.rio@ee.ucl.ac.uk
[+]Department of Information Systems, University of Minho, 4800-058 Guimarães, Portugal, email: pcortez@dsi.uminho.pt

Abstract. This paper presents a traffic engineering framework able to optimize OSPF weight setting administrative procedures. Using the proposed framework, enhanced *OSPF* configurations are now provided to network administrators in order to effectively improve the QoS performance of the corresponding network domain. The envisaged NP-hard optimization problem is faced resorting to Evolutionary Algorithms, which allocate OSPF weights guided by a bi-objective function. The results presented in this work show that the proposed optimization tool clearly outperforms common weight setting heuristics and, even under unfavorable scenarios, effective QoS improvement is achieved in the network domain.

1 Introduction

The onset of new types of applications and their incremental integration in IP based networks have fostered the development of novel network solutions, aiming at providing end-users with Quality of Service (QoS) support [1]. In the context of a QoS aware networking domain, Internet Service Providers (ISPs) have Service Level Agreements (SLAs) [2] with their clients and with peered ISPs that have to be strictly obeyed in order to avoid financial penalties. To successfully face such requirements, there is an important set of configuration tasks that have to be performed by administrators in order to assure that correct resource provisioning is achieved in the ISP domain. These configuration tasks may vary according to specific QoS provisioning solutions adopted by ISPs and corresponding traffic control mechanisms in place.

Independently of the large set of mechanisms and alternatives that might be in place in any QoS capable infrastructure, there are some components which, by their nature, have crucial importance irrespective of the particular QoS solution adopted. One of such components has the ability to control the data path followed by packets traversing a given Wide Area Network (WAN). In a TCP/IP WAN, consisting of a single administrative domain, there are alternative strategies for this purpose: Intra-domain routing protocols or Multi-Protocol Label Switching (MPLS) [3]. However, the use of MPLS presents some drawbacks when used in the context of packet switching, when compared with the simplicity of some routing protocols. As regards intra-domain routing protocols, the most commonly used today is Open Shortest Path First (OSPF) [4][5]. Since, in OSPF, the link weight setting process is the only way administrators can affect the network behavior, this choice is of crucial importance. Nevertheless, in practice, simple

*The authors wish to thank the Portuguese National Conference of Rectors (CRUP)/British Council Portugal (B-53/05 grant), the Nuffield Foundation (NAL/001136/A grant), the Engineering and Physical Sciences Research Council (EP/522885 grant) and Project SeARCH (Services and Advanced Research Computing with HTC/HPC clusters), funded by FCT.

rules of thumb are typically used in this task, like setting the weights inversely proportional to the link capacity. This approach often leads to sub-optimal network resource utilization. An ideal way to improve the process of OSPF weight setting is to implement traffic engineering. This was the approach taken by Fortz et al [6] where this task was viewed as an optimization problem by defining a cost function that measures the network congestion. The same authors proved that this task is a NP-hard problem and proposed some local search heuristics that compared well with the MPLS model. However, such approach did not accommodate delay based constraints that are also crucial to implement QoS aware networking services in the Internet.

In this paper, Evolutionary Algorithms (EAs) are employed to calculate link-state routing weights, that optimize traffic congestion, while simultaneously complying to specific delay requirements. In this way, the framework proposed in this paper should be viewed as a network management tool which, while focusing only at the OSPF routing level, aims at optimizing the overall QoS performance of a given domain. Based on our experiments, networks using the proposed optimization framework are able to clearly outperform the QoS performance obtained by networks using common OSPF weight setting heuristics. This means that irrespective of the QoS solutions in place, the proposed framework will always be an add-on to improve the QoS level of network domains using OSPF routing mechanisms. The paper is organized as follows: firstly, the problem is defined under the model developed and the EAs designed to tackle this problem are described; the following section presents the experiments and corresponding results; finally, conclusions are drawn and the future work is revealed.

2 Problem Formulation

The network scenario depicted in Figure 1(a) includes a set of network nodes which are interconnected using links with distinct capacities and propagation delays. It is assumed that the ISP can map the clients demands into a matrix (there are several techniques on how to obtain such matrices, e.g. Medina et al [7]). This matrix summarizes, for each source/destination router pair, a given amount of bandwidth and end-to-end delay required to be supported by the ISP. Figure 1(a) shows a scenario involving an individual demand between two network nodes (A and B). Assuming that this demand is mainly expressed in terms of a delay target, then the ISP, in the absence of other traffic, should be able to compute OSPF weights that will result in a data path with the minimum end-to-end delay between A and B (see PATH 2 in Figure 1(a)). In opposition, if no delay requirements are imposed in the demand, and the only constraint between A and B is a given bandwidth requirement (e.g. 90Mbps), then the optimization methods would try to minimize the network congestion and, consequently, assign OSPF weights to force a data path inducing the lowest level of losses in the traffic (PATH 1 in the case of the scenario presented in Figure 1(a)). For these two distinct optimization aims, two distinct sets[1] of OSPF weights are presented in Figure 1(b)(c). It can be observed that the OSPF weights are assigned in order that, after running the Dijkstra algorithm, the shortest paths between nodes A and B are a perfect match of PATH 1 and PATH 2 from Figure 1(a). Note that in a real network scenario, the end-to-end delays of the network paths depicted in Figure 1(a) comprise the link propagation delays and the average queuing delays occurring at each individual node. However, to simplify the selected illustrative scenarios, in the previous example only the link propagation delays were

[1] To simplify, in the selected examples the OSPF weights range from 1 to 3

used to compute the optimal delay constrained path. As discussed later, the proposed optimization framework is also able to integrate the queuing delay component induced at network nodes, if additional technical information is made available by the ISP.

