Robust Offline LSP Calculation for MPLS Networks

Paulo Roberto C. Estante, Edgard Jamhour

Pontifícia Universidade Católica do Paraná – PUCPR
Rua Imaculada Conceição, 1115 – Prado Velho – Curitiba – PR – CEP: 80215-901
Programa de Pós-Graduação em Informática Aplicada – PPGIA
{estantep,jamhour}@ppgia.pucpr.br

Abstract. *This paper proposes an offline method to compute LSPs (Label Switched Paths) for MPLS-based networks with support to path-protection. The objective of the method proposed in this paper is to generate optimal working and recovery paths for multiples demands subjected to capacity, delay and path constraints. The path constraints are imposed in order to achieve a robust LSPs planning, where the traffic demands could still be accommodated in the case of a single link failure in the network. By using a modified k-shortest path algorithm, we model the LPSs planning problem as search problem, which is solved using a genetic algorithm approach.*

1. Introduction

MPLS [Rosen et al. 2001] provides traffic-engineering capabilities to IP networks through the establishment of LSPs (Label Switched Paths) that are similar to the ATM virtual circuits. There are a few signaling protocols available to establish LSPs, such as RSVP-TE [Awduche et al. 2001] e CR-LDP [Jamoussi et al. 2002]. Presently, the RSVP-TE is the most usual signaling protocol. It has been implemented by major router vendors and it is being extensively used on production networks. RSVP-TE allows signalizing LSPs by using the link-state protocol routing information (e.g., OSPF or IS-IS) or by imposing explicit routes.

When the link-state protocol approach is used, the path taken by a LSP is automatically chosen using the distributed routing information. Link-state protocol extensions allow selecting distinct routes to the same destination by imposing path constraints (e.g., Kats et al 2003). Although link-state routing protocols distribute network related information, they do not carry the offered load (traffic demands) information. Therefore, achieving global optimization goals is usually impractical by using the link-state approach, because the routing decisions for a given flow does not take into account the other flows.

The explicit route feature is useful to the offline calculation of MPLS paths. Offline calculation is usually performed by a centralized system that knows the network topology and the entire predicted offered load. The offline approach has the advantage of allowing a globally optimal network design. In particular, when path protection is an issue, offline calculation allows predicting the load distribution behavior after a failure. Alternatively, signaling protocols such as RSVP-TE offer dynamic recovery facilities, where LSPs can be automatically re-signalized in case of failure. If necessary, the recovery LSP can take resources of lower priority LSPs already established using the

preemption mechanism supported by the signaling protocol. In most practical scenarios the dynamic recovery using the preemption mechanism is not recommended, as it can lead to service disruption on many LSPs.

This paper proposes an offline method to compute LSPs for MPLS-based networks with support to path-protection. The objective of the method proposed in this paper is to generate optimal working and recovery paths for multiples demands subjected to capacity, delay and path constraints. The recovery paths are planned in advance, in order to avoid the use of the preemption mechanism. A recovery path does not consume any network resource, as it is signalized only in case of failure of the corresponding working path.

The path constraints are imposed in order to achieve a certain degree of path protection. The degree of protection offered to the network can be expressed in terms of the number of simultaneous node or link failures supported without leading to link congestion. Because the multiple failure problem can be solved only to very redundant networks, in this paper, we limit the degree of protection to a single link failure, i.e., the traffic demands are required to be accommodated in the case of failure of "any" single-link in the network. Even in the case of a single link failure, multiple working paths may have to be switched to their corresponding recovery paths. Therefore, selecting recovery paths in order to minimize service disruption and links congestion during failures is a complex problem that can't be manually solved for complex network topologies. By using a modified k-shortest path algorithm, we model the LPSs with path protection planning problem as search problem with a multi-objective cost function, which is solved using a genetic algorithm approach. Multiple simultaneous link or node failures will be addressed as future works.

The remainder of this paper is structured as follows. Section 2 presents the related work concerning the IP/MPLS traffic engineering and the offline LSP calculation. Section 3 discusses the path protection concept and illustrates the necessity of including the path protection constraints into the LSP calculation problem. Section 4 models the LSP calculation as an optimization search problem and presents the proposed solution. Section 5 evaluates the proposed approach from logic and performance viewpoints using sample topologies. Finally, section 6 concludes this paper and presents future works.

2. Related Works

This paper proposes an offline method for traffic engineering on MPLS-based networks with support to path protection. In the literature, we find several works addressing the traffic engineering issue, and less frequently, the path protection issue. Because the number of works published in this domain is large, we have selected only the works that are closely related to our proposal.

