Multiobjective Evolutionary Algorithms for Intradomain Routing Optimization

Miguel Rocha, Tiago Sa and Pedro Sousa
CCTC/ Dep. Informatics
Universidade do Minho, Portugal
Email: {mrocha,tiagosa,pns}@di.uminho.pt

Paulo Cortez
Algoritmi Center
Dep. Information Systems
Universidade do Minho, Portugal
Email: pcortez@dsi.uminho.pt

Miguel Rio
Dep. Electronic and Electrical Engineering
University College London, UK
Email: m.rio@ee.ucl.ac.uk

***Abstract*—Evolutionary Algorithms (EAs) have been used to develop methods for Traffic Engineering (TE) over IP-based networks in the last few years, being used to reach the best set of link weights in the configuration of intra-domain routing protocols, such as OSPF. In this work, the multiobjective nature of a class of optimization problems provided by TE with Quality of Service constraints is identified. Multiobjective EAs (MOEAs) are developed to tackle these tasks and their results are compared to previous approaches using single objective EAs. The effect of distinct genetic representations within the MOEAs is also explored. The results show that the MOEAs provide more flexible solutions for network management, but are in some cases unable to reach the level of quality obtained by single objective EAs. Furthermore, a freely available software application is described that allows the use of the mentioned optimization algorithms by network administrators, in an user-friendly way by providing adequate user interfaces for the main TE tasks.**

Keywords: Multiobjective optimization, traffic engineering, Quality of Service, intra-domain routing, OSPF

I. Introduction

Internet Service Providers (ISPs) are increasingly facing requests from their clients to support diverse types of applications over IP based networks and to provide increasingly demanding Quality of Service (QoS) levels. In this context, and among numerous other mechanisms to build a QoS capable infrastructure, a number of proposals have recently been put forward to achieve Traffic Engineering (TE) using intra-domain routing protocols [1] [2].

This is the case of the well known Open Shortest Path First (OSPF) protocol [3] [4], where the administrator assigns weights to each link in the network, which are then used to compute the best path from each source to each destination, being the results used to compute the routing tables for each router. Since, in this case, the weight setting process is the only way administrators can affect the network behaviour, this choice is of crucial importance and may have a major impact in the network performance. Nevertheless, in practice, simple 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.

The process of OSPF weight setting can be improved assuming that the administrator has access to a matrix representing traffic demands between each pair of nodes in the network. This was the approach followed by Fortz et al [1] where this task is viewed as an optimization problem, by defining a penalty based cost function that measures the network congestion.

Following this trend several authors have proposed single and multi-objective optimization approaches for this task: Sqalli et al [5] proposed the use of the number of congested links as a further objective and useed Simulated Annealing as the optimization method; a similar approach is followed by Sait el tal [6] that use a cost function based on the utilization and the extra load caused by condested links; finally, Brostrom et al [7] also considered a multi-objective approach, where the second optimization objective was related to the capability of the network to endure link failures;

The previous approaches did not, however, accommodate delay based constraints that are crucial to implement QoS aware networking services. Thus, in previous work, the authors proposed an extended multiconstrained QoS aware optimization framework, where Evolutionary Algorithms (EAs) are used to calculate link-state routing weights that optimize traffic congestion, while simultaneously complying with specific delay requirements [2]. Within this framework, a mathematical model of the problem, that accommodates both congestion and delay constraints was proposed and a bi-objective cost function is defined. Since each of the objectives within this function is normalized in the same range, a linear weighting scheme was used with good results. In this approach, a parameter was selected that can be used to tune the trade-off between both components of the cost function.

The previous approach has one important limitation, since it assumes that there is a single trade-off that is optimum. Also, the user needs to guess which value of the weighting parameter better fits the needs of a particular network. An alternative is to have algorithms that can calculate a set of solutions with distinct trade-offs between the two objectives, and let the network administrator decide which solution to implement. A number of algorithms have been proposed in the last few years to address this task, in the arena of Multiobjective Optimization (MOO). Multiobjective EAs (MOEAs) are among the most well succeeded ones [8].

In this work, the main aim is to evaluate the performance of MOEAs in the above defined TE task and to compare its

results with the previously proposed single objective EAs. Also, a complementary goal is to allow the use of these methods by network administrators resorting to simple and intuitive network management computational tools able to put such knowledge in practice.