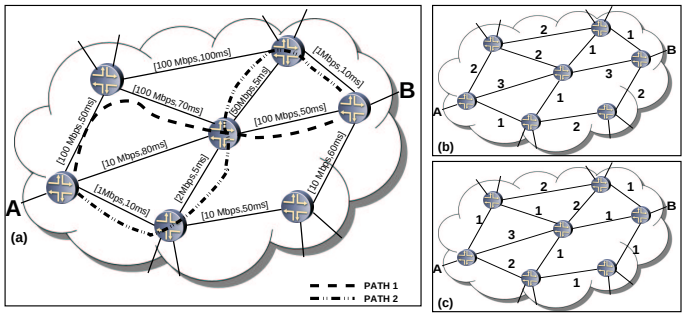


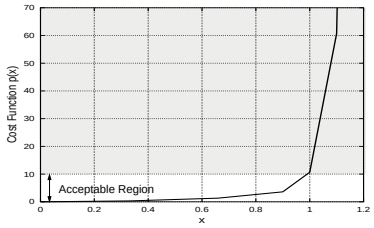
Fig. 1. Example of an ISP network scenario with distinct end-to-end paths between nodes A and B.

Additionally, note that in Figure 1(a) if a given demand has simultaneously bandwidth and delay constraints, it is expected that the OSPF weights set by the optimization algorithms are chosen in order to find a data path representing a tradeoff between the bandwidth and delay metrics. The example of Figure 1(a) is extremely simple, due to the fact that one simple demand was considered in the traffic traversing the network domain. Considering now that each router pair of a given ISP has specific bandwidth and delay demands, it is easy to understand how difficult it is to correctly set OSPF weights using simple heuristics. The proposed optimization framework assumes that the OSPF routing scheme works only with one level of OSPF weights (i.e. one weight per link), which is the currently most common scenario. This assumption represents a difficulty for the optimization model due to the fact that no traffic differentiation is possible at the routing level. In opposition, if the OSPF routing scheme allows the definition of multiple levels of weights, better optimization results might be easily achieved since, at each node, traffic with distinct QoS requirements could be forwarded through distinct routes. Although this work assumes OSPF scenarios with an unique level of weights, which might be considered as more challenging and difficult for the optimization purposes, it is also intended to adapt the proposed optimization model for OSPF schemes considering multiple levels of weights, as will be discussed in the future work section.

The general routing problem [8], that underpins our work, represents routers and transmission links by a set of nodes (N) and a set of arcs (A) in a directed graph $G = (N, A)$. In this model, c_a represents the capacity of each link $a \in A$. Additionally, a demand matrix D is available, where each element d_{st} represents the traffic demand between each pair of nodes s and t from N. Let us assume that, for each arc a, the variable $f_a^{(st)}$ represents how much of the traffic demand between s and t travels over arc a. The total load on each arc a (l_a) can be defined by Eq. (1), while the link utilization rate u_a is given by Eq. (2). It is then possible to define a congestion measure for each link (Φ_a), using a cost function p that has small penalties for values near 0. However, as the values approach the unity it becomes more expensive and exponentially penalizes values above 1 (Figure 2). Given this function, the congestion measure for a given arc can be defined by Eq. (3). Under this framework, it is possible to define a linear programming instance, where the purpose is to set the value of the variables f_a^{st} that

minimize the objective function defined by Eq. (4). The complete formulation can be found in [6]. In the following the optimal solution to this problem is denoted by Φ_{Opt}.

$$l_a = \sum_{(s,t)\in N\times N} f_a^{st} \quad (1) \qquad u_a = \frac{l_a}{c_a} \quad (2) \qquad \Phi_a = p(u_a) \quad (3) \qquad \Phi = \sum_{a\in A} \Phi_a \quad (4)$$



$$p(x) = \begin{cases} x, & x \in [0, 1/3) \\ 3x - 2/3, & x \in [1/3, 2/3) \\ 10x - 16/3, & x \in [2/3, 9/10) \\ 70x - 178/3, & x \in [9/10, 1) \\ 500x - 1468/3, & x \in [1, 11/10) \\ 5000x - 16318/3, & x > 11/10 \end{cases}$$

Fig. 2. Representation and definition of the penalty function $p(x)$.

