Evolutionary Computation for Quality of Service Internet Routing Optimization

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

[1] Dep. Informatics / CCTC - Univ. Minho - Braga - Portugal
mrocha@di.uminho.pt, pns@di.uminho.pt
[2] Dep. Information Systems/ Algoritmi - Univ. Minho - Guimarães - Portugal
pcortez@dsi.uminho.pt
[3] University College London - London - UK
m.rio@ee.ucl.ac.uk

Abstract. In this work, the main goal is to develop and evaluate a number of optimization algorithms in the task of improving *Quality of Service* levels in TCP/IP based networks, by configuring the routing weights of link-state protocols such as OSPF. Since this is a complex problem, some meta-heuristics from the *Evolutionary Computation* arena were considered, working over a mathematical model that allows for flexible cost functions, taking into account several measures of the network behavior such as network congestion and end-to-end delays. A number of experiments were performed, resorting to a large set of network topologies, where *Evolutionary Algorithms (EAs)*, *Differential Evolution* and some common heuristic methods including *local search* were compared. *EAs* make the most promising alternative leading to solutions with an effective network performance even under unfavorable scenarios.

Keywords: Traffic Engineering, Quality of Service Routing, Evolutionary Algorithms, Differential Evolution, OSPF.

1 Introduction

The relevance of implementing *Quality of Service (QoS)* support mechanisms in IP-based networks has been fostered in the last few years by the integration of a number of new applications. Several distinct *QoS* aware architectures and traffic control mechanisms have been proposed in order to provide distinct service levels to networked applications [13]. In this context, *Internet Service Providers (ISPs)* have agreements with their clients and with other *ISPs* that have to be obeyed. To face such requirements, there is an important set of configuration tasks that have to be performed by network administrators in order to assure that correct resource provisioning is achieved in the domain.

There is not an unique solution to create a *QoS* aware infrastructure and any solution requires a number of components working together. However, independently of the particular solutions adopted that might be in place, there are components which have a crucial importance. One of such components has the ability to control the data path followed by packets traversing a given *Wide*

M. Giacobini et al. (Eds.): EvoWorkshops 2007, LNCS 4448, pp. 71–80, 2007.

Area Network (WAN). In a *WAN*, consisting of a single administrative domain, there are alternative strategies for this purpose: Intra-domain routing protocols or *Multi-Protocol Label Switching (MPLS)* [2].

This work will focus on intra-domain routing protocols, and more specifically on the most commonly used today: *Open Shortest Path First (OSPF)* [12]. Here, the administrator assigns weights to each link in the network, which are used to compute the best path from each source to each destination node using the *Dijkstra algorithm* [5]. The results are then used to compute the routing tables at each node. Since in *OSPF* the weight setting process is the only way administrators can affect the network behavior, this choice is of crucial importance, having a major impact on network performance. Nevertheless, in practice, simple rules are typically used, like setting the weights inversely proportional to the link capacity, often leading to sub-optimal resource utilization.

An ideal way to improve the process of *OSPF* weight setting is to implement traffic engineering, assuming that the administrator has access to the traffic demands between each pair of nodes in the network. This was the approach taken by Fortz et al [7] where this task was viewed as an NP-hard optimization problem by defining a cost function that measures network congestion. Some local search heuristics have been proposed, as well as the use of meta-heuristics [6]. However, such approaches did not accommodate delay based constraints that are crucial to implement *QoS* aware networking services.

In this paper, a number of optimization algorithms (*Evolutionary Algorithms*, *Differential Evolution*, local search) are employed to calculate link-state routing weights, that optimize traffic congestion while simultaneously complying to specific delay requirements. A mathematical model of the problem that accommodates both congestion and delay constraints is used to define a bi-objective cost function and therefore to develop fitness functions for the algorithms, which are then used to calculate the optimal OSPF weights for each network link.

An important and direct outcome of the research work presented in this paper is the ability of developing network management tools which automatically provide network administrators with near-optimal routing configurations for *QoS* constrained networking scenarios. In this context, devising efficient and accurate routing optimization methods will be a major contribution for pursuing optimal routing configurations in the Internet.