In the following sections the optimization problem and the proposed algorithms are described. Then, the experiments conducted in this work are described and the results are put forward. The next section describes the application that is made available and the paper closes with some conclusions.

II. Problem description

The proposed optimization framework deals with the provision of network administrators with efficient OSPF link configurations, taking into account the users demands, the topology and other features of a given network domain. This work assumes that client demands are mapped into a matrix, summarizing, for each source/destination router pair, a given amount of bandwidth and end-to-end delay required to be supported by the network domain.

The general routing problem [9], 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 in the following way:

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

while the link utilization rate u_a is given by:

$$u_a = \frac{l_a}{c_a} \tag{2}$$

It is then possible to define a congestion measure for each link ($\Phi_a = p(u_a)$), where p is a penalty 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 (see [1] for details).

In OSPF, all arcs are associated with an integer weight. Every node uses these weights in the Dijkstra algorithm [10] to calculate the shortest paths to all other nodes in the network, where 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) [11].

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:

$$\Phi(w) = \sum_{a\in A} \Phi_a(w) \tag{3}$$

for the loads and corresponding penalties $(\Phi_a(w))$ calculated based on the given OSPF weights. 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 over distinct topology scenarios, dividing by a scaling factor defined as:

$$\Phi_{UNCAP} = \sum_{(s,t)\in N\times N} d_{st}h_{st} \tag{4}$$

where h_{st} is the minimum hop count between nodes s and t.

Finally, the scaled congestion measure cost is defined as:

$$\Phi^*(w) = \frac{\Phi(w)}{\Phi_{UNCAP}} \tag{5}$$

and the following relationships hold [1]:

$$1 \leq \Phi^*_(w_{opt}) \leq 5000 \tag{6}$$

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 2/3. This value will be considered as a threshold that bounds the acceptable working region of the network.

To enable an enlarged set of QoS constraints, an extension to this model is proposed in this work. This enrichment allows the inclusion of delay requirements for each pair of routers in the network. These are modelled 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 origin s and destination t (denoted by DR_{st}). A cost function was developed to evaluate the delay compliance for each scenario (a set of OSPF weights). 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.

It was considered that, in the scenarios where this work would be applicable, the delay in each path is dominated by the component given by propagation delays in its arcs and that queuing delays can be neglected. However, if required, queuing delays can be introduced in the model by approximating its values resorting to queuing theory [12], taking into account the following parameters at each node: the capacity of the corresponding output links, their utilization rates and more specific technical parameters such as the mean packet size and the overall queue size associated with each link.

The *delay compliance ratio* for a given pair $(s,t) \in N \times N$ is, therefore, defined as:

$$dc_{st} = \frac{Del_{st}}{DR_{st}} \tag{7}$$

A penalty for delay compliance can be calculated using function p. So, the γ_{st} function is defined according to the following equation:

$$\gamma_{st} = p(dc_{st}) \tag{8}$$

This, in turn, allows the definition of a delay minimization cost function, given a set of OSPF weights (w):

$$\gamma(w) = \sum_{(s,t)\in N\times N} \gamma_{st}(w) \quad (9)$$

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 (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):

$$\gamma^*(w) = \frac{\gamma(w)}{\sum_{(s,t)\in N\times N} minDel_{st}} \quad (10)$$

It is now possible to define the optimization problem addressed in this work that is clearly multi-objective. Indeed, given a network represented by a graph $G = (N, A)$, 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 single objective optimization, the algorithms use a linear weighting scheme where the cost of the solution is given by:

$$f(w) = \alpha\Phi^*(w) + (1-\alpha)\gamma^*(w), \alpha \in [0,1] \quad (11)$$

III. Algorithms for OSPF weight setting

A. Single objective Evolutionary Algorithms

In the proposed single objective EA [2], each individual encodes a solution as a vector of integer values, where each value (gene) corresponds to the weight of a link (arc) in the network (the values range from 1 to w_{max}). Therefore, the size of the individual equals the number of links in the network. The individuals in the initial population are randomly generated, with the arc weights taken from a uniform distribution.

In order to create new solutions, several reproduction operators were used, more specifically two mutation operators and one crossover operator:

- *Random Mutation*, replaces a given gene by a random 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 [13], suitable when the order of the variables in the individual (solution) is not important.