In OSPF, all arcs are associated with an integer weight. All nodes use these weights in the Dijkstra algorithm [9] to calculate the shortest paths to all other nodes. Each of these paths has a length equal to the sum of its arcs. All the traffic from a given source to a destination travels along the shortest path. If there are two or more paths with the same length, between a given source and a destination, traffic is evenly divided among the arcs in these paths (load balancing) [10]. Let us assume a given solution, i.e. a weight assignment (w), and the corresponding utilization rates on each arc(u_a). In this case, the total routing cost is expressed by Eq. (5), for the loads and corresponding penalties ($\Phi_a(w)$) calculated based on the given OSPF weights. In this way, the OSPF weight setting problem (as defined in [6]) is equivalent to finding the optimal weight values for each link (w_{opt}), in order to minimize the function $\Phi(w)$. The congestion measure can be normalized over distinct topology scenarios, by using a scaling factor defined in [6] (Eq. (6)), where h_{st} is the minimum hop count between nodes s and t.

$$\Phi(w) = \sum_{a\in A} \Phi_a(w) \qquad (5) \qquad\qquad \Phi_{UNCAP} = \sum_{(s,t)\in N\times N} d_{st} h_{st} \qquad (6)$$

Finally, the scaled congestion measure cost is defined as Eq. (7) and the relationships defined in Eq. (8) hold, where $\Phi OptOSPF^*$ is the normalized congestion imposed by the optimal solution to the OSPF weight setting problem. It is important to note that when Φ^* equals 1, all loads are below $1/3$ of the link capacity; on the other hand, when all arcs are exactly full the value of Φ^* is 10 2/3. This value will be considered as a threshold that bounds the acceptable working region of the network.

$$\Phi^*(w) = \frac{\Phi(w)}{\Phi_{UNCAP}} \quad (7) \qquad\qquad 1 \leq \Phi^*_{OPT} \leq \Phi^*_{OptOSPF} \leq 5000 \qquad (8)$$

In order to include other QoS metrics, it was necessary to include delay constraints in this model. Delay requirements were modeled as a matrix DR, that for each pair of nodes $(s,t) \in N \times N$ (where $d_{st} > 0$) gives the delay target for traffic between s and t (denoted by DR_{st}). In a way similar to the congestion model presented before, a cost function was developed to evaluate the delay compliance for each scenario (a given solution defined by the set of weights in the OSPF). This function takes into account the average delay of the traffic between the two nodes (Del_{st}), a value calculated by considering all paths between s and t with minimum cost and averaging the delays in

each. The delay in each path is the sum of the propagation delays in its arcs ($Del_{st,p}$) and queuing delays in the nodes along the path ($Del_{st,q}$). Note that in some network scenarios the latter component might be neglected (e.g. if the propagation delay component has an higher order of magnitude than queuing delays). However, if required, the $Del_{st,q}$ component might be approximated, resorting to queueing theory [11], taking into account the following parameters at each node: the capacity of the corresponding output link (c_a), the link utilization rate (l_a) and more specific parameters such as the mean packet size and the overall queue size associated with the link. The delay compliance ratio for a given pair $(s,t) \in N \times N$ is, therefore, defined by Eq. (9). As before, a penalty for delay compliance can be calculated using function p. So, the γ_{st} function is defined according to Eq. (10). This, in turn, allows the definition of a delay minimization cost function, for a given a set of OSPF weights (w), expressed by Eq. (11). In Eq. (11), the $\gamma_{st}(w)$ values represent the delay penalties for each end-to-end path, given the routes determined by the OSPF weight set w. This function can be normalized dividing the values by the sum of all minimum end-to-end delays[2], as expressed by Eq. (12). It is now possible to define the optimization problem addressed in this work, that is clearly multiobjective. Indeed, given a network represented by a graph G of nodes N and arcs A, a demand matrix D and a delay requirements matrix DR, the aim is to find the set of OSPF weights that simultaneously minimizes the functions $\Phi^*(w)$ and $\gamma^*(w)$.

$$dc_{st} = \frac{Del_{st}}{DR_{st}} \tag{9}$$

$$\gamma_{st} = p(dc_{st}) \tag{10}$$

$$\gamma(w) = \sum_{(s,t)\in N\times N} \gamma_{st}(w) \tag{11}$$

$$\gamma^*(w) = \frac{\gamma(w)}{\sum_{(s,t)\in N\times N} minDel_{st}} \tag{12}$$

$$f(w) = \alpha\Phi^*(w) + (1-\alpha)\gamma^*(w) \tag{13}$$

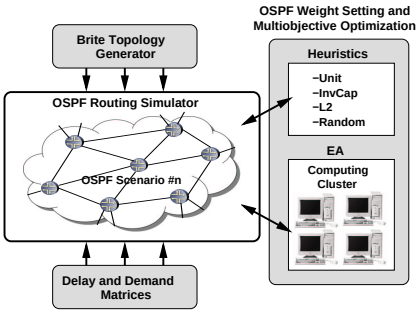


Fig. 3. Platform for OSPF performance evaluation.