Fortz and Thorup (2000) propose a method to control the routes selected by the OSPF protocol by optimizing the metrics assigned to network links. The method proposed by the authors is purely IGP (Interior Gateway Protocol), i.e., it does not employ MPLS. The authors take into account the projected demand (a traffic array) in order to achieve a better distribution of the traffic and avoid link congestion. The general routing problems is model as an optimization problem, where the cost function

penalizes unbalanced solutions, i.e., solutions that leads to high occupation rates of the network links. The cost function was defined in terms of a piece-wise linear increasing and convex function. The higher the occupation rate of link, the higher the cost assigned to the solution. Occupation rates above 100% are strongly penalized. In our work, we have employed a similar cost function to represent the traffic engineering component of our multi-operational objective function.

Mulyana and Killat (2004) propose an offline traffic engineering method to hybrid IGP/MPLS schemes. The authors address MPLS-shortcut [Shen and Smit 2004] scenarios, where LSPs are not required to be signalized from the ingress to the egress routers, but only to partial paths. LSPs shortcuts are seen as virtual links by the IGP. This approach offers a certain degree of traffic engineering to the network, while reducing the total number of LSPs used. Given traffic array with the projected demand and a maximum number of LSPs, the authors defines a search problem which consists in defining a set of LSPs shortcuts that minimizes the maximum link occupation rate. Similar to our work, the authors employ a genetic algorithm to solve the optimization problem. However, they do not address the path protection issue. Skivée et al. (2006) also addresses the IGP/MPLS scenario. Instead of a genetic algorithm, the authors employ a simulated annealing meta-heuristic to compute a nearly optimal set of LSPs that minimizes the congestion of the maximum occupied link and provides load balance. The method selects the LSPs from a predefined list that contains all allowed LSP candidates.

Several works in the literature propose offline traffic engineering methods for pure MPLS scenarios. Whilst most works defines the offline calculation of LSPs as an optimization problem, the techniques employed to determine the optimal LSPs are quite diverse. Lahoud et al. (2005) defines a linear programming framework to solve the multi-objective optimizing problem that minimizes congestion on links and resource consumption (by promoting shorter LSP paths), using a minimum number of LSPs. Erbas and Mathar (2003) formulates the problem of selecting optimal LSPs as a mixed integer problem solved using CPLEX 6.6. Similar to our work, the candidate LSPs are determined using a k-shortest path algorithm. The authors evaluate the effect of different objective function components: minimizing the route cost, increasing load balancing and reducing the total number of LSPs. This formulation is limited to simple topologies. The authors pointed out as future work developing a heuristic method for solving the problem on large network scenarios.

The protection of MPLS networks can be addressed from two viewpoints: path protection and local repair [Huang et al. 2002]. Path protection are end-to-end protection mechanisms where disjoint working and recovery paths are planned for each LSP demand, from the ingress to the egress router. The recovery path is always signalized by the ingress router. This method is considered slow due to the time taken for the ingress router to perceive the failure and to signalize a whole new path. Alternatively, in the local protection approach, a recovery segment can be signalized by an intermediate node, immediately upstream of the fault, reducing both, the failure perceiving time and the recovering time. The local approach, however, has the drawback of leading to a sub-optimal resource allocation after a failure. Mélon et al. (2003) proposed a real-time and decentralized method for LSP calculation that follows the local repair approach. The method reduces the resource allocation required to obtain

the protection by assuming that, at any given time, at most a single failure will occur in the network, and the recovering LSPs can share bandwidth. Giansante et al. (2004) proposes an offline centralized method that follows the path protection approach. The heuristic method presented by the authors is a three phase offline path protection algorithm, where recovery paths are calculated aiming to protect against link failures. The first phase defined the working paths using the best routes available, the second phase calculates recovery paths (using resources left from phase one). Because phases I and II employs only the best routes, the obtained solution can be non-optimal in terms of the number of traffic demands served. Therefore, the authors propose a third phase where a greedy algorithm is used to degrade primary and recovery paths in order to accommodate more traffic demands.

The offline method described in this paper employs several ideas already explored in other works such as the k-shortest path algorithms, a non-linear cost function to obtain load balance and a genetic algorithm to solve the LSP calculation as a search problem. However, it differs from the previous works as it supports the calculation of optimal LSPs from both viewpoints: path protection and traffic engineering. To the extent of our knowledge, there are fewer works following this approach. The work presented by Giansante et. all (2004) is a close match, but follows a totally distinct approach, because the resources are reserved to the recovery paths. Our method follows the Mélon et al. (2003) approach, where the single link failure assumption permits to achieve path protection without unnecessarily wasting network resources.

3. Path Protection Discussion

This paper proposes an offline method to compute LSPs paths for MPLS-based networks with support to path-protection. For each traffic demand (i.e., a certain amount of bandwidth to be reserved between two end points), our method is required to find two disjoint paths: a working path and a recovery path. Both paths can be configured in the MPLS ingress router by using the explicit route RSVP-TE feature. Most commercial routers are capable to automatically switch from the working to the recovery path in case of failure of, at least, one of the links of the working path.

