View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Universidade do Minho: RepositoriUM

Class-Based OSPF Traffic Engineering Inspired on Evolutionary Computation

Pedro Sousa[1], Miguel Rocha[1], Miguel Rio[2], and Paulo Cortez[3]

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

Abstract. This paper proposes a novel traffic engineering framework able to automatically provide near-optimal OSPF routing configurations for QoS constrained scenarios. Within this purpose, this work defines a mathematical model able to measure the QoS compliance in a class-based networking domain. Based on such model, the NP-hard optimization problem of OSPF weight setting is faced resorting to Evolutionary Algorithms. The presented results show that, independently of other QoS aware mechanisms that might be in place, the proposed framework is able to improve the QoS level of a given domain only taking into account the direct influence of the routing component of the network. The devised optimization tool is able to optimize OSPF weight configurations in scenarios either considering a single level of link weights or using multiple levels of weights (one for each class) in multi-topology routing scenarios.

1 Introduction

The integration of new types of applications in TCP/IP based networks has fostered the development of several solutions to provide QoS (Quality of Service) [1] support to end-users and corresponding applications in place. In this perspective, ISPs (Internet Service Providers) have Service Level Agreements (SLAs) [2] with end-users and with other peered ISPs that should be obeyed. However, there is not an unique solution to create a QoS aware networking domain and, in general, any solution requires a number of components working together. Independently of specific QoS solutions adopted in a given network domain, there are a set of components which, by their nature, have a major influence in the QoS performance of the network. One example of such components is the routing mechanism that is used in a given domain.

The research efforts presented in this paper focus on the most commonly used intra-domain routing protocol, the Open Shortest Path First (OSPF) [3,4], trying to devise a traffic engineering framework able to provide network managers with near-optimal OSPF link weight configurations. To accomplish this goals, this work will follow the traffic engineering perspective of previous works (e.g. [5]) assuming the existence of a demand matrix associated with the network (there are several alternatives to estimate such matrices, see [6] [7]). In practice, this matrix represents an estimation of the traffic demands between each source/destination router pair of the network domain (e.g. an

© Springer-Verlag Berlin Heidelberg 2007

ingress/egress node pair). Based on such information, the aim is to devise optimal OSPF link weight configurations which optimize a given objective function, a process which is usually viewed as a NP-hard optimization problem. As result, the main objective of the current proposal is to devise a traffic engineering framework able to automatically provide near-optimal OSPF configurations to network administrators. This main objective is supported by several innovative aspects of the proposed framework which are effective contributions when compared with previous work in the area (e.g. [5]): *(i)* extend previous models to tackle multiconstrained QoS optimization; *(ii)* develop an optimization framework in order to support multiservice networks based on the Class of Service paradigm; *(iii)* allow for versatile multiconstrained optimization based on an unique level of OSPF weights or resorting to scenarios using multiple levels of OSPF weights (as proposed by IETF in [8]) and *(iv)* achieve near-optimal OSPF configuration resorting to the field of Evolutionary Computation to improve the network performance.

In this context, 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. This does not hinder that other complementary QoS aware mechanisms might be used by network administrators, either to improve the network performance or to provide more strict QoS guarantees. However, the key point is that, based on our experiments, class-based networks using the proposed optimization framework are able to clearly outperform the QoS performance obtained by networks using common OSPF weight setting heuristics.

2 Problem Description

In the example of Figure 1 several network nodes are interconnected by links with distinct capacities and propagation delays representing a given ISP networking domain. Lets assume that the ISP resorts to specific techniques in order to have an estimate of the clients overall demands. In a network traffic engineering perspective such information is usually modeled and viewed as a *demand matrix* which summarizes for each source/destination router pair a given amount of resources required to be supported by the ISP. To obtain such information the ISP may resort to techniques having distinct