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. This EA is implemented using JECoLi, a general-purpose Evolutionary Computation library recently developed by the authors [14].

B. Multiobjective algorithms

One alternative to the use of EAs are Multi-objective EAs (MOEAs) that are able to give as output not only one optimal solution, but rather a set of solutions that are non-dominated. A solution is dominated by another solution, if the first is worse than the second in at least one of the objectives and it is not better in none. More precisely, the aim of these methods is to return a Pareto front (PF), i.e. a set of non-dominated solutions, for a given problem. This PF should be as near as possible to the optimal set of non-dominated solutions and also as distributed as possible, i.e. it should cover the whole set of possible trade-offs between the optimization aims.

In order to evaluate the performance of MOEAs, in the OSPF weight setting task previously defined, two different implementations were addressed. As a first attempt, a number of state-of-the-art alternatives was implemented based on the use of the *jMetal* software [15], that served as a basis to implement the following algorithms: SPEA2, NSGA-II, PESA-II, PEAS, PSO, AbYSS and MOCell. These encompass several types of MOEAs based on distinct optimization approaches: Particle Swarm Optimization, Evolution Strategies and Scatter Search, thus covering the most popular multiobjective optimization algorithms available today. In this case, the solutions were represented using real-valued representations and the default configurations.

To test if the representation used could have in important influence over the results, it was decided to implement the two MOEAs with the best results (NSGA-II and SPEA2) using JECoLi, which allowed the use of the same integer based representation and reproduction operators that were used by the EAs described above.

IV. Experiments and results

A. Experimental setup

To evaluate the proposed algorithms, a number of experiments were conducted. All the algorithms and the OSPF routing simulator were implemented using the Java language. A set of 3 networks was created using the Brite topology generator [16], varying the number of nodes ($N = 30, 50, 80$), while the average degree of each node is 4. This resulted in networks ranging up to 390 links. The link bandwidth was generated by a 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).

Next, the demand and delay constraints matrices (D and DR) were generated. For each network, a set of three distinct D and DR matrices were created. A parameter (D_p) was considered, representing the expected mean of congestion in each link (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 were 3, 4 and 5). Overall, a set of 27 instances of the optimization problem were considered.

The termination criteria for all algorithms consisted in the maximum number of solutions evaluated. This value ranged from 50000 to 300000, increasing linearly with the number of links in the instance. In all cases, w_{max} was set to 20. For all the stochastic algorithms, 10 runs were executed in each case. For the single objective EA, three distinct values of α were tested: 0.25, 0.5 and 0.75.

B. Metrics

Evaluating the performance of MOO algorithms is a complex task and to compare the results of MOO approaches with traditional single objective methods is still trickier. This study does not intend to be exhaustive in this comparison and two simple performance metrics were used to evaluate the approaches:

- *C-measure* [17]: It is based on the concept of solution dominance. Given two PFs ($PF1$,$PF2$), the measure $C(PF1, PF2)$ returns the fraction of solutions in $PF2$ that are dominated by at least one solution in $PF1$. A value of 1 indicates that all points in $PF2$ are dominated by points in $PF1$, so values near 1 clearly favour the method that generated $PF1$; values near 0 show that few solutions in $PF2$ are dominated by solutions in $PF1$. This concept can be extended to traditional single-solution methods by calculating $C(S1, PF2)$ where $S1$ is a single solution. Therefore, it simply indicates the proportion of solutions in $PF2$ that are dominated by $S1$.
- *Trade-off analysis (TOA)*: For a pareto front $PF1$, and given a value of β, the solution that maximizes $\beta\Phi^* + (1-\beta)\gamma^*$ is selected. Parameter β can take distinct values in the range $[0, 1]$, thus defining different trade-offs between the objectives (working in a way similar to the parameter α in previous sections, but applied only after the optimization process). The values with the same β can be compared among the several MOO algorithms and also with those from traditional algorithms. In this last case, only one solution is available, so the process is simplified.

C. Results

In Table I, the results for the C-measure are shown. The overall mean value for all the distinct instances and runs was computed. For each instance, $C(M1, M2)$ is computed for all pairs of distinct runs of $M1$ and $M2$. The first set of algorithms (rows a to g) provides results for the MOEAs implemented using jMetal, while rows h and i refer to JECoLi based MOEAs. In the final 3 rows are listed the single objective EAs, being given the value of α for each case.