2 Problem Description

The general routing problem [1] represents routers and links by a set of nodes (N) and arcs (A) in a directed graph $G = (N, A)$. In this model, c_a represents the capacity of each link $a \in A$. A demand matrix D is available, where each element d_{st} represents the traffic demand between nodes s and t. 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 as:

$$l_a = \sum_{(s,t) \in N \times N} f_a^{st} \tag{1}$$

while 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 ($\Phi_a = p(u_a)$) [7], 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 (Figure 1).

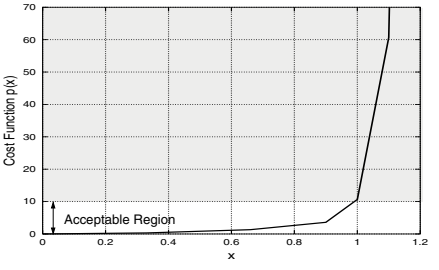

Fig. 1. Graphical representation of the penalty function p

In *OSPF*, all arcs have an integer weight. Every node uses these weights in the Dijkstra algorithm [5] 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) [10].

Let us assume a given solution, a weight assignment (w), and the corresponding utilization rates on each arc (u_a). In this case, the total routing cost is expressed by $\Phi(w) = \sum_{a \in A} \Phi_a(w)$ for the loads and corresponding penalties ($\Phi_a(w)$) calculated based on the given OSPF weights w. In this way, the *OSPF weight setting problem* 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 ($\Phi^*(w)$) over distinct topology scenarios and its value is in the range [1,5000]. It is important to note that when Φ^* 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^* is $10\frac{2}{3}$. This value will be considered as a threshold that bounds the acceptable working region of the network.

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 $(s,t) \in N \times N$ gives the delay target for traffic between s and t (denoted by DR_{st}). 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 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 compliance ratio* for a given pair $(s,t) \in N \times N$ is, therefore, defined as $dc_{st} = \frac{Del_{st}}{DR_{st}}$. A penalty for delay compliance can be calculated using

function p. The γ_{st} function is defined according to $\gamma_{st} = p(dc_{st})$. This allows the definition of a delay cost function, given a set of *OSPF* weights (w):

$$\gamma(w) = \sum_{(s,t)\in N\times N} \gamma_{st}(w) \tag{2}$$

where 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 to reach the value of $\gamma^*(w)$ (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, a demand matrix D and a delay requirements matrix DR, the aim is to find the set of *OSPF* weights w that simultaneously minimizes the functions $\Phi^*(w)$ and $\gamma^*(w)$. When a single objective is considered the cost of a solution w is calculated using functions $\Phi^*(w)$ for congestion and $\gamma^*(w)$ for delays. For multi-objective optimization a quite simple scheme was devised, where the cost of the solution is given by: $f(w) = \alpha\Phi^*(w) + (1-\alpha)\gamma^*(w)$. This scheme, although simple, can be effective since both cost functions are normalized in the same range.

3 Algorithms for OSPF Weight Setting

3.1 Evolutionary Algorithms

In this work, *Evolutionary Algorithms (EAs)* [9] are proposed to address the problems defined in the previous section, both by considering the single or the multi-objective formulation. 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 one crossover operator:

- *Random Mutation*, replaces a given gene by a randomly generated value, within the allowed range;
- *Incremental/decremental Mutation*, replaces a given gene by the next or by the previous value (with equal probabilities) within the allowed range;
- *Uniform crossover*, a standard crossover operator [9].

In each generation, every operator is used to create new solutions with equal probabilities. 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 the experiments a population size of 100 individuals was considered.

3.2 Differential Evolution

The *DE* method differs from the previous *EA* essentially in the reproduction operators. *DE* generates trial individuals by calculating vector differences between other randomly selected members of the population. In this work, a variant of the *DE* algorithm called *DE/rand/1* was considered that uses a binomial crossover [11]. In this case, the following scheme is followed for each individual i:

1. Randomly select 3 individuals r_1, r_2, r_3 distinct from i;
2. Generate a trial vector based on: $\boldsymbol{t} = \boldsymbol{r}_1 + F \cdot (\boldsymbol{r}_2 - \boldsymbol{r}_3)$
3. Incorporate coordinates of this vector with probability CR;
4. Evaluate the candidate and use it in the new generation if it is at least as good as the current individual.