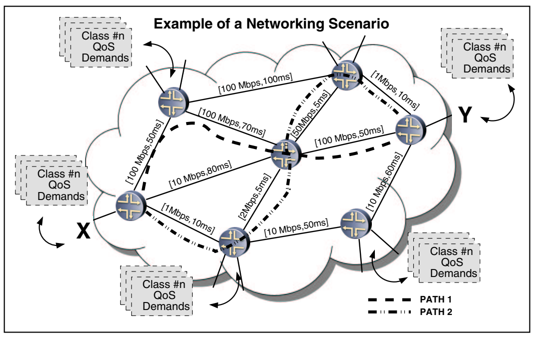

Fig. 1. Example of a network scenario with distinct end-to-end paths between nodes X and Y

levels of accuracy and requiring different computational efforts [6]. Based on such information, and taking into account that the routing process might have an high influence in the QoS performance of a given domain, it is possible to optimize the OSPF weight setting process in order to obtain more efficient configurations of the network. To illustrate such concepts lets assume the ISP network domain example depicted in Figure 1. Lets also assume that after studying the network behaviour, the ISP can have an estimate of the client demands and is able to map such values (demands) to matrices summarizing for each source/destination router pair (e.g. an ingress/egress router pair) a given amount of bandwidth and/or end-to-end delay required to be support by the ISP. Based on that, the optimization methods of OSPF weight setting will try to find configurations aiming at maximizing the overall QoS performance of the network.

Figure 1 shows a very simple scenario involving an individual demand between two network nodes (X and Y). Assuming that this demand is expressed as 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). In opposition, if the demand is mainly expressed in terms of a delay target[1], 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 propagation delays between X and Y (see PATH 2 in Figure 1). Moreover, in Figure 1 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 (i.e. in a multiobjective perspective). Also note that if the network domain represented in Figure 1 is also viewed as a multiservice domain, e.g. supported by a class-based IP infrastructure, then each router pair of a given ISP might have also specific per-class bandwidth and delay demands. It is easy now to understand the NP-hard nature of the problem and how difficult it is to correctly set OSPF weights using simple heuristics. The proposed optimization framework assumes that the OSPF routing scheme is able to operate with one level of OSPF weights (i.e. one weight per link), which is the currently most common scenario, but is also able to provide near-optimal solutions when multiple levels of OSPF weights are used in the network. By this way, an additional feature of the proposed optimization model is the ability to assess the QoS improvements obtained when the network domain migrates to a multi-topology routing perspective, allowing for class-based QoS routing.

3 Mathematical Model

The mathematical model used in this work represents routers and links by a set of nodes (N) and arcs (A) in a directed graph $G = (N, A)$ [9]. In this model, c_a represents the capacity of each link $a \in A$. A demand matrix D_c is available for each class c $(c \in C)$, where each element d^c_{st} represents the demand of traffic from class c, between nodes s and t. For each arc a, $f^c_{st,a}$ represents how much of the traffic demand from class c between s and t travels over arc a. The total load on each arc a for class c

[1] e.g. if a large part of the traffic crossing the domain through nodes X and Y is high delay sensitive.

(l_a^c) can be defined as in Eq. (1) and the total load in arc a (l_a) is therefore given by: $l_a = \sum_{c \in C} l_a^c$. The link utilization rate u_a is given by: $u_a = \frac{l_a}{c_a}$. It is then possible to define a congestion measure for each link ($\Phi_a = p(u_a)$), using a penalty function p that has small values near 0, but as the values approach the unity it becomes more expensive and exponentially penalizes values above 1 [5]. To obtain the penalty related to each class, this value is weighted and Φ_a^c can be obtained as in Eq. (2).

$$l_a^c = \sum_{(s,t) \in N \times N} f_{st,a}^c \quad (1) \qquad \Phi_a^c = \Phi_a \frac{l_a^c}{l_a} \quad (2) \qquad \gamma_c(w) = \sum_{(s,t) \in N \times N} \gamma_{st}^c(w) \quad (3)$$

In *OSPF*, all arcs have an integer weight and every node uses these weights in the Dijkstra algorithm [10] to calculate the shortest paths to all other nodes in the network. 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, traffic is evenly divided among the arcs in these paths (load balancing) [11]. Let us assume a given solution, a weight assignment (w), and the corresponding loads and utilization rates on each arc. In this case, the total routing cost for each class is expressed by $\Phi_c(w) = \sum_{a \in A} \Phi_a^c(w)$ for the loads and corresponding penalties calculated based on the given OSPF weights w. The congestion measures can be normalized (Φ_c^*) over distinct topology scenarios and its value is in the range [1,5000]. It is important to note that when Φ_c^* equals 1, all loads are below $1/3$ of the link capacity; in the case when all arcs are exactly full, the value of Φ_c^* is $10\frac{2}{3}$. This value will be considered as a threshold that bounds the acceptable working region of the network. As explained, it is also useful to include delay constraints in this model. Delay requirements were modeled as a matrix DR_c (one per each class c), that for each pair of nodes $(s,t) \in N \times N$ gives the delay target for traffic of class c between s and t (denoted by DR_{st}^c). In a way similar to the model presented before, a cost function was developed to evaluate the delay compliance for a solution, that takes into account the average delay of the class c traffic between the two nodes (Del_{st}^c), 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 ($Del_{st,p}^c$) and queuing delays in the nodes along the path ($Del_{st,q}^c$). 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 queuing theory [12], taking into account the following parameters at each node: the capacity of the corresponding output link (c_a), the classes link utilization rates (l_a^c) and more specific technical information such as the type of scheduling mechanisms used in the network nodes and corresponding parameter and queue size configurations.