TABLE I
RESULTS FOR THE C-MEASURE IN MOO (MEAN OF $C(M1, M2)$ IN PERCENTAGE COMPUTED OVER ALL INSTANCES AND RUNS).

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
a) AbYSS	0	35	3	11	16	10	2	0	0
b) MOCell	49	0	3	12	21	12	1	1	0
c) NSGAII	92	91	0	46	71	65	35	15	8
d) PAES	65	63	22	0	45	33	17	8	3
e) PESAII	68	60	17	23	0	34	12	4	2
f) PSO	77	73	12	26	42	0	7	4	1
g) SPEA2	89	87	42	42	74	63	0	15	8
h) NSGAj	99	97	70	77	90	80	57	0	25
i) SPEAj	99	95	81	78	93	79	78	52	0
EA(0.25)	82	80	51	39	61	56	44	30	18
EA(0.5)	78	79	46	36	56	57	42	26	15
EA(0.75)	67	69	37	30	46	55	33	21	12

In Table II, the TOA results are displayed for the same set of algorithms. As before, means were calculated over all instances and runs, for each value of β.

TABLE II
RESULTS FOR THE TOA IN MOO (MEAN OVER ALL INSTANCES AND RUNS GIVEN THE VALUE OF β).

Algorithm	$\beta = 0$	$\beta = 0.25$	$\beta = 0.5$	$\beta = 0.75$	$\beta = 1$
AbYSS	8.9	31.1	53.3	75.4	97.6
MOCell	8.1	33.9	59.7	85.5	111
NSGAII	4.2	12.8	21.3	29.9	38.4
PAES	5.8	96.2	187	277	367
PESAII	5.5	30.3	55.1	80.0	105
PSO	6.1	26.4	46.7	67.0	87.3
SPEA2	3.4	11.8	20.3	28.7	37.1
NSGAj	2.4	4.0	4.0	3.9	3.4
SPEAj	2.6	4.1	5.0	5.9	6.5
EA(0.25)	3.2	2.9	2.6	2.3	2.0
EA(0.5)	3.8	3.3	2.7	2.2	1.6
EA(0.75)	4.2	3.5	2.8	2.2	1.5

An analysis of the results of both tables shows that, when comparing the MOO approaches, the NSGA-II and SPEA2 outperform all other alternatives. In fact, in Table I they show the highest values in the rows and the lowest in the columns, thus having PFs with few dominated and many dominating solutions. In Table II, they also present the lowest results for the distinct values of β.

When the *jMetal* and the *JECoLi* versions of those two algorithms are compared, the advantage of the latter is quite noticeable. Therefore, it seems that the use of an integer representation closer to the problem domain brings significant advantages in the final results.

When comparing the MOO performance with the one obtained by the single objective EA, it is also clear that the solutions obtained by the jMetal based MOEAs are, in general, not good alternatives for network management (from Table II). In fact, they are quite far from the results obtained by the EA that, regardless of the values of α (used in the optimization) and β, always show quite low values. Therefore, the proposed EA with linear weighting shows a better trade-off between both objectives, while MOEAs show a bias, behaving better in delay optimization and failing in congestion. From Table I

it is also possible to conclude that a large number of MOO solutions are dominated by the single objective EA's solutions, while the reverse is not true. In fact, the columns for the EAs are not shown in the table because its values were always zero. This means that the final solution for the EA is never dominated by any solution obtained by a MOO algorithm.

The previous trends are partially also true, when JECoLi based MOEAs are considered, but the differences are substantially attenuated. In fact, these MOEAs have a much smaller number of solutions dominated by the EAs, which means that they are able to better explore the universe of possible trade-offs between both objectives. Also, from TOA analysis it can be concluded that all results from these MOEAs are within the working region of the network and, therefore, in every case are reasonable solutions. Still, it must also be noted that these MOEAs do behave better in delay optimization where they obtain even better solutions than EAs but do not reach the same level of quality in the congestion component. Overall, the NSGAII from JECoLi seems to provide the best MOEA for the job.