Initially, no bandwidth is reserved for the recovery paths. However, in the case of a link failure, several LSPs may possibly switch to their respective recovery path. Our method needs to assure that, even in this case, no network link will exceed its bandwidth capacity. Therefore, in order to achieve a robust LSP planning, the definition of the working paths needs to leave network resources for the recovery paths.

The competition for resources between working and recovery paths is illustrated in Figure 1. Consider a sample scenario where there are five flow demands of 600Mbps (LSP2 to 6) and one flow demand of 400Mbps (LSP1) to be carried from router 1 to router 5. Figure 1 shows the optimal path planning considering only a load-balance objective among the network links. This path planning considers only working paths, and does not support path protection for all LSP demands. For example, if a link of the 1-3-5 path fails, one LSP demand can be switched to the 1-4-5 path. The available resources after the link failure and the path switching are illustrated in Figure 2. In spite

of existing 700Mbps of bandwidth available between routers 1 and 5, it is not possible to find an alternative path for the second LSP affected by the failure.

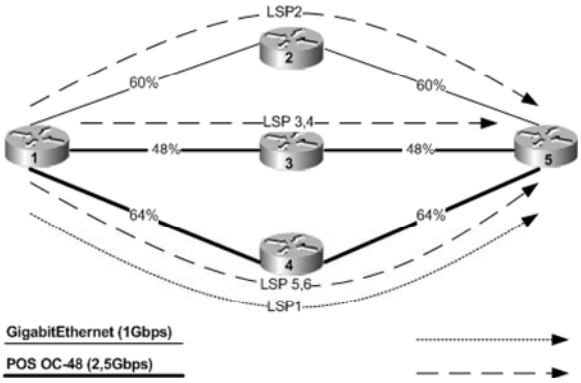

Figure 1. Optimal load-balance allocation considering only working paths

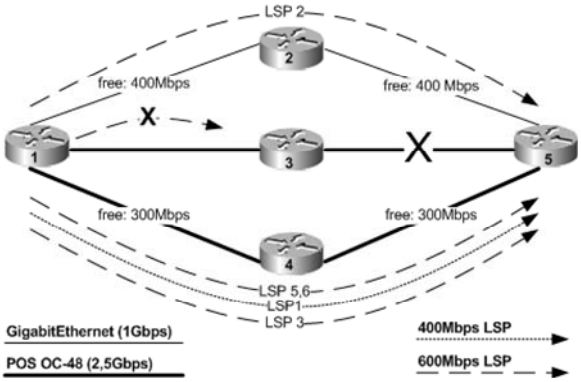

Figure 2. Scenario Topology simulating link 3-4 failure.

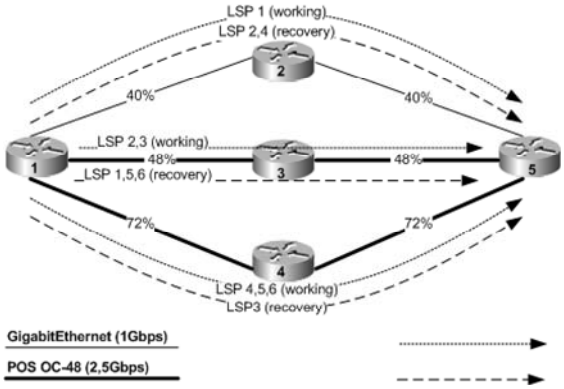

Figure 3. Scenario Topology with path protection.

The amount of free resources after the failure indicates that it is possible to find a solution offering path protection to all LSP demands. A possible solution is illustrated in figure 3. The 400Mbps demand is assigned to the 1-2-5 working path. Two 600Mbps demands are assigned to the 1-3-5 working path and three 600Mbps demands are assigned to the 1-4-5 paths. The corresponding recovery paths are shown in the same

figure. After a single link failure, the proposed solution can always accommodate the affected demands by switching to the recovery paths without exceeding the links capacity.

For simple scenarios, the problem of defining working and recovery paths can be manually solved. For complex network topologies and great number of LSP demands, it is necessary to employ computational techniques in order to solve this problem.

4. Proposal

The algorithm proposed to compute the LSP paths takes as input the network topology (including link capacity and latency), the projected demand to the network (a traffic array) and the maximum delay tolerated by each individual flow. The solution returned from the algorithm can be one of the following three cases.

- Case I: The algorithm was able to find working and recovery paths for all input demands without exceeding the link capacities in the case of no-link failures and in the case of a single link failure. The algorithm does not guarantee network protection to the case of multiple link failures.
- Case II: The algorithm was able to find working LSP paths, but some link failure situations leads to link overload. In this case, the algorithm outputs all the link failure situations that generates congestion on the network, e.g. should link a_{j1} fail, then link a_{j2} might have to transmit a certain (more than 100) percentage.
- Case III: The algorithm was not able to find the working LSP paths for all traffic demands, due to network topology or link capacity limitations. In this case, the algorithm will remove exceeding demands, one by one, from the lowest to the highest priority. After a demand is removed, the algorithm is executed again. The process of removing lowest priority demands is repeated until a solution is found.