Given this framework, the *delay compliance ratio* for a given pair $(s,t) \in N \times N$ and class c is, therefore, defined as $dc_{st}^c = \frac{Del_{st}^c}{DR_{st}^c}$. A penalty for delay compliance can be calculated using function p. The γ_{st}^c function is defined according to $\gamma_{st}^c = p(dc_{st}^c)$. This allows the definition of a delay cost function, given a set of *OSPF* weights (w), where the $\gamma_{st}^c(w)$ values represent the delay penalties for each end-to-end path, given the routes determined by the *OSPF* weight set w (see Eq. (3)). This function can be normalized dividing the values by the sum of all minimum end-to-end delays to reach the value of $\gamma_c^*(w)$ (for each pair of nodes the minimum end-to-end delay is calculated as the delay of the path with minimum possible overall delay). It is now possible to define the optimization problem addressed in this work. Indeed, given a network represented by a graph G, the demand matrices D_c and the delay requirements matrix's DR_c, the aim is to find the set of *OSPF* weights w that simultaneously minimize the functions $\Phi^*_c(w)$ and $\gamma^*_c(w)$, for $c \in C$. This is a multi-objective optimization problem and a quite simple scheme was devised to define an overall cost function, where the cost of the solution is given by Eq. (4) where $\sum_{c \in C}(\alpha_c + \beta_c) = 1$. This scheme, although simple, can be effective since all cost functions are normalized in the same range. The previous problem formulation considers a single *OSPF* weight set that is applied to all the traffic. An alternative is to consider that each class has its own distinct weight set (represented by w_c), as in Eq. (5).

$$f(w) = \sum_{c \in C}(\alpha_c \Phi^*_c(w) + \beta_c \gamma^*_c(w)) \quad (4) \qquad f(w) = \sum_{c \in C}(\alpha_c \Phi^*_c(w_c) + \beta_c \gamma^*_c(w_c)) \quad (5)$$

4 Evolutionary Algorithms and Heuristics for OSPF Setting

In order to improve the quality of the OSPF configurations this work resorts to *Evolutionary Algorithms (EAs)*. 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 vector equals the number of arcs in the graph (links in the network). In the case of multiple sets of weights (one per each class), the size of the solution is given by the number of classes multiplied by the number of links. The sets of weights are, in this case, still encoded in a single linear vector, that represents the concatenation of the individual sets of weights. Therefore, the *EA* is similar in both situations varying only in the decoding process. 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* and *Two-point crossover*, two standard crossover operators, applied in the traditional way [13]. All operators have equal probabilities in generating new solutions. The selection procedure is done by converting the fitness value into a linear ranking in the population, and then applying a roulette wheel scheme. In each generation, 50% of the individuals are kept from the previous generation, and 50% are bred by the application of the genetic operators.

In order to assess the order of magnitude of the improvements obtained by the proposed framework a number of traditional weight setting heuristic methods was also implemented [5], to provide a comparison[2] with the results obtained by the *EA*, namely:

[2] The results of the heuristics are only compared with the EAs results for scenarios with a single level of weights. For multiple levels of weights only the EAs results will be plotted to assess the improvement obtained in the network QoS.

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.

5 Experimental Framework and Results

The experimental platform that was used in this work is presented in Figure 2. In order to evaluate the effectiveness of the proposed *EAs*, a number of experiments was conducted. For this purpose, a set of 12 networks was generated by using the Brite topology generator [14], varying the number of nodes ($N = 30, 50, 80, 100$) and the average degree of each node ($m = 2, 3, 4$). This resulted in 12 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. $Del^c_{st,q} = 0$).

