Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Computer Networks 43 (2003) 557–572
COMPUTER NETWORKS
www.elsevier.com/locate/comnet

Prompt partial path optimization in MPLS networks

Dániel Orincsay *, Balázs Szviatovszki, Géza Böhm

Ericsson Research, Traffic Analysis and Network Performance Laboratory, P.O. Box 107, 1300 Budapest, Hungary

Received 15 November 2002; received in revised form 14 April 2003; accepted 14 April 2003

Responsible Editor: J. Crowcroft

Abstract

In this paper we evaluate approaches that allow the re-routing of a set of already established label switched paths (LSPs) in multiprotocol label switching (MPLS) networks when a new LSP demand faces constrained shortest path first failure. Two optimization algorithms are proposed: one is based on an integer linear programming formulation of the problem, while the other one is a heuristic method based on Dijkstra's shortest path first algorithm. We analyze the efficiency of our algorithms in a dynamic network environment. The simulation experiments show that our method significantly increases the total network throughput. We also investigate the limitations of the applicability of proposed methods considering their running time.

Keywords: MPLS; Traffic engineering; Routing; Optimization

1. Introduction

Providing Internet protocol based interconnection services in a competitive environment fortifies the importance of expanding the customer base of an Internet service provider (ISP), which requires the use of traffic growth forecast and economical resource provisioning. Accomplishing these in the most cost-effective way requires a central infrastructure consisting of traffic monitoring, trend analysis, capacity planning, routing evaluation and optimization functions. The applied technology in these components may differ between service providers; one ISP may use multiprotocol label switching (MPLS) overlay based routing optimization [13], while another may prefer link weight optimization of link-state routing protocols (e.g., OSPF) as the traffic engineering (TE) solution [8,17]. This study concentrates on TE methods based on explicitly routed paths with bandwidth requirements. Although MPLS terminology is used throughout the paper, the investigated methods can be applied in any technological environment enabling source routing and bandwidth reservation along the paths.

For shorter term label switched path (LSP) provisioning a local distributed and automated mechanism is needed, that can quickly react upon arriving LSP establishment requests. Generally, the *constrained shortest path first* (CSPF) algorithm implemented in label edge routers (LERs) is

* Corresponding author. Tel.: +36-1-437-7179; fax: +36-1-437-7767.

E-mail addresses: daniel.orincsay@eth.ericsson.se (D. Orincsay), balazs.szviatovszki@eth.ericsson.se (B. Szviatovszki), gbohm@archidata.hu (G. Böhm).

1389-1286/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S1389-1286(03)00290-1

used for this purpose. Various CSPF algorithms can be found in the literature [1,14–16,23,24,26]. Each proposal aims at optimizing or improving the network performance by concentrating on different tasks, e.g., balance the load in case of low and middle network traffic, or minimize the blocking in case of network overload. To achieve this, CSPF algorithm proposals differ in the applied metrics and the ordering in Dijkstra's shortest path algorithm. For example the first constraint of the *shortest–widest path* algorithm [26] is to find a path with maximum bottleneck bandwidth, and when there are more such widest paths, it chooses the one which is the shortest. The *widest–shortest path* algorithm [1] intends to minimize the used resources of flows, while as a secondary objective aims at avoiding the loaded network segments. The first metric of the *discrete link cost* algorithm [23] is hop count, similarly to the previous case, however, the method enables the adjustment of various parameters. The *minimum interference routing* algorithm tries to prepare for potential future demands, by using one-level metric with a complex weight function [15]. For a more detailed overview of different CSPF methods, see [24].

The common property of the discussed CSPF algorithms is that they route the new LSP demand on arrival and do not change its path any more. Although the routing methods aim at achieving the best network utilization, decisions that seem to be good considering the current circumstances can result in a lower network performance on the long term. The well-known solution for this problem is the *global optimization* of LSPs with the help of a centralized off-line network optimization tool [2,3,7,10,13,19,20,25]. This optimization can be performed periodically, when the network state deviates considerably from the optimal state. The complete optimization of all LSPs—depending on the number of paths and size of network—can take relatively long time on a dedicated server [13]. Furthermore, this action often causes the change of almost all LSP routes, which has two undesirable effects: (1) increased signaling overhead compared to the normal operation, and (2) a transitional state—due to the major path restructuring—in which some LSPs may have to be torn down before their new routes can be established [11]. For these reasons it is not advisable to perform global optimization too frequently, namely, it should be considered on a larger time-scale, e.g., few days or a week.