4.1. Problem Formulation

In this work, we model the network topology as a graph. The topology graph **G** is defined by its vertex set, **V**, and its arc set, **A**. This is expressed as follows:

$$\mathbf{G} = (\mathbf{V}, \mathbf{A}) \quad (1)$$

$$\mathbf{V} = \{v_i \mid i = 1,2, ..., N\} \quad (2)$$

$$\mathbf{A} = \{a_j \mid j = 1,2, ..., M\} \quad (3)$$

Where N is the number of nodes and M is the number of links. The a_j link connecting the adjacent nodes *i* and *k* is usually noted by:

$$a_j = (v_i, v_k) \quad (4)$$

Each a_j link, ($a_j \in A$), has the following attributes: a capacity c_j (expressed in Mbps) and a latency l_j (expressed in ms). A traffic requirement t_k between nodes v_i and v_j is represented by a tuple containing the bandwidth requirement d_{ij} (corresponding to the demand to be carried from the origin *i* to the destination *j*) and the maximum end-to-end delay (expressed in ms).

$$t_k = (b_{ij}, d_{ij}) \quad (5)$$

The traffic array **T** contains the point-to-point requirements between origin/destination (OD) pair of nodes:

$$\mathbf{T} = \{ t_k = (b_{ij}, d_{ij}) |\ i, j \in [1,N] \} = \{t_1 \ldots t_k \ldots t_D\} \quad (6)$$

where: D = number of traffic demands

The problem consists in determining the optimal LSP paths (working and recovery) corresponding to each traffic requirement t_k. The LSP paths corresponding to a $t_k = (b_{ij}, d_{ij})$ requirement is noted as follows:

$$lsp_k = (\mathbf{W}_k, \mathbf{R}_k) \quad (7)$$

The paths $\mathbf{W}_k$ and $\mathbf{R}_k$ correspond to a sequence of arcs, and can be represented as follows:

$$\mathbf{W}_k = \left(a_{w1} \ \ldots \ a_{wj} \ \cdots \ a_{wJ}\right) \quad wj \in [1, M] \quad (8)$$

$$\mathbf{R}_k = \left(a_{r1} \ \ldots \ a_{rj} \ \cdots \ a_{rJ}\right) \quad rj \in [1, M] \quad (9)$$

The sequence $\mathbf{W}_k$ and $\mathbf{R}_k$ are required to be disjoint, i.e., they have no common link, i.e.,

$$a_j \in \mathbf{W}_k \Rightarrow a_j \notin \mathbf{R}_k \wedge a_j \in \mathbf{R}_k \Rightarrow a_j \notin \mathbf{W}_k \quad \forall j \in [1, M] \quad \forall k \in [1, D] \quad (10)$$

Let **lsp** = $\{lsp_k \mid k=1..D\}$ be a candidate solution corresponding to all demands the traffic array **T**, defined in (5). A solution LSP is considered feasible if:

- The resulting traffic load assigned to any link does not exceed the link capacity.
- The resulting end to end delay corresponding to a LSP path does not exceed the delay traffic requirement.
- In the case of failure of any single link, and the switch of the affected LSPs to the recovery paths, the resulting load assigned to any link does not exceed the link capacity.

An optimization problem can be defined in terms of a cost function f_c(**lsp**). For a candidate solution **lsp**, f_c(**lsp**) represents the "level of rejection" of the corresponding solution. In order to be valid a solution **lsp** must be feasible, i.e., it must satisfy a set of constraints related to the nature of the problem being solved. Let **LSP** be the space of feasible solutions that satisfy the all the problem constraints. Then, the optimization problem consists in finding the element **lsp**=**lsp***$\in$ **LSP** that minimizes the f_c(**lsp**) function. Mathematically, it can be expressed as follows:

$$\mathbf{lsp}^* \in \mathbf{LSP} \quad \text{and} \quad f_c(\mathbf{lsp}^*) = \inf_{\mathbf{lsp} \in \mathbf{LSP}} f_c(\mathbf{lsp}) \quad (11)$$