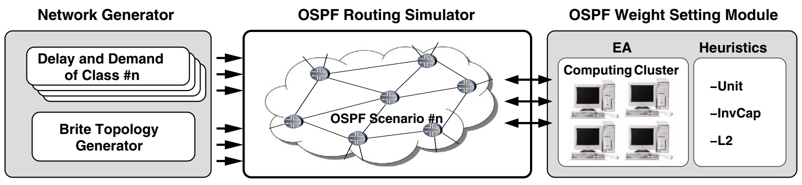

Fig. 2. Experimental platform for OSPF performance evaluation

Next, for each network, the overall demand matrices (D) were generated. For each of the 12 instances a set of three distinct instances of D 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). In the experiments a scenario with two classes was considered. Class 1 was defined as a class with an average of 75% of the overall traffic and class 2 with 25%. In each origin/destination pair, the demand from D was split between the two classes. The proportion of traffic assigned to class 1 (d^1_{st}) was generated, for each case, from a uniform distribution within the range $P_1(1 \pm h)$, where P_1 is the average proportion of class 1 and h is a parameter that defines traffic heterogeneity between the different origin/destination nodes (h is set to 20% in this work). In each case, the traffic demand of class 2 is the remaining from the original traffic ($d^2_{st} = d_{st} - d^1_{st}$). Using this method the matrices D_1 and D_2 were created for each problem instance. 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 matrices DR_c

(values for DR_p in the experiments were 3, 4 and 5[3]). This method was used to create the DR_2 matrices, since the class 1 was considered not to impose delay constraints[4].

In the experiments the following weights were set to each optimization aim: $\alpha_1 = 0.5$, $\beta_1 = 0$, $\alpha_2 = 0.25$ and $\beta_2 = 0.25$. In this way, both classes have a total weight of 50%, and in the case of class 2 both aims are taken to be of equal importance. Due to the fact that both class have a similar overall contribution for the optimization aim (50%), it is expected that the optimization process using the EA will give similar importance to the objectives of each class. Overall, a set of $12 \times 3 \times 3 = 108$ instances of the optimization problem were considered. 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 EA's population size was kept in 100 and the w_{max} was set to 20. The running times varied from a few minutes in the small networks, to a few hours in the larger ones. So, in order to perform all the tests, a computing cluster with 46 dual Xeon nodes was used. For all the optimization instances several results were collected, to allow to assess the effectiveness of the EA and of the heuristics used for comparison. Since the number of performed experiments is quite high, it was decided present aggregate results to draw conclusions. In this way, in the next sections the results obtained in all of the 108 optimization instances are averaged by D_p and DR_p (to understand the quality of the obtained solutions for distinct difficulty levels of the optimization instances) and by the number of edges considered in the experiments (to study the scalability issues of the solutions). In all figures presented in the following sections the data was plotted in a logarithmic scale, given the exponential nature of the penalty function adopted. In the figures the white area represents the acceptable working region whereas a gray area is used to identify regions with increasing levels of QoS degradation.

5.1 One Level of OSPF Weights

Figures 3 and 4 plot the QoS results obtained for class 1. As previously explained, this class is only constrained by the congestion performance (function Φ_1^*). The comparison between the methods in both figures 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 (both for results averaged by D_p and by DR_p), while the other heuristics manage very poorly. Regarding the results averaged by D_p, even $InvCap$, an heuristic quite used in practice, gets poor results when D_p is 0.2 or 0.3, 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 acceptable levels of congestion (i.e. results within the white area of the figures). For the results averaged by DR_p (Figure 4) the superiority of the EAs results is clearly visible for all the scenarios considered. Figure 5, on the other hand, represents

[3] Note that in this case lower values of DR_p represent harder optimization problems.

[4] In this specific experimental scenario, the class 1 was considered as only having bandwidth constraints while class 2 imposes both delay and bandwidth constraints. A practical example of this scenario might be obtained if one considers that class 1 is used to support elastic traffic (e.g. generated by TCP sources) while class 2 traffic is used to support delay sensitive traffic.