The above mentioned methods represent two options in the space of possible routing approaches. Obviously, CSPF is the simplest since it does not affect the paths of already established LSPs, while global optimization—by rerouting possibly all LSPs in the network—is the most complex. However, above these approaches, there are other options as well. One could intend to introduce a constraint to a global optimization algorithm, and restrict the optimization to a predefined maximum number of LSPs. Alternatively, CSPF could be enhanced as well, by allowing the re-routing of few already established LSPs. This extension would be used in case of a CSPF failure due to the shortage of reservable bandwidth in the network. We have searched the literature for such algorithms, however, we found that these research directions are yet unexplored.

In this study we intend to provide algorithms for the latter case, namely to trigger a *prompt partial path optimization* (PPPO) in order to route the new LSP when CSPF failure occurs. In this method, instead of re-routing all LSPs of the network, we intend to re-route only minimum number of LSPs. As a consequence, reservable capacity may be released along the possible routes of the new LSP demand, so that it can be established. Since this action can be performed relatively often, a fast optimization algorithm is necessary that affects only few LSPs' established paths in the network. To underline our assumption about such a dynamic control of transmission paths we mention a dynamic, on-demand VPN service where after requesting a VPN connection its provisioning should take, e.g., less than an hour. In this environment, most probably CSPF would be used to set up new requests, while global path re-computation would be performed periodically (e.g., daily) to assure optimal resource utilization. In case CSPF based path setup fails, the service provider could use our proposed PPPO method, i.e., re-route some already established LSPs instead of blocking the new LSP demand. Although dynamic LSP setup is not

wide-spread in current MPLS networks, some studies propose dynamic concept. To balance the utilization of network links a dynamic approach can be used in case long-lived traffic flows are detected on-line [22]. This proposal uses on-line path computation that can adapt to changes promptly. A dynamic architecture for accessing λ paths is another current research topic [6]. Here the basic concept is to allow the management and control of wavelengths from the edge of the network. Once the path setup is triggered dynamically from the edge of the network, the importance of automated CSPF like methods will increase, while global optimization methods will probably have relatively less significance. This underlines our research direction to develop fast local partial path optimization methods.

In [12] we examined the basic case of prompt partial path optimization when the number of LSPs to be re-routed is limited by one, in other words we investigated how the re-routing of a single LSP can help the establishment of a new LSP demand. Afterwards, in [18] we extended the problem to the cases of protected LSPs as well as explicitly defined loose and strict hops. It was shown in both papers that prompt partial path optimization can efficiently extend the CSPF algorithm, resulting in a higher network performance on the long term. Therefore, in this paper we expand our study to a more general case when we allow the re-routing of more than a single LSP. The difficulty of the approach is to explore the state space, because as the number of re-routable LSPs increases, the number of cases for which the routability should be checked explodes. As the main part of the paper, in Section 2 we propose two methods that can decide whether an LSP subset is re-routable or not. The first method uses an integer linear programming (ILP) formulation, while the other one is based on a recursive Dijkstra algorithm. Both algorithms are general in the sense that they could be applied for any n re-routable LSPs. For the ILP method we present heuristic filters that can identify candidate re-routable LSP subsets, in order to decrease the running time. However, during our numerical investigations it turned out that—taking into consideration that a real-time application needs to have acceptable running time—it is worth examining only such cases when maximum three LSPs are allowed to be re-routed.

The rest of the paper is organized as follows. In Section 2 we present our new prompt partial path optimization algorithm in details. Then, we describe the simulation environment used for performance evaluation, and analyze the obtained results in Section 3. This section includes discussions of issues related to the applicability, scalability and long-term performance of the proposed optimization algorithms