In this work, we considered a multi-objective cost function that addresses the following issues:

a) We penalize candidate solutions that lead to overload links, considering the situation where the network has no link failure (only working paths are used). This is useful for accommodating new traffic demands without modifying the already assigned ones. When the load assigned to a link exceeds its capacity, the corresponding solution is severely penalized. As noted by Fortz and Thorup (2000), the load balance can be achieved by adopting a convex increasing non-linear cost function. We note this component of the f_c function as f_{ca} , and it is defined as follows:

$$f_{ca}(\mathbf{lsp}) = \sum_{j=1}^{M} \gamma(\rho_j) \tag{12}$$

where: $$\gamma(\rho_j) = \begin{cases} \rho_j{}^3 & \text{for } 0 \le \rho_j \le 1 \\ 100 \cdot \rho_j{}^3 & \text{for } 1 < \rho_j \end{cases}$$

and: ρ_j = occupation rate of the a_j link

b) We strongly penalize candidate solutions when, after a link failure, the resulting load exceeds the capacity of at least one link. The traffic load is evaluated considering that all LSPs affected by the link failure have switched to their respective recovery paths. We note this component of the f_c function as f_{cb}.

$$f_{cb}(\mathbf{lsp}) = \frac{1}{M} \sum_{k=1}^{M} f_{cfailure}(\mathbf{lsp}, k) \tag{13}$$

$$f_{cfailure}(\mathbf{lsp}, k) = \sum_{j=1,\, j \neq k}^{M} \gamma(\rho_j) \tag{14}$$

where: $f_{cfailure}$ is the cost function when the k-th link fails

c) Just avoiding network congestion by better load distribution leads to paths being longer, which results in extra bandwidth consumption [Lahoud et al. 2005]. We penalize candidate solutions when the working LSP path does not correspond to the optimum path (i.e., the path determined using the dijkstra algorithm in the case of no link failure). We note this component of the f_c function as f_{cc}.

$$f_{cc}(\mathbf{lsp}) = \frac{1}{D} \sum_{k=1}^{D} f_{ref\_dijkstra}(lsp_k) \tag{15}$$

$$f_{ref\_dijkstra}(lsp_k) = \frac{\text{delay}(lsp_k)}{\text{delay}(lsp_{k,dijkstra})} \tag{16}$$

where: delay(lsp_k) is the delay of the candidate lsp for the demand t_k

delay(lsp_k,dijkstra) is the delay considering the optimum path

The cost function assigned to a solution lsp is the defined as follows:

$$f_c(\mathbf{lsp}) = f_{ca}(\mathbf{lsp}) + \alpha \cdot f_{cb}(\mathbf{lsp}) + \beta \cdot f_{cc}(\mathbf{lsp}) \tag{17}$$

The parameters α and β are weigh factors, and can be used to determine the relative importance of the components of the cost function.

4.2. Candidate LSPs Computation

Our proposal solves the optimization problem defined in (17) using a search algorithm. A common approach to adapt the path definition to a search algorithm consists in defining a set of candidate solutions for each LSP demand using a k-shortest path algorithm (see Mulyana and Killat (2004) and Skivée et al. (2006), for example). That means that for each LSP demand, besides the shortest path defined by the dijkstra algorithm, alternative (longer) paths will also be considered. The search method is responsible for choosing a path for each LSP demand among these candidates.

In our approach, the k-shortest path algorithm is used to define a set of working paths and recovering paths. The multiple working paths are obtained by removing links from best path found (dijkstra), one by one. In some situations, removing only a link at time from the best path is not enough to provide several distinct paths. This happens if the alternative solution is disjoint with respect to the best path. In order to overcome this limitation, after all links from the best path were removed, our algorithm initiates a second round where a link from the best path and a link from the second best path are simultaneously removed. This assures that, if the topology allows, at least three candidate working paths are found for each LSP demand.

The recovery paths are computed in a similar fashion to the working path computation. For determining a recovery path, the k-shortest path algorithm is computed after removing all links of the respective working path. Figure 4 shows all possible candidates for our example topology (Figure 3). Each flow demand t_k has a list of candidate LSPs. Each candidate LSP is composed by a working and a recovery path. Note that we may have different candidates for the same demand with equal working or recovery paths, but at least one the paths must be distinct.

Candidate 0	W	1 - 2 - 5
	R	1 - 3 - 5
Candidate 1	W	1 - 2 - 5
	R	1 - 4 - 5
Candidate 2	W	1 - 3 - 5
	R	1 - 2 - 5
Candidate 3	W	1 - 3 - 5
	R	1 - 4 - 5
Candidate 4	W	1 - 4 - 5
	R	1 - 2 - 5
Candidate 5	W	1 - 4 - 5
	R	1 - 3 - 5

Figure 4. Possible candidates for each demand t_k

The metric used for the Dijkstra algorithm is the link's propagation (and internal routing/switching) delay. Links that cannot transport a determined demand are excluded from the graph. Also, a candidate LSP is only considered for calculations if both working and recovery paths honors the maximum delay defined for the flow demand. This is useful when for calculating LSPs for delay sensitive demands (e.g., satellite links may be avoided for voice or multimedia traffic). If delay is not an issue on a given network, this metric can be replaced by a simple hop count.