In this work, Evolutionary Algorithms (EAs) are proposed to address the OSPF weight setting problem, both by considering the multiobjective formulation, or by taking each of the two distinct aims separately. In the proposed EA, each individual encodes a solution as a vector of integer values, where each value (gene) corresponds to the weight of an arc in the network (the values range from 1 to w_{max}). Therefore, the size of the individual equals the number of arcs in the graph (links in the network). The individuals in the initial population are randomly generated, with the arc weights taken from a uniform distribution in the allowed range. In order to create new solutions, several reproduction operators were used, more specifically two mutation and two crossover operators: **Random Mutation** - replaces a given gene by a new randomly generated value, within the allowed range $[1, w_{max}]$; **Incremental/decremental Mutation** - replaces a given gene by the next or by the previous value (with equal probabilities) and constrained to respect the range of allowed values; **Uniform crossover**

[2] For each pair of nodes the minimum end-to-end delay, $minDel_{st}$, is calculated as the delay of the path with minimum possible overall delay

and **Two-point crossover** - two standard crossover operators, applied in the traditional way [12]. All operators have equal probabilities in generating new solutions. When a single objective is considered the fitness of an individual (encoding weight set w) is calculated using functions $\Phi^*(w)$ for congestion and $\gamma^*(w)$ for delays. For multiobjective optimization a quite simple scheme was devised. The fitness ($f(w)$) of the individual is, in this case, derived by Eq. (13). This scheme, although simple, can be effective since both cost functions are normalized in the same range and use a similar penalization function.

3 Experiments and Results

The conceptual model of the experimental platform that was implemented and used in this work for results evaluation is presented in Figure 3. For this purpose, a set of 12 networks was generated by using the Brite topology generator [13], varying the number of nodes ($N = 30, 50, 80, 100$) and the average degree of each node ($m = 2, 3, 4$). This resulted in networks ranging from 57 to 390 links (graph edges). The link bandwidth (capacity) was generated by an uniform distribution between 1 and 10 Gbits/s. The network was generated using the Barabasi-Albert model, using a heavy-tail distribution and an incremental grow type (parameters HS and LS were set to 1000 and 100, respectively). In the generated examples, the propagation delays were assumed as the major component of the end-to-end delay of the networks paths. Thus, the network queuing delays at each network node were not considered (i.e. $Del_{st,q} = 0$)[3]. For each of the twelve network instances a set of three distinct instances of D and DR were created. A parameter (D_p) was considered which determined the expected mean of the congestion in each link (u_a) (values for D_p in the experiments were 0.1, 0.2 and 0.3). Although the values used for D_p seem, at a first glance, to be low they represent averages of the links and don't take the network topology into account. So, even with these values, there is still a high probability that a number of links get high congestions. Since the penalty function heavily penalizes congested links, higher values of this parameter would result in networks that would not be realistic, since they would not be capable of reaching an acceptable behaviour. For the DR matrices, the strategy was to calculate the average of the minimum possible delays, over all pairs of nodes. A parameter (DR_p) was considered, representing a multiplier applied to the previous value to get the matrix DR (values for DR_p in the experiments were 3, 4 and 5). Overall, a set of $12 \times 3 \times 3 = 108$ instances of the optimization problem were considered.

A number of heuristic methods was implemented, to provide a comparison with the results obtained by the EAs: **Unit** - sets all arc weights to 1 (one); **InvCap** - sets arc weights to a value inversely proportional to the capacity of the link; **L2** - sets arc weights to a value proportional to the physical Euclidean distance (L2 norm) of the link; **Random** - a number of randomly generated solutions are analyzed and the best is selected, where the number of solutions considered is always equal to the number of solutions evaluated by the EA in each problem. The proposed EA, the heuristics and the OSPF routing simulator were implemented by the authors using the Java programming language. The EA was run for a number of generations ranging from 1000 to 6000, a value that was incremented proportionally to the number of variables optimized by the EA. The running times varied from a few minutes in the small networks, to a few hours

[3] However, and as previously mentioned, if additional parameters are provided by the ISP this parameter can be also considered by the proposed optimization framework.

in the larger ones. So, in order to perform all the tests, a computing cluster with 46 dual Xeon nodes was used. The population size was kept in 100 and the w_{max} was set to 20. Since the EA and the Random heuristic are stochastic methods, R runs were executed in each case (R was set to 10 in the experiments). For a better understanding, the results are grouped into three sets according to the cost function used. The first two consider single objective cost functions, for the optimization of congestion and delays respectively. These are used mainly as a basis for the comparison with the results obtained with the last group, that presents the results using the multiobjective cost function. In all figures presented in this section the data was plotted in a logarithmic scale, given the exponential nature of the penalty function adopted. Since the number of performed experiments is quite high, it was decided in the following sections to present all the results for just one of the networks (out of the 12) in order to explain the experimental methodology, and then to show some aggregate results to draw conclusions.