V. A User-Friendly Traffic Engineering Application

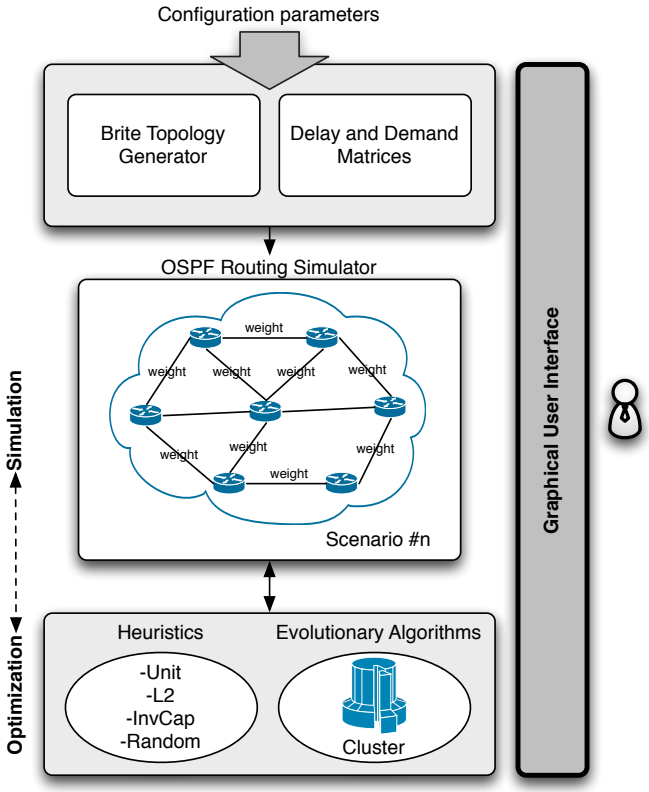

Fig. 1. High level components of the developed application.

This section reports on the development of an application that aims to simplify the use of the existing framework, hiding the complexity of the optimization tasks and making the administration job easier and more efficient. It allows the application of the developed optimization methods in real environments, like ISPs or large-scale networking domains. In this context, and as emphasized in Figure 1, it will act as a bridge between the TE platform and the user, hiding all the complexity associated with the platform and the corresponding optimization mechanisms devised in this work.

This section overviews some of its features: it allows the definition of the network topology and the definition of traffic and delay demand matrices that should be considered; based on such inputs, the administrator is able to simulate specific configuration scenarios and perform the optimization of the weights to be used to configure the OSPF routing protocol.

A. Requirements and Functionalities

The aim of developing a user-friendly TE based application imposed a set of requirements which were taken into consideration as guidelines for the implementation task. One of the main goals of the application was to provide an easy way to use the available optimization framework. This structured application hides the complexity of the problem from the user, possibly a network administrator without major programming skills, by creating an abstraction layer between the user and the system. Another major requirement is modularity, since there are different methods which can be applied on the optimization tasks. Some of those have already been implemented, but many others can still be developed and need to be plugged into the application, provided that those new functions meet the specified API.

To provide a solution for these issues, in the proposed application the user can handle different types of data, representing elements such as network topology, demands and delay requests, among others. Software functionalities or available actions are represented as *operations*. When calling an operation, the corresponding interface is launched and the input data objects are selected. After being triggered, an operation typically creates an instance of an output datatype. The required application features were mapped in operations, divided in distinct groups, easily accessible form a graphical interface. Those operations are listed below:

- **File**
  - New project from files - Allows to create a project from text files specifying the details of routers (*nodes*) and *links*.
  - Random Demands/Delay Requests - Enables the generation of random *Demands/DelayRequests* instances, based on a scale parameter (useful for benchmarking).
  - Load/Save - Load or Save data files (Demands, DelayRequests, OSPFWeights).
- **Simulation**
  - Weight Generation - This creates *OSPFWeights* based on the network topology using different heuristics (InvCap, Unit, L2, Random).

  - Simulate Scenario - Computes resulting *Loads* and *Delays*, based on the topology and selected *Demands*, *DelayRequests* and *OSPFWeights*.
- **Optimization** - Under this operation the different optimization algorithms (EAs, MOEAs) compute *OSPFWeights*, based on the selected parameters.
- **Serialization** - Load or Save objects using Serialization.

B. Implementation & Interfaces

The optimization framework and the developed application are fully implemented in the Java language, which has the advantage of being platform independent. The application is built on top of AIBench [18], a development framework resulting form a collaborative project between researchers from the University of Vigo and the University of Minho. AIBench is a lightweight and non-intrusive Java application framework that eases the connection, execution and integration of operations with well defined input/output. The platform was particularly conceived to facilitate the development of research applications based on general input–processing–output cycles, where the framework acts as the glue between each executed task.