4.3. Genetic Encoding and Evolution Cycle

A genetic algorithm (GA) is a search method based on natural evolution, where a population of candidate solutions goes through an evolution cycle till a convergence criterion is reached. The method creates a population of individuals (solutions), ranking them according to their fitness (objective function). Next, one performs the crossover operation among the best ranked individuals, where new individuals are formed by combining genes (characteristics) from their parents. The new individuals are used to replace the individuals with worse fitness, creating a new generation of individuals. This completes an iteration of the algorithm. This iteration process is repeated for a certain number of generations. The idea is that best ranked individuals will provide the better characteristics to the future generations, improving the solution to the optimization problem.

Besides the crossover function, GAs can also perform mutation on the individuals. This operation randomly changes genes on the individuals (possibly introducing new characteristics). The mutation operation helps the algorithm to explore new areas in the solution space, as well as distracting the algorithm from converging [Haupt and Haupt 1998], avoiding the solution from being trapped in a local minima.

For the population encoding, we have used a similar model than Mulyana and Killat (2004), where each individual represents a candidate solution (i.e., an **lsp** set in equation 11) for the entire traffic array. Each gene of the individual indicates a candidate solution, lsp_k, for a specific t_k demand. Figure 5 illustrates the encoding of an individual considering a traffic array with six demands. In this example, there are six candidates for each demand computed using the k-shortest path algorithm. Each gene of an individual corresponds to an index that point to a specific solution among these candidates.

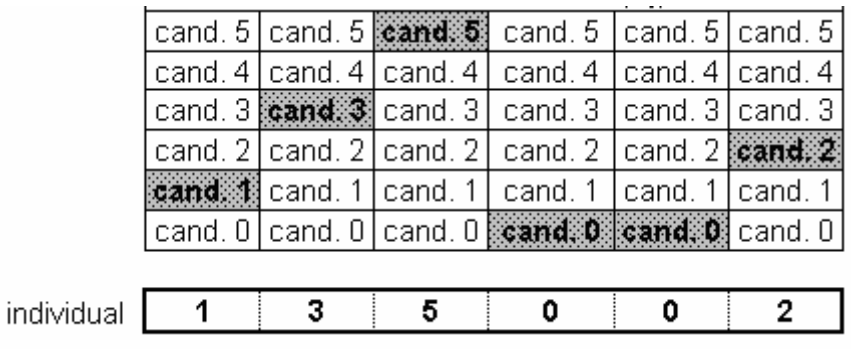

Figure 5. An example individual (chromosome) encoding

First, the algorithm randomly generates an initial population. Then, at each iteration, all individuals are ranked according to their cost function. A percentage of the best ranked individuals are used as the parents for the crossover operation. This percentage is noted as “parent rate” and it is a tuning parameter for the algorithm. Every crossover operation generates two offspring, being the second one the complement from the first one (i.e., if offspring #1 inherits the first gene from parent #2, then offspring #2 will inherit the first gene from parent #1, and so on). The crossover mechanism used was the uniform-crossover, where each offspring has equal probability of inheriting each gene from one of the two parents. After the crossover, a mutation operation is randomly applied to a percentage of the genes of the population, excluding the individuals that were used as parents for new offspring. This percentage is noted as “mutation rate”.

The convergence criteria were defined in terms of the maximum number of generations or a specific number of generations reached without any improvement. In this work we have adopted the convergence criteria of 50 consecutive generations without improvement or 500 generations.

5. Evaluation

We evaluate our method from two distinct viewpoints: logic and performance. From the logic viewpoint we are interested in determining if the algorithm could output a solution that respects the constraints and the objective function. From the performance viewpoint we are interested in evaluate if the algorithm was able to output a solution for a large topology and traffic demand in a reasonable computation time.

To validate the algorithm from the logic view point we used the simple scenario presented in section 3 (Scenario I). To validate the algorithm from the performance viewpoint we create a fictitious scenario based on the international research network GÉANT, assuming a full mesh demand among all network routers (Scenario II).

5.1. Scenario I

The topology and the demands used on this test were exactly as presented on Section 3 (Figure 1 to 3). Running a C program with the algorithm on a Linux-based system (AMD Opteron 2.2GHz), with parameters: α=1 and β = 0.1, the algorithm found the best solution in the fourth generation, and converged after 54 generations. The solution returned was the same as presented on Figure 3. Population size used was 50 individuals, parent rate being 20% and 2% mutation rate. The total running time (including candidates LSP computation and optimization until convergence) was 0.045 second.

5.2. Scenario II

The GÉANT network topology (Figure 6) had to be slightly modified in order to provide a more significant scenario for testing the algorithm performance. In order to increase the number of possibilities for creating recovery paths we have inserted six bidirectional links on the original network topology, as denoted on Table 1. Also, all links inputted were at least 155Mbps. We have used unitary hop count as the delay metric.