3.1 Congestion

Table 1 shows the results for the optimization of the congestion, for one of the networks (with 100 nodes and 197 links). In this table, the first column represents the demand generation parameter D_p (higher values for this parameter indicate higher mean demands, thus harder optimization problems). The remaining columns indicate the congestion measure ($\Phi^*(w)$) for the best solution (w) obtained by each of the methods considered in this study. In the case of the EAs and Random heuristic the values represent the mean value of the results obtained in the set of runs. Table 2 shows the results for all the 12 available networks, averaged by the demands levels (value of D_p), including in the last line the overall mean value for all problem instances. It is clear that the results for all the methods get worse with the increase of D_p, as would be expected. The comparison between the methods shows an impressive superiority of the EA when compared to the heuristic methods. In fact, the EA achieves solutions which manage a very reasonable behavior in all scenarios (worse case is 1.49), while the other heuristics manage very poorly. Even $InvCap$, an heuristic quite used in practice, gets poor results when D_p is 0.2 or 0.3 (Figure 4)[4], which means that the optimization with the EAs assures good network behavior in scenarios where demands are at least 200% larger than the ones where $InvCap$ would assure similar levels of congestion.

Table 1. Results for the optimization of congestion (Φ^*) in one example network with 100 nodes and 197 links.

D_p	Unit	L2	InvCap	Random	EA
0.1	3.62	190.67	1.68	12.05	1.02
0.2	136.75	658.66	135.07	280.27	1.25
0.3	264.02	874.89	488.53	551.65	1.49

Table 2. Results for the optimization of congestion (Φ^*) - averaged results by demand levels

D_p	Unit	L2	InvCap	Random	EA
0.1	8.03	215.94	1.50	75.75	1.02
0.2	99.96	771.87	57.70	498.74	1.18
0.3	227.30	1288.56	326.33	892.87	1.73
Overall	111.76	758.79	128.51	489.12	1.31

Figure 5, on the other hand, represents the results for congestion, but aggregated by the number of arcs (links). It is clear in both cases that the results obtained by the EAs are quite scalable, since the quality levels are not affected by the number of nodes or edges in the network graph. The results obtained in this section show that the EA makes

[4] In the figures the white area represents the acceptable working region whereas an increasing level of gray is used to identify working regions with increasing levels of service degradation.

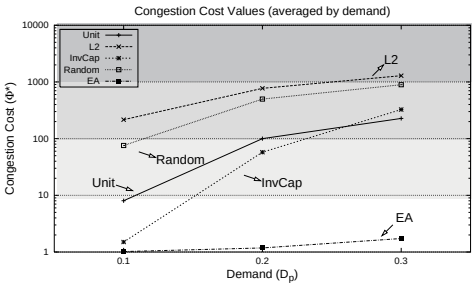


Fig. 4. Results obtained by the different methods in congestion optimization (averaged by D_p).

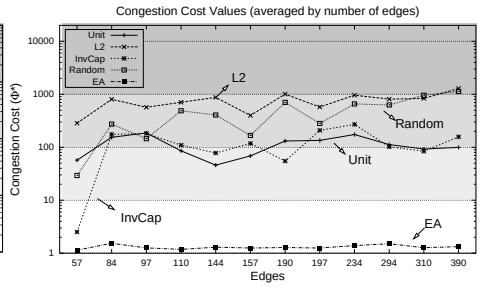


Fig. 5. Results obtained by the different methods in congestion optimization (averaged by the number of network links).

an effective method for the optimization of OSPF weights, in order to minimize the congestion of the network. These results confirm the findings of other single objective OSPF optimization works (e.g. Ericsson et al [14]), although a precise comparison of the approaches is impossible since the data is not available.

3.2 Delays

Regarding the optimization of delays (cost function γ^*), a similar methodology was adopted. Indeed, in Table 3 the results for the same example network are shown. The methods used in the optimization are the same as in the previous section. In this case, the first column represents the parameter used for the generation of delay requirements (DR_p). On the other hand, Table 4 and Figure 6 represent the results obtained for the delay optimization averaged by the parameter used in the generation of delays requirements (DR_p). In this case, the results of all methods improve when the value is higher, since higher delay requirements are easier to comply.

Table 3. Results for the optimization of delays (function γ^*) in one example network with 100 nodes and 197 links.

DR_p	Unit	L2	InvCap	Random	EA
3	13.50	1.38	201.62	4.36	1.38
4	2.00	1.13	18.33	1.82	1.13
5	1.47	1.04	3.62	1.54	1.04

Table 4. Results for the optimization of delays (γ^*)- averaged results by the delay requirements parameter (DR_p)

DR_p	Unit	L2	InvCap	Random	EA
3	152.37	2.94	577.94	156.62	2.85
4	28.78	1.25	158.85	24.35	1.25
5	6.59	1.10	44.13	4.29	1.10
Overall	62.58	1.76	260.30	61.75	1.73

The relative performance of each method shows a good behavior of the EA, as before, but now there is a simpler heuristic method - the L2 - that achieves very similar results. This is not a surprise, since in the proposed model only propagation delays were considered and these are proportional to the length of each link. The L2 heuristic considers the OSPF weights to be proportional to the arc length, which means they are also directly proportional to the delays. So, it is clear that the L2 heuristic exhibits a near-optimal behavior in this problem. It is important to notice that in the context of network management, the minimization of propagation delays, disregarding congestion, is typically not an optimization aim by itself. So, the results in this section will be used mainly as a basis for comparison with the results of multiobjective optimization. As before, the results for the delay minimization are also shown aggregated by the number of links (Figure 7). The scalability of both L2 and the EAs prevails in these results.