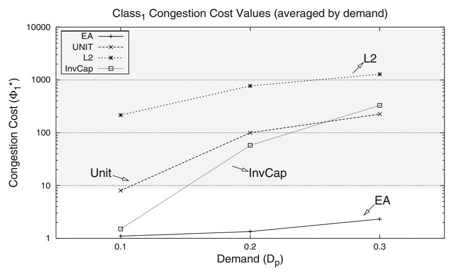

Fig. 3. Class 1 congestion results averaged by D_p (1 level of weights)

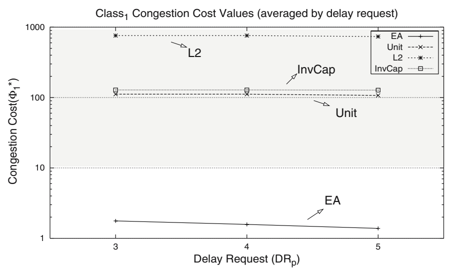

Fig. 4. Class 1 congestion results averaged by DR_p (1 level of weights)

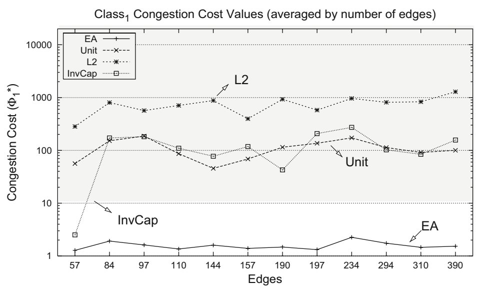

Fig. 5. Class 1 congestion results averaged by edges (1 level of weights)

the congestion values obtained by class 1, but aggregated by the number of arcs (links). It is clear 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.

To analyse the performance of class 2 a distinct graphical representation is used. As explained before, class 2 was considered as a multiconstrained QoS class, both in a congestion and in a delay perspective. In that way, the graphical representation of the results plotted by Figures 6 and 7 (D_p and DR_p averaged values, respectively) have the values of the two penalty measures in each axis (x axis for congestion and y axis for delay). In these graphs, the good overall network behavior of the solutions provided by the EA is clearly visible, both in absolute terms, regarding class 2 QoS 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 class 2 congestion results are acceptable only for $D_p = 0.1$ but fail completely in the delays. EAs, on the other hand, are capable of a good compromise between both optimization targets. As observed, EA solutions are well within the white area of the figures, which means that on average the congestion and delay demands of class 2 are satisfied by the networking domain using the solutions provided by the proposed framework using an unique level of OSPF weights.

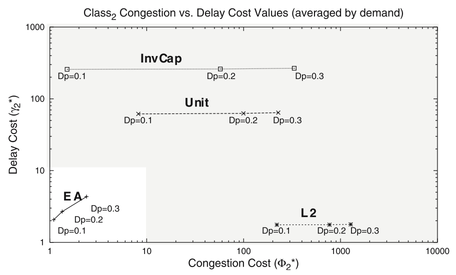

Fig. 6. Class 2 congestion vs delay results averaged by D_p (1 level of weights)

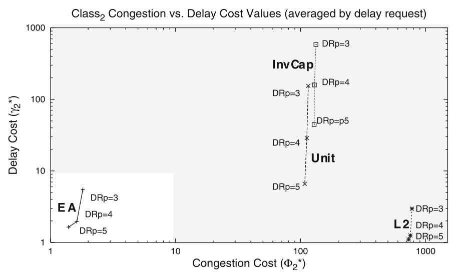

Fig. 7. Class 2 congestion vs delay results averaged by DR_p (1 level of weights)

5.2 Two Levels of OSPF Weights

The results discussed in the previous section showed that EAs are able to obtain near-optimal OSPF weight settings satisfying the QoS demands of the traffic classes. This section will now focus in the task of using several levels of OSPF weights to improve the network performance. Thus, the objective is to verify if the good results obtained by the EAs in scenarios with one level of weights can even be improved if two levels of weights are considered. For this purpose, in the optimization process of the overall cost function, two distinct levels of OSPF weights are now considered. One of the levels is used to compute the network paths for traffic belonging to class 1, while a distinct set of weights is used to route class 2 traffic. Based on this assumption, novel EA solutions were obtained for all instances of the optimization problem previously described (108 instances). The figures included in this section provide a comparison between the results obtained assuming one level of OSPF weights and the ones obtained using two levels of weights[5].