We have assumed a full mesh scenario, where a bidirectional demand was assigned to every pair of nodes in the topology. Independent LSP paths are computed for each direction of the traffic demand. The amount of bandwidth assigned to each demand was defined according to the equation 18. This equation normalizes the bandwidth requirements for each LSP according to the smallest capacity of each node in the pair. The node capacity corresponds to the summation of its all link capacities. This approach allows routers with low capacity links to handle full-mesh LSPs with every other router on the topology. The constant 1/4 was used in order to leave some free throughput on the routers so they could serve as transit for other LSPs.

$$t_{ij} = \frac{Min(throughpt_i, throughput_j)}{4 * node\_count} \tag{18}$$

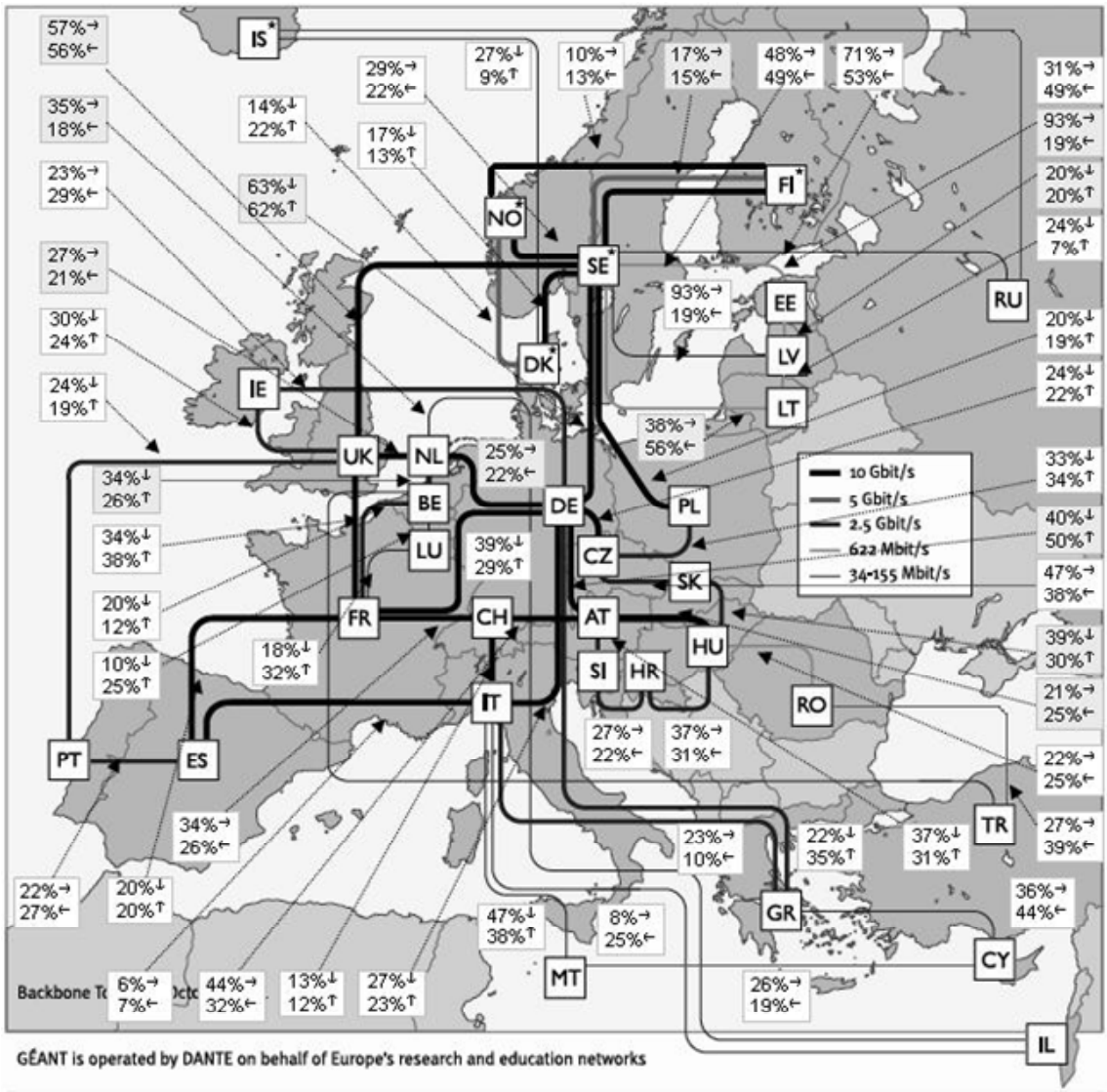

Figure 6. The GÉANT topology

Table 1. Links added to the GÉANT network topology for the test

	IS-RU	NO-FI	EE-LV	LV-LT	RO-TR	CY-MT
Speed	155Mbps	10Gbps	622Mbps	622Mbps	155Mbps	155Mbps

As the number of nodes is N=33, the traffic demand for this scenario consists of D=1056 LSPs demands. The number of links (arcs) is M=53. Running the program on the same machine as the one from the example 5.1, the program took 0.9 second to compute 6124 LSP candidates (average of 5.8 LSP candidates per demand). The population size used was 50 individuals, parent rate being 20% and 2% mutation rate (identical to the scenario 1). Using $\alpha = 1$ and $\beta = 0.1$, the algorithm was found a feasible solution (for working paths) on the 28th generation (taking about 30 seconds of program running time). After that, solutions have been continuously improved until the stop criterion was achieved after the 500 generations. The total running time for this scenario was 8m56s.