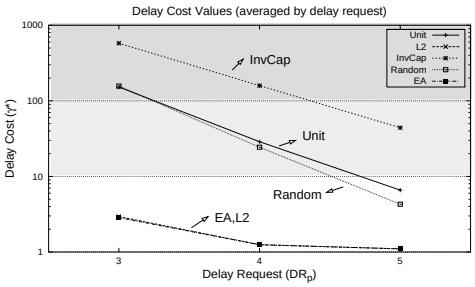


Fig. 6. Results obtained by the different methods in delay optimization (averaged by DR_p).

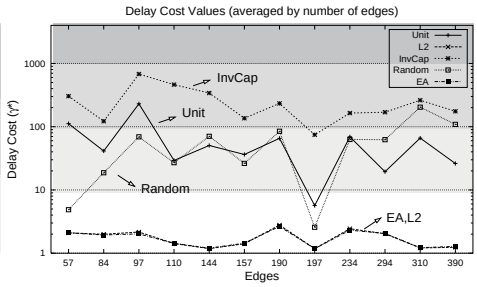


Fig. 7. The results obtained by the different methods in delay optimization (averaged by the number of links).

3.3 Multiobjective optimization

From the set of methods discussed before, only the EA and the Random heuristic can be used to perform multiobjective optimization by considering the optimization of function f (Equation 13) as the aim. In all other heuristic methods, the solution is built disregarding the cost function, so the results for multiobjective optimization can be pasted from the ones obtained in the previous sub-sections. The results of both EAs and Random methods are presented in terms of the values for the two objective functions (Φ^* and γ^*), since the value of f for these solutions can be easily obtained and is not relevant to the analysis (it does not represent any real measure for the network behavior). Three distinct values for α will be tested: 0.25, 0.5 and 0.75. The value of 0.5 considers each aim to be of equal importance, while the 0.25 favours the minimization of delays and 0.75 will give more weight to congestion. Table 5 represents the results obtained in the example network, for the the multiobjective optimization obtained by the EAs and Random heuristics. In this table only the results for $\alpha = 0.5$ are shown. The methodology for the other values of α is essentially the same. In the table, the first two columns represent the parameters for demand and delay requirements; the next two indicate the results for the Random heuristic in both aims and, finally, the last two give the results of the EA for both congestion and delay, each with an extra information indicating the percentage by which this results exceed the ones obtained by the corresponding EA under with a single objective cost function.

Table 5. Results for multiobjective optimization (α is kept at 0.5) in one example network with 100 nodes and 197 links.

D_p	DR_p	**Random**		**EA**	
		Φ^*	γ^*	Φ^* (%)	γ^* (%)
0.1	3	27.36	39.97	1.14 (11.4%)	1.52 (10.2%)
0.1	4	7.22	16.06	1.09 (6.9%)	1.26 (11.8%)
0.1	5	8.82	2.28	1.08 (6.1%)	1.13 (8.9%)
0.2	3	356.25	29.42	1.47 (17.4%)	1.75 (26.2%)
0.2	4	274.06	2.37	1.40 (11.9%)	1.42 (25.9%)
0.2	5	339.06	1.96	1.38 (9.8%)	1.29 (23.7%)
0.3	3	587.51	48.72	1.76 (18.4%)	2.04 (47.8%)
0.3	4	495.32	7.08	1.61 (8.2%)	1.56 (38.4%)
0.3	5	601.00	2.34	1.56 (5.0%)	1.37 (31.3%)

Table 6. Overall results for the multiobjective optimization - averaged by α

α	**Random**		**EA**	
	Φ^*	γ^*	Φ^* (%)	γ^* (%)
0.25	544.47	107.99	2.02 (47.2%)	2.33 (32.5%)
0.5	506.45	130.81	1.68 (25.7%)	2.49 (43.8%)
0.75	468.04	175.82	1.61 (19.5%)	2.92 (69.5%)

Table 7. Results for the multiobjective optimization - averaged by the number of nodes.

Node	**Random**		**EA**	
	Φ^*	γ^*	Φ^* (%)	γ^* (%)
30	283.32	74.77	1.58 (19.9%)	2.25 (24.3%)
50	442.16	165.63	1.78 (36.0%)	2.96 (51.9%)
80	619.14	170.75	1.62 (22.8%)	2.37 (42.7%)
100	681.17	112.09	1.75 (24.3%)	2.38 (56.2%)

In Table 6 the results obtained were aggregated averaging by the parameter α. The results shown in this table make clear its effect, once it is possible to observe different

trade-offs between the two objectives. Indeed, when α increases the results on congestion improve, while the reverse happens to the delay minimization. The intermediate value of α (0.5) provides a good compromise between the two objectives. In this case, the overall results show that, in average, there is a 25% decrease in the congestion performance and around 44% in the delays minimization, both when comparing to single objective optimization. These values are quite good and, in conjunction with the average values for both cost functions, indicate an acceptable performance and, therefore, a good behavior of the network. Table 8 shows the results aggregated averaging by the demand level (D_p) and also by α. It is clear that when the problem gets harder in terms of congestion, both optimization aims are affected the previous. However, even in the worst case (when D_p equals 0.3) the network still manages an acceptable behavior. It is important to notice that in this scenario, and even when the D_p equals 0.2, all heuristics behave quite badly. A similar picture is found looking at Table 9, where the results are averaged by the delay requirement parameter DR_p. In fact, with the increase of DR_p the results improve on both aims, both in absolute terms and considering the percentage of deviation from single objective optimization. Still, and as before, the results are quite acceptable in terms of network behaviour and the deviation from single objective results are within reasonable ranges.