Building applications over AIBench brings important advantages to both developers and users, given its design principles and architecture. AIBench based applications follow the Model-View-Control (MVC) design pattern, leading to units of work of high coherence that can be combined and reused easily. Furthermore, it is plug-in based; i.e. applications are developed adding components each containing a set of AIBench objects, allowing the reuse and the integration of functionalities of past and future developments based on AIBench.

The software development has been oriented to build a tool aimed at network administrators. The main goal in the development process was to provide good usability for the end user. Every AIBench application is divided into three kinds of components: operations, implementing the algorithms and data processing routines; data-types, storing relevant problem-related information; views, rendering data-types obtained from executed operations. Based on these concepts, a user-friendly GUI was developed.

Figure 2 briefly overviews some interfaces of the proposed application. The layout of the components can be observed in Figure 2 a). The clipboard (see Figure 2 f)) keeps all data objects created within the application in a logical hierarchy, being grouped by their datatypes. The root of this tree is the *ProjectBox* container, that keeps a list of instances, representing different problems. The components of a project are graphically shown in the form of explicit hierarchical containers, namely:

- The **Network Topology**, which includes information about nodes, edges, capacities, and all the network details;
- The **Demands Box** and **Delay Requests Box**, holding one or more instances of *Demands* or *DelayRequests*, respectively.
- **OSPFWeights**, hold sets of OSPF weights, one per each link of the network. These can be loaded from files or generated by the implemented operations and are grouped in the OSPFWeights box.
- Both the **ResultSimul** and **ResultOptim** aggregate the resulting information of the operations.

When the user double-clicks an object in the clipboard, the views of its datatype will be launched on the right side of the working area. Examples of two views of the network topology are shown in Figures 2 d) and 2 e). All available operations are easily accessible, through the menu in the top or by right clicking the item in the clipboard. Snapshots of simulation and optimization operation input dialogs are shown in Figures 2 b) and 2 c), respectively. In the first case, the administrator resorts to the platform to generate OSPF weights based on the *InvCap* heuristic (it generates weights that are inversely proportional to the link capacity) and to analyze the network performance resulting form such configuration strategy. In the second case, the administrator instructs the application to apply a multi-objective optimization algorithm to assist the OSPF weight setting task. As previously mentioned, operation outputs are grouped together in the respective *ResultBox*.

Figure 2 g) shows an example of a visualization window containing a set of non-dominated solutions obtained by the *SPEA2* MOEA algorithm, for a particular network scenario. Based on such visualization the administrator is able to perceive, for each specific solution, the resulting network performance (in this particular case measured in terms of congestion and delays). Figure 2 h) shows the OSPF weights table associated with a particular MOEAs solution, thus allowing the administrator to use a near-optimal routing configuration in the network domain. All operations are, as far as possible, default-oriented, hiding behind scenes their complexity (e.g. avoiding the definition of non-obvious parameters). Nevertheless, they allow more advanced users to fine-tune the parameters available to the operation.

The optimization part of the application makes use of JECoLi, an open-source Java-based library that was developed to implement the meta-heuristic optimization algorithms with a focus on EA based methods [14], both single and multi-objective. The graphical presentation of the network topology (e.g. see Figure 2 e)) was produced using Jung (http://jung.sourceforge.net/), a software library that provides a common and extendible language for the modeling, analysis, and visualization of data that can be represented as a graph or network, being adequate to be used by the proposed TE optimization application.

C. Availability

A preliminary version of the software is made available, together with other resources, in the home page accessible at http://darwin.di.uminho.pt/netopt, including the source code and different releases of the application, which is still under development. A full case study is detailed in the software documentation given in the project web site. The site includes

a set of How To's that detail how the major operations can be achieved with the application.

VI. Conclusion

The development of adequate traffic engineering methods is essential in modern network management, given the diversity of applications deployed over IP-based networks and the increasingly demanding QoS restrictions. In this work, we provide a study of the application of multiobjective EAs in the task of reaching the best link weights to configure the well known OSPF routing protocol. Several MOEAs were tested and compared to their single objective counterparts.