The solution found for this scenario corresponds to a Case II, i.e., feasible working paths were found for all demands, but not all links could be protected. That indicates that in order to protect all links, the number of LSPs should be reduced. For the working paths, the algorithm was able to distribute traffic with an average of 29.1% of occupation rate for the links. This result is illustrated in Figure 6, where the occupation rate at each direction of the bidirectional links is indicated. The solution computed did not tolerate the failure of 13 among the 53 links in the network topology. These links are indicated with gray demands in the figure. In average, during a failure, 1.31 links have their capacity exceeded due the switching to the recovering paths. The occupation rate for these links was about 136%, in average.

6. Conclusion

This paper presented a method for computing optimal LSPs for MPLS-based networks. The optimization approach uses a multi-objective cost function that permits the traffic engineer to balance the importance between the quality of the working paths and path protection when performing the LSP calculation. The evaluation scenarios indicated that the proposed method is scalable with respect to the network topology and the size of the traffic array, as it was capable of computing a fictional scenario with more than 1 thousand LSPs in less than 10 minutes. That indicates that this method can be expanded for computing solutions with a small number of simultaneous link and node failures. However, as the computation of the f_{cb} component of the cost function (see equation 13) is expected to increase exponentially with the number of simultaneous failures, other approaches should be researched in order to treat scenarios with several simultaneous failures.

References

Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V. and Swallow G. (2001) "RSVP-TE: Extensions to RSVP for LSP Tunnels", IETF RFC 3209.

Awduche, D., Chiu, A., Elwalid, A., Widjaja, I. and Xiao, X. (2002) “Overview and Principles of Internet Traffic Engineering”, IETF RFC 3272.

Erbas, S. and Mathar, R. (2003) “A Multiobjective Offline Routing Model for MPLS Networks”, Proc. of the 18th International Teletraffic Congress.

Fortz, B. and Thorup, M. (2000) “Internet Traffic Engineering by Optimizing OSPF Weights”, IEEE INFOCOM.

Giansante, E., Iovanna, P., Oriolo, G., Pascali, F.,.Romagnoli, A. and Sabella, R. (2004) “Offline Protection Algorithm in a MPLS-based scenario”, Workshop on Traffic Engineering, Protection and Restoration for NGI.

Haupt, R. and Haupt S. (1998) “Practical Genetic Algorithms”, John Wiley & Sons, p. 41-42.

Huang, C., Sharma, V., Owens, K., Makan, S., “Building reliable MPLS networks using a path protection mechanism”, IEEE Communication Magazine, pp. 156-162, March 2002.

Jamoussi, B., Ed., Andersson, R., Callon, R., Dantu, R., Wu, L., Doolan, P., Worster, T., Feldman, N., Fredette, A., Girish, M., Gray, E., Heinanen, J., Kitly, T. and A. Malis, (2002), "Constraint-Based LSP Setup using LDP", IETF RFC 3212.

Katz, D., Kompella, K., Yeung, D. (2003), "Traffic Engineering (TE) Extensions to OSPF Version 2", IETF RFC 3630.

Lahoud, S., Texier, G. and Toutain, L. (2005) “Offline Flow Allocation for Traffic Engineering in MPLS Networks”, LANMAN Greece.

Mélon, L., Blanchy, F. and Leduc, G. (2003) “Decentralized Local Backup LSP Calculation with Efficient Bandwidth Sharing”, IEEE ICT.

Mulyana, E. and Killat, U. (2004) “Optimization of IP Networks in Various Hybrid IGP/MPLS Routing Schemes”, 3rd Polish-German Teletraffic Symposium.

Rosen, E., Visnathan, A. and Callon, R. (2001) “Multiprotocol Label Switching Architecture”. RFC 3031, IETF.

Shen, N. and Smit, H. (2004) “Calculating Interior Gateway Protocol (IGP) Routes Over Traffic Engineering Tunnels”, IETF RFC 3906.

Skivée, F., Balon, S. and Leduc, G. (2006) “A scalable heuristic for hybrid IGP/MPLS traffic engineering – Case study on an operational network”. ICON '06. 14th IEEE International Conference on Networks, 2006.