Table 8. Multiobjective optimization - averaged by D_p

α	D	Random		EA	
		Φ^*	γ^*	Φ^* (%)	γ^* (%)
0.25	0.1	110.51	89.76	1.28 (25.4%)	1.85 (7.0%)
	0.2	544,74	107.08	1.64 (39.9%)	2.19 (24.6%)
	0.3	978.16	127.11	3.15 (76.3%)	2.95 (66.0%)
0.5	0.1	88.00	106.79	1.17 (14.5%)	1.92 (12.8%)
	0.2	481.50	136.68	1.47 (25.1%)	2.32 (35.2%)
	0.3	949.85	148.96	2.41 (37.5%)	3.23 (83.3%)
0.75	0.1	73.92	142.41	1.10 (8.3%)	2.12 (25.1%)
	0.2	469.42	180.58	1.35 (14.9%)	2.58 (53.6%)
	0.3	914.79	204.49	2.38 (35.1%)	4.05 (129.9%)

Table 9. Multiobjective optimization - averaged by DR_p.

α	DR	Random		EA	
		Φ^*	γ^*	Φ^* (%)	γ^* (%)
0.25	3	616.36	246.07	2.63 (80.8%)	4.03 (46.2%)
	4	536.07	60.37	1.77 (34.2%)	1.60 (28.0%)
	5	480.98	17.52	1.67 (26.5%)	1.36 (23.4%)
0.5	3	535.28	283.16	1.95 (42.8%)	4.22 (55.2%)
	4	505.69	82.04	1.59 (20.3%)	1.78 (41.8%)
	5	478.37	27.23	1.51 (14.2%)	1.48 (34.4%)
0.75	3	506.84	372.04	1.89 (37.1%)	5.05 (94.1%)
	4	468.14	116.96	1.48 (11.6%)	2.03 (62.3%)
	5	483.14	38.48	1.46 (9.7%)	1.68 (52.2%)

Table 7, on the other hand, confirms the good scalability properties of the EA. In fact, and as seen in the previous sections for both congestion and delay optimization, the results are almost constant for the different network sizes (in this case, measured by the number of nodes). A different view is offered by Figures 8 and 9 where the results are plotted with the two objective functions in each axis. The former shows the results averaged by the demand levels and the latter by the delay requirements parameter. In both cases the value considered for α was 0.5, although the overall view for different values of this parameter would be very similar. In these graphs, the good overall network behavior of the solutions provided by the EA is clearly visible, both in absolute terms, regarding the network behavior in terms of congestion and delays, and when compared to all other alternative methods. In fact, it is easy to see that no single heuristic is capable of acceptable results in both aims simultaneously. L2 behaves well in the delay minimization but fails completely in congestion; InvCap is better on congestion (although in a very limited range) but fails completely in the delays. EAs, on the other hand, are capable of a good compromise between both optimization targets.

Finally, Figures 10 and 11 show similar graphs but considering only the EAs and plotting the results for different values of α. In the figures the trade-offs between the two objectives are clear. Regarding to Figure 10 the obtained delay and congestion cost

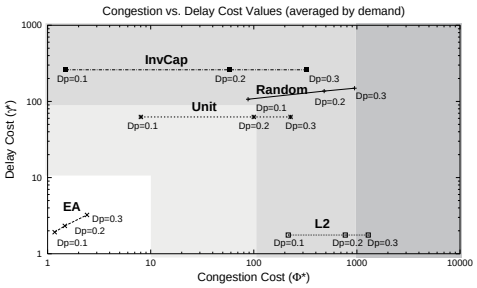


Fig. 8. Results obtained by the different methods in the multiobjective optimization (averaged by D_p for $\alpha = 0.5$).

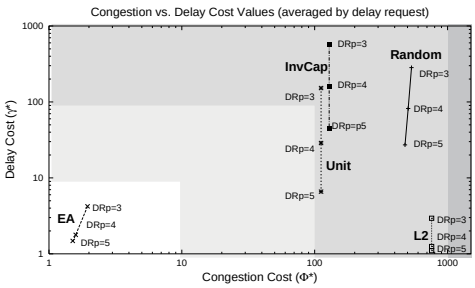


Fig. 9. Results obtained by the different methods in the multiobjective optimization (averaged by DR_p for $\alpha = 0.5$).

values are averaged for distinct values of traffic demands ($D_p = 0.1, 0.2$ and 0.3). Moreover, three distinct lines are plotted, each one representing the results obtained assuming distinct values of α (0.25, 0.5 and 0.75). As observed in Figure 10, the results show the correctness of the proposed optimization model. In fact, within a given demand value, the plots are shifted towards the upper left region of the graph as the α value increases. This behaviour corroborates the optimization model underpinning concept, in which higher values of α lead to an improvement in the congestion metric but, at the same time, a penalization in the delay performance. The results plotted in Figure 11 show the obtained delay and congestion cost values averaged now for distinct values of the delay requests ($DR_p = 3, 4$ and 5). As in the case of Figure 10, the results clearly show the correctness of the system dynamics, as the delay and congestion performance of distinct experimental scenarios is controlled by the α parameter. As observed, network configuration assuming lower values for α achieve a better delay performance. As obvious, and as in the case of Figure 11, this improvement is achieved at the cost of a congestion penalization of the system.