The results show that MOEAs can achieve good results in this task, when an integer representation, closer to the problem domain, and well designed reproduction operators are used. Still, in some cases, the overall performance of single objective EAs with a linear weighting scheme enables a better equilibrium between both objectives, This is probably due to the nature of both objective functions that are normalized within the same range. It is important to mention that MOEAs provide a less expensive alternative, in terms of computational effort, since they are able to obtain multiple solutions with distinct trade-offs between both objectives in a single run.

When faced with a selection between single and multiobjective EAs, the authors would recommend: (i) firstly, conduct a quick exploration of the space of possible solutions without having to specify any extra parameter, using a MOEA; it will provide multiple solutions with distinct trade-offs, from where the network administrator can select; (ii) if enough computational resources are available, further explore the most promising areas of the fitness space by using single objective EAs, tuning the weighting parameter accordingly.

Finally, this work reports on the development of a free software tool for the use of network administrators that enables users to take advantage of these optimization methods in real world scenarios. This application provides an user-friendly GUI that makes available a number of TE tasks.

As further work, it would be interesting to explore the integration of distinct classes of QoS demands in the proposed optimization model and to implement the support to those methods in the software tool.

Acknowledgment

The work is funded by FEDER through the Program COMPETE and by the Portuguese FCT through the project ref. PTDC/EIA-EIA/115176/2009. The work of Tiago Sa was supported by the grant UMINHO/BII/061/2009 also funded by Portuguese FCT.

References

[1] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," in *Proceedings of IEEE INFOCOM*, 2000, pp. 519–528.

[2] P. Sousa, M. Rocha, M. Rio, and P. Cortez, "Efficient OSPF Weight Allocation for Intra-domain QoS Optimization," in 6_{th} *IEEE International Workshop on IP Operations and Management, IPOM 2006*, G. Parr, D. Malone, and M. Foghlu, Eds. Dublin, Ireland: LNCS 4268, Springer-Verlag, October 2006, pp. 37–48.

[3] J. Moy, "RFC 2328: OSPF version 2," Apr. 1998.

[4] T. ThomasII, *OSPF Network Design Solutions*. Cisco Press, 1998.

[5] M. Sqalli, S. Sait, and M. Mohiuddin, "An enhanced estimator to multi-objective ospf weight setting problem," in *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, 2006, pp. 240–247.

[6] S. Sait, M. Sqalli, and M. Mohiuddin, "Engineering evolutionary algorithm to solve multi-objective ospf weight setting problem," in *Lecture Notes in Computer Science*, vol. 4304, 2006, pp. 950–955.

[7] P. Brostrom and K. Holmberg, "Multiobjective design of survivable ip networks," *Annals of Operations Research), volume = 147, number = 1, pages = 235-253, year = 2006*.

[8] C. C. Coello, *Recent Trends in Evolutionary Multiobjective Optimization*. London: Springer-Verlag, 2005, pp. 7–32.

[9] R. Ahuja, T. Magnati, and J. Orlin, *Network Flows*. Prentice Hall, 1993.

[10] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 269-271, 1959.

[11] J. Moy, *OSPF, Anatomy of an Internet Routing Protocol*. Addison Wesley, 1998.

[12] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi, *Qeueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications*. Wiley-Interscience; 2 edition, 2006.

[13] G. Syswerda, "Schedule optimization using genetic algorithms," in *Handbook of Genetic Algorithms*, L. Davis, Ed. Van Nostrand, 1991.

[14] P. Evangelista, P. Maia, and M. Rocha, "Implementing metaheuristic optimization algorithms with jecoli," in *International Conference on Intelligent Systems Design and Applications (ISDA 2009), Pisa, Italy*. IEEE Computer Society, 2009, pp. 505–510.

[15] J. Durillo, A. Nebro, F. Luna, B. Dorronsoro, and E. Alba, "jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics," Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos, Tech. Rep. ITI-2006-10, December 2006.

[16] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: Universal Topology Generation from a User's Perspective," http://citeseer.ist.psu.edu/article/medina01brite.html, Tech. Rep. 2001-003, Jan. 2001.

[17] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[18] D. Glez-Peña, M. Reboiro-Jato, P. Maia, M. Rocha, F. Díaz, and F. Fdez-Riverola, "Aibench: A rapid application development framework for translational research in biomedicine," *Computer Methods and Programs in Biomedicine*, vol. 98, no. 2, pp. 191–203, 2010.