Since *OSPF* weights are integer, it is necessary to round the values used in the *DE* before the evaluation. It is important to notice that in the *DE* all individuals in the population go throught the previous reproduction step. In the experiments, the population size was 20, F was set to 0.5 and CR to 0.6.

3.3 Local Search

A *local search (LS)* scheme was devised to improve the quality of a solution and works as follows: taking a set of weights w_i, a link is randomly selected to start the process. Firstly, it tries to increase the value of this weight by 1, if this implies that the solution is better. This process is repeated while the solution improves. If the first increase operation did not lead to a better solution, a decrease is tried and repeated while the solution improves.

The process is repeated for the next position, until all positions have been tested. The overall process is then repeated while the solution improves. Based on this *LS* operator, a *multi-start LS (MS-LS)* algorithm was devised: it starts with a random solution and applies the *LS* operator; this process is repeated and the best solution found is kept. The process is terminated when a maximum number of solutions has been evaluated.

3.4 Heuristic Methods

A number of heuristic methods were implemented [7] in order to assess the order of magnitude of the improvements obtained by the proposed methods when compared with the traditional weight setting heuristics, namely:

- **InvCap** - sets each link weight to a value inversely proportional to its capacity;
- **L2** - set each link weight to a value proportional to the its Euclidean distance;
- **Random** - a number of randomly generated solutions are analyzed and the best is selected.

4 Experiments and Results

In order to evaluate the proposed algorithms, a number of experiments was conducted. The experimental platform used in this work is presented in Figure 2. All the algorithms and the OSPF routing simulator were implemented using the *Java* language. The first step was the generation of a set of 12 networks by using the Brite topology generator [8], 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 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 all experiments only propagation delays were considered.

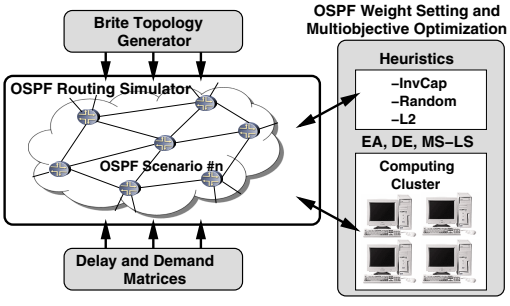

Fig. 2. Experimental platform for OSPF performance evaluation

Next, the demand and delay constraints matrices (D and DR) were generated. For each of the networks a set of three distinct D and DR matrices were created. A parameter (D_p) was considered, giving the expected mean of congestion in each link (u_a) (values for D_p in the experiments were 0.1, 0.2 and 0.3). For 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 (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.

The termination criteria of the optimization algorithms (*EAs*, *DE* and *LS*) was the maximum number of solutions evaluated that ranged from 50000 to 300000, increasing linearly with the number of links in the problem. The running times varied from a few minutes to a few hours, in the larger instances. In all cases, w_{max} was set to 20. For all the stochastic algorithms, 10 runs were executed in each case.

The results are grouped into two sets according to the cost function used. The first considers a single objective cost function, for the optimization of network congestion. The latter considers the case of a multi-objective cost function,

dealing with both congestion and delay optimization. In all figures presented in this section, the data was plotted in a logarithmic scale, given the exponential nature of the penalty function adopted.

4.1 Congestion

Since the number of performed experiments is quite high, it was decided to show only some aggregate results that can be used to draw conclusions. Table 1 shows the results for all the available networks, averaged by the demand levels (D_p), including in the last line the overall mean value for all problem instances. It is clear that the results get worse with the increase of D_p, as would be expected. Figure 3 plots the same results in a graphical way, showing in the the white area the acceptable working region, whereas an increasing level of gray is used to identify working regions with increasing levels of service degradation.