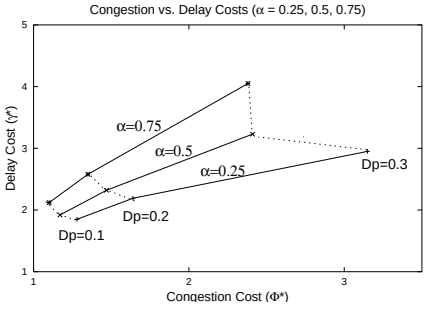


Fig. 10. Results obtained by the EAs (averaged by D_p).

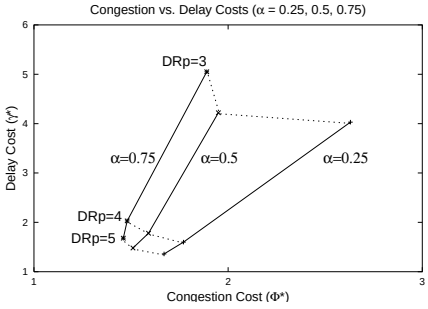


Fig. 11. Results obtained by the EAs (averaged by DR_p).

4 Conclusions and further work

This work presented an optimization scheme based on Evolutionary Algorithms with an integer representation for the purpose of multiobjective routing in the Internet. To achieve this aim, an analytical model was developed allowing the performance evaluation of several QoS constrained OSPF routing scenarios of a given ISP. Resorting to

a large set of network topology configurations, each one constrained by several bandwidth and end-to-end delay requirements, it was shown that the proposed EAs were able to provide OSPF weight settings able to satisfy the users demands. The performance of EAs was compared with several heuristics, typically used by network administrators, clearly showing the superiority of the proposed optimization approach.

The research results presented in this work give ground to the idea that it is possible to develop network management tools which automatically provide network administrators with optimal configurations for a given network topology and corresponding sets of QoS demands. In this way, ISP resource provisioning management tasks can be now simplified while providing better results and, consequently, strong financial improvements can be achieved by organizations using the proposed OSPF optimization scheme. Although a simple weighting method was used to face the multiobjective nature of the problem, the results were of high quality. Nevertheless, the consideration of more specific EAs to handle this class of problems [15][16] will be taken into account in future work. Another topic for future work is the integration of distinct classes of QoS demands in the proposed optimization model. On this topic, the Internet Engineering Task Force (IETF) has proposed standards on Multi-topology Routing aiming at providing different paths for different types of traffic [17]. The final objective is to adapt the proposed optimization model to deal with OSPF routing schemes having the ability to use multiple levels of weights.

References

1. Zheng Wang. *Internet QoS: Architectures and Mechanisms for Quality of Service*. Morgan Kaufmann Publishers, 2001.
2. D. Verma. *Supporting Service Level Agreement on IP Networks*. McMillan Publishing, 1999.
3. B. Davie and Y. Rekhter. *MPLS: Multiprotocol Label Switching Technology and Applications*. Morgan Kaufmann, USA, 2000.
4. J. Moy. RFC 2328: OSPF version 2, April 1998.
5. T.M. ThomasII. *OSPF Network Design Solutions*. Cisco Press, 1998.
6. B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proceedings of IEEE INFOCOM*, pages 519–528, 2000.
7. A. Medina et al. Traffic matriz estimation: Existing techniques and new directions. *Computer Communication Review*, 32(4):161–176, 2002.
8. Ravindra et al. *Network Flows*. Prentice Hall, 1993.
9. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(269-271), 1959.
10. J. Moy. *OSPF, Anatomy of an Internet Routing Protocol*. Addison Wesley, 1998.
11. G. Bolch et al. *Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications*. Jhon Wiley and Sons INC., 1998.
12. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, USA, third edition, 1996.
13. A. Medina et al. BRITE: Universal Topology Generation from a User's Perspective. Technical Report 2001-003, January 2001.
14. M. Ericsson, M.G.C. Resende, and P.M. Pardalos. A Genetic Algorithm for the Weight Setting Problem in OSPF Routing. *J. of Combinatorial Optimization*, 6:299–333, 2002.
15. C.M. Fonseca and P.J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.
16. C.A. Coello Coello. *Recent Trends in Evolutionary Multiobjective Optimization*, pages 7–32. Springer-Verlag, London, 2005.
17. P. Psenak et al. Multi-topology (mt) routing in ospf (internet draft), January 2006.