The results presented in Figures 8 and 9 show the congestion cost values associated with class 1 (cost function Φ_1^*). In this case, the congestion cost values obtained in all optimization instances are averaged by demand levels, D_p, and delay requirements, DR_p (see Figures 8 and 9, respectively). As observed, in both scenarios the improvements of using two levels of weights are visible for all values of D_p and DR_p. As expected, higher improvements are obtained in scenarios assuming harder QoS requirements, i.e. for $D_p = 0.3$ and $DR_p = 3$. In these scenarios, the cost function Φ_1^* achieves an improvement close to 21%. As expected, the results also show that as harder the QoS requirements get in a given network domain, the higher are the improvements expected to be obtained from the use of multiple levels of OSPF weights.

A different view of the congestion results of class 1 is presented in Figure 10. In this case, the values are averaged by the number of edges of the optimization instances. As before, the improvements obtained using two levels of weights are clear. In fact, for all values of number of edges, the Φ_1^* function cost values decrease in scenarios assuming two levels of weights (the higher improvement is obtained for the scenario with 234 edges where Φ_1^* is reduced in a value close to 36%). As regards to the

[5] In this case, due to the quality of the solutions obtained by the EAs, the figures only include the acceptable working region, i.e. the white area of the figures presented in the previous section.

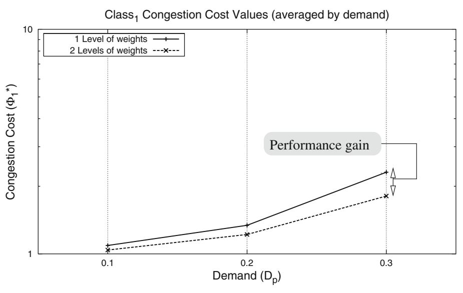

Fig. 8. Class 1 congestion results averaged by D_p (1 vs 2 levels of weights)

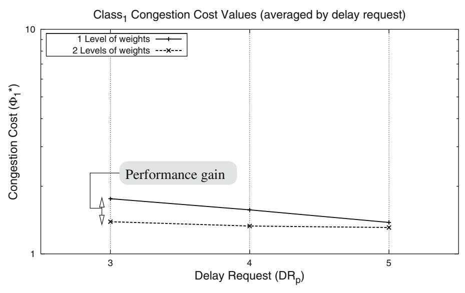

Fig. 9. Class 1 congestion results averaged by DR_p (1 vs 2 levels of weights)

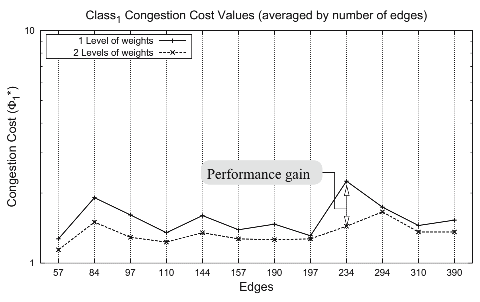

Fig. 10. Class 1 congestion results averaged by edges (1 vs 2 levels of weights)

performance of class 2, Figures 11 and 12 present the results of the improvements obtained in the congestion (Φ_2^*) and delay (γ_2^*) cost functions averaged by the number of edges used in the experimental scenarios. The improvements on each function are visible, since in all scenarios there is a congestion and delay performance gain. As example, there is a maximum congestion cost improvement close 30% (scenario with 84 edges) and a delay cost improvement with reaches a value of 60% (scenario with 234 edges).

As for the case of class 1 it is important to study the improvements observed in the class 2 performance when harder optimization problems are considered. In this context, the congestion and delay cost values of class 2 are also averaged according with the D_p and DR_p values. This information is provided by Figures 13 and 14 where the performance of class 2 is analysed for distinct values of the D_p and DR_p. As observed in Figure 13, for similar values of D_p there is a congestion and delay performance gain in class 2 when two levels of weights are used. The same reasoning is valid for the case of Figure 14 showing the values of the congestion and delay cost functions averaged by the DR_p parameters. In fact, it can be observed in both graphics that as the number of weight levels increases, the plots are shifted toward to the lower left corner of the graphs, which confirms the achievement of solutions having simultaneously a better delay and congestion performance.