The comparison between the methods shows a superiority of the *EA*. In fact, the *EA* achieves solutions which manage a very reasonable behavior in all scenarios (worse case is 1.49). The heuristics manage very poorly, and 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 levels of congestion. The results of *DE* and *MS-LS* are

Table 1. Results for the optimization of congestion (Φ^*) - averaged by demand levels

D_p	Random	EA	DE	MS-LS	L2	InvCap
0.1	75.75	1.02	1.02	1.12	215.94	1.50
0.2	498.74	1.18	1.41	1.50	771.87	57.70
0.3	892.87	1.73	3.64	6.08	1288.56	326.33
Overall	489.12	1.31	2.02	2.90	758.79	128.51

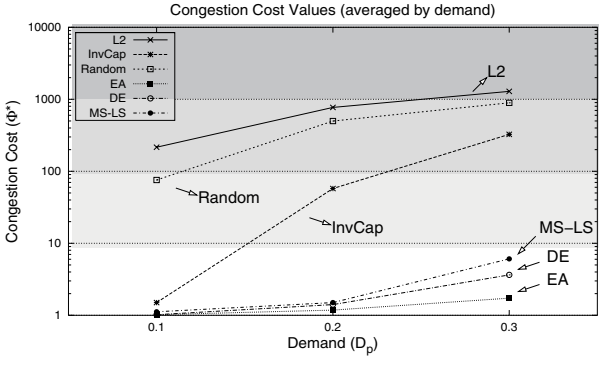

Fig. 3. Graphical representation of the results obtained by the different methods in congestion optimization (averaged by D_p)

acceptable, but nevertheless significantly worse than the ones obtained by the *EA*, and the gap increases with larger values of D_p.

4.2 Multi-objective Optimization

In this section, the results for the multi-objective optimization are discussed. The results 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 are not relevant to the analysis. Given the space constraints only the value of 0.5 will be considered for parameter α, thus considering each aim to be of equal importance. Table 2 shows the results averaged by the demand level (D_p). From the table it is clear that the *EA* outperforms all other algorithms, followed by the *DE* and *MS-LS*. The heuristics behave quite badly, when both aims are taken into account. A similar picture is found looking at Table 3, where the results are averaged by the delay requirement parameter DR_p.

Table 2. Results for the multi-objective optimization - averaged by D_p

D	Random		EA		DE		MS-LS		L2		InvCap	
	Φ^*	γ^*	Φ^*	γ^*	Φ^*	γ^*	Φ^*	γ^*	Φ^*	γ^*	Φ^*	γ^*
0.1	88.00	106.79	1.17	1.92	1.18	2.04	1.73	4.07	215.94	1.76	1.50	260.30
0.2	481.50	136.68	1.47	2.32	1.65	2.92	3.38	8.30	771.87	1.76	57.70	260.30
0.3	949.85	148.96	2.41	3.23	4.58	5.64	15.31	15.95	1288.56	1.76	326.33	260.30
Overall	506.45	130.81	1.68	2.49	2.47	3.53	6.81	9.44	758.79	1.76	126.51	260.30

Table 3. Results for the multi-objective optimization - averaged by DR_p

DR	Random		EA		DE		MS-LS		L2		InvCap	
	Φ^*	γ^*	Φ^*	γ^*	Φ^*	γ^*	Φ^*	γ^*	Φ^*	γ^*	Φ^*	γ^*
3	535.28	283.16	1.95	4.22	2.78	6.42	9.65	21.29	758.79	2.94	128.51	577.94
4	505.69	82.04	1.59	1.78	2.44	2.36	6.12	4.65	758.79	1.25	128.51	158.85
5	478.37	27.23	1.51	1.48	2.38	1.82	4.65	2.39	758.79	1.10	128.51	44.13

A different view is offered by Figures 4 and 5 where the results are plotted with the two objectives in each axis. The former shows the results averaged by the demand levels and the latter by the delay requirements parameter. 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. *DE* gets results that are in an acceptable range, but are always significantly worse than those of the *EAs*, and *MS-LS* does not manage good results when the problem instances get harder.