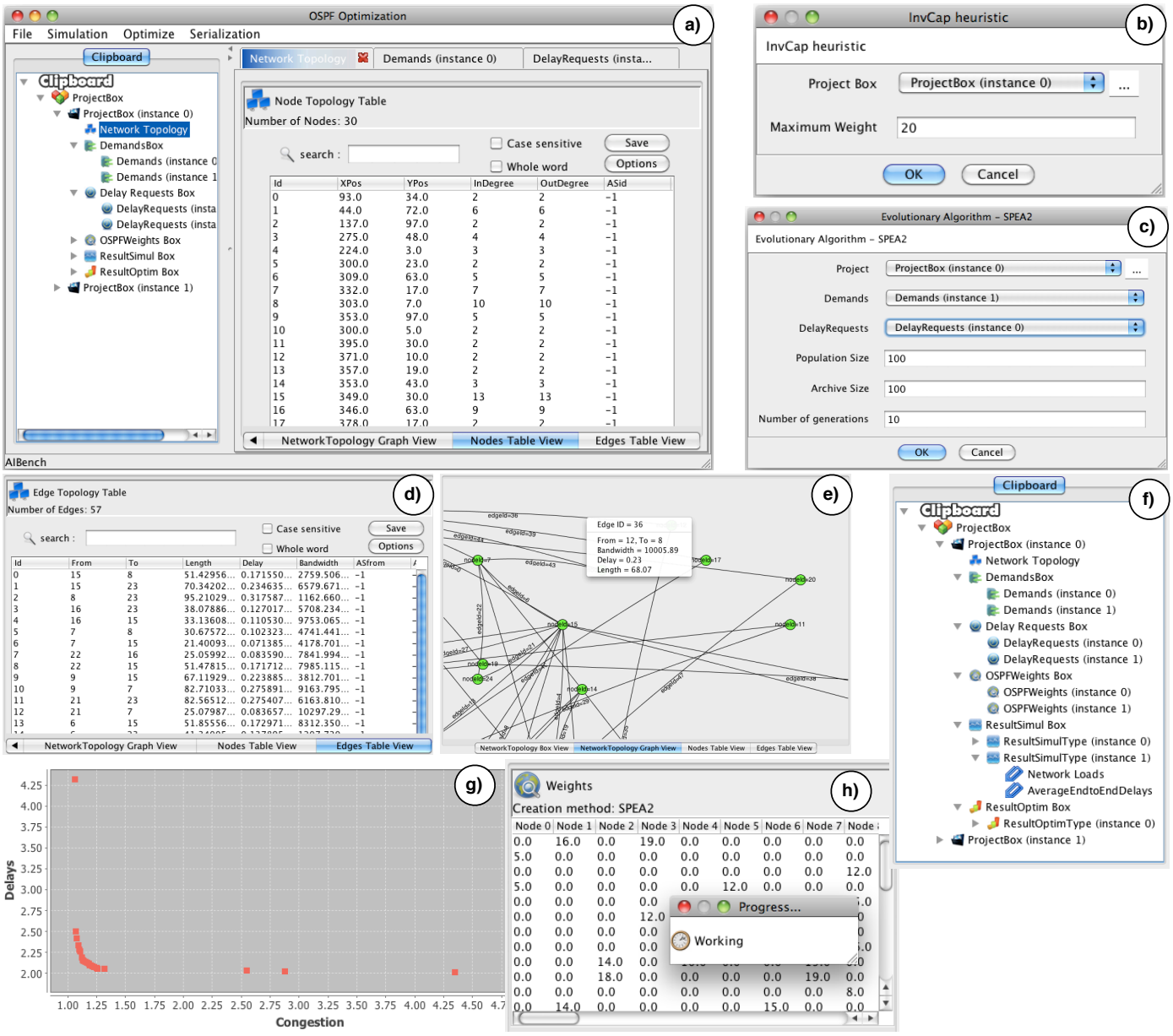

Fig. 2. Screenshots of the application: a) Main application window; b) Weight generation heuristic input dialog; c) Evolutionary Algorithm input dialog; d) Edge topology view; e) Another topology view; f) Clipboard displaying the main datatypes; g) Example of a view displaying the solutions obtained by MOEAs; h) Table with optimized routing configuration weights (from a specific solution of the ones obtained by MOEAs).