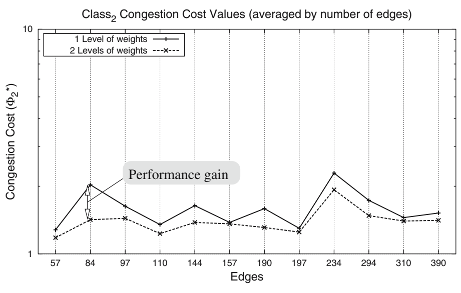

Fig. 11. Class 2 congestion results averaged by edges (1 vs 2 levels of weights)

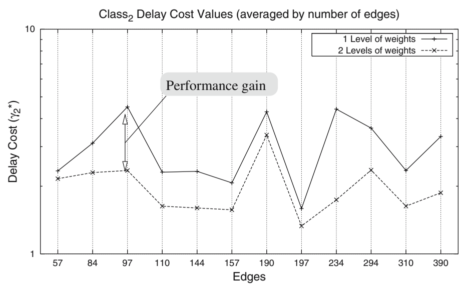

Fig. 12. Class 2 delay results averaged by edges (1 vs 2 levels of weights)

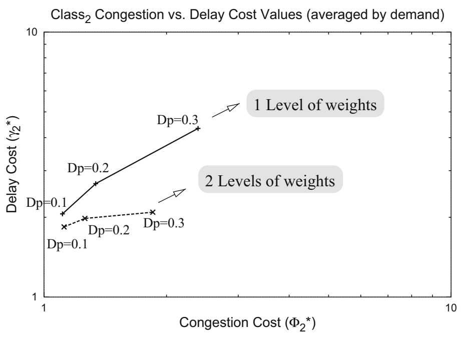

Fig. 13. Class 2 congestion vs delay results averaged by D_p (1 vs 2 levels of weights)

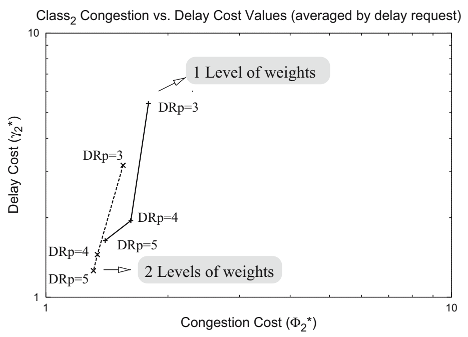

Fig. 14. Class 2 congestion vs delay results averaged by DR_p (1 vs 2 levels of weights)

6 Conclusions and Further Work

This work presented a novel traffic engineering optimization framework able to provide near-optimal OSPF weight configurations to network administrators. Resorting to a large number of QoS constrained scenarios, it was shown that high quality network configurations can be obtained using techniques inspired on the Evolutionary Computation field. The proposed framework is able to deal with single or multiple levels of OSPF weights. The results showed that, independently of other QoS aware mechanisms that might be in place, the proposed framework is able to improve the QoS level of a given class-based domain only taking into account the direct influence of the routing component of the network. In the future, the consideration of more specific EAs to handle this class of multi-objective problems [15][16] will also be taken into account. In a similar way, it is also possible to study the impact of link failures in the solutions devised by the proposed framework. Another important future research topic is the study of the sensibility of the achieved EA solutions to changes in the demand matrices. This specific research topic will allow for the use of this type of management tools in scenarios where the demands matrices are obtained in a more finer-grain temporal perspective.

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. J. Moy. RFC 2328: OSPF version 2, April 1998.
4. T.M. ThomasII. *OSPF Network Design Solutions*. Cisco Press, 1998.
5. B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proceedings of IEEE INFOCOM*, pages 519–528, 2000.
6. A. Medina et al. Traffic matriz estimation: Existing techniques and new directions. *Computer Communication Review*, 32(4):161–176, 2002.
7. Alan Davy, Dmitri Botvich, and Brendan Jennings. An efficient process for estimation of network demand for qos-aware ip networking planning. In Gerard Parr, David Malone, and Mícheál Ó Foghlú, editors, 6^{th} *IEEE International Workshop on IP OPerations and Management, IPOM 2006, LNCS 4268*, pages 120–131. Springer-Verlag, 2006.
8. P. Psenak et al. Multi-topology (mt) routing in ospf (internet draft), November 2006.
9. Ravindra et al. *Network Flows*. Prentice Hall, 1993.
10. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(269-271), 1959.
11. J. Moy. *OSPF, Anatomy of an Internet Routing Protocol*. Addison Wesley, 1998.
12. G. Bolch et al. *Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications*. Jhon Wiley and Sons INC., 1998.
13. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, USA, third edition, 1996.
14. A. Medina et al. BRITE: Universal Topology Generation from a User's Perspective. Technical Report 2001-003, January 2001.
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.