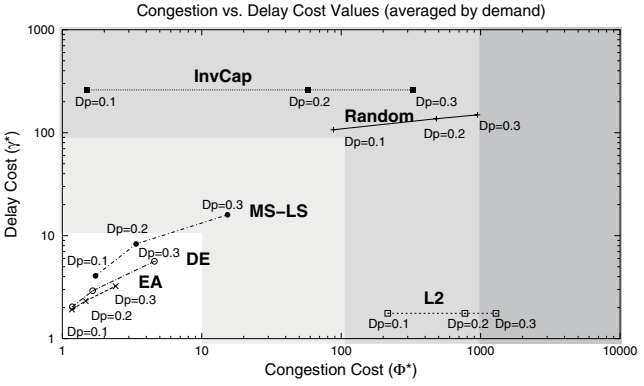

Fig. 4. Graphical representation of the results obtained by the different methods in the multi-objective optimization (averaged by D_p)

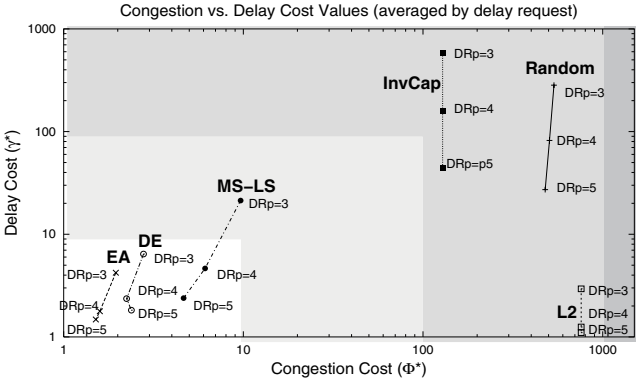

Fig. 5. Graphical representation of the results obtained by the different methods in the multi-objective optimization (averaged by DR_p)

5 Conclusions and Further Work

The optimization of OSPF weights brings important tools for traffic engineering, without demanding modifications on the basic network management model. This work presented *Evolutionary Computation* approaches for multi-objective routing optimization in the Internet. Resorting to a set of network configurations, each constrained by bandwidth and delay requirements, it was shown that the proposed *EAs* were able to provide OSPF weights that can lead to good network behavior. The performance of *EAs* was compared with other algorithms (*DE*, local search, heuristics) clearly showing its superiority. The proposed optimization framework, although requiring some computational effort, can be achieved in useful time and implemented in a real-world scenario.

Although a simple weighting method was used to face the multi-objective nature of the problem, the results were of high quality. This is probably due to the effort of normalizing both cost functions. Nevertheless, the consideration of specific *EAs* to handle this class of problems [4] will be taken into account in future work. *Memetic Algorithms*, that consider local optimization procedures embedded in the *EA*, have also been attempted in the congestion optimization problem [3]. Their application in this bi-objective scenario is also a research direction that has a strong potential.

References

1. R. K. Ahuja, T. L. Magnati, and J. B. Orlin. *Network Flows.* Prentice Hall, 1993.
2. D. Awduche and B. Jabbari. Internet traffic engineering using multi-protocol label switching (MPLS). *Computer Networks*, 40:111–129, 2002.
3. L. Buriol, M. Resende, C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks*, 2003.
4. C.A. Coello Coello. *Recent Trends in Evolutionary Multiobjective Optimization*, pages 7–32. Springer-Verlag, London, 2005.
5. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(269-271), 1959.
6. M. Ericsson, M. Resende, and P. Pardalos. A Genetic Algorithm for the Weight Setting Problem in OSPF Routing. *J. Combinatorial Optimiz.*, 6:299–333, 2002.
7. B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proceedings of IEEE INFOCOM*, pages 519–528, 2000.
8. A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: Universal Topology Generation from a User's Perspective. Technical Report 2001-003, 2001.
9. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, USA, third edition, 1996.
10. J. Moy. *OSPF, Anatomy of an Internet Routing Protocol.* Addison Wesley, 1998.
11. R. Storn and K. Price. Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997.
12. T.M. ThomasII. *OSPF Network Design Solutions.* Cisco Press, 1998.
13. Zheng Wang. *Internet QoS: Architectures and Mechanisms for Quality of Service.* Morgan Kaufmann Publishers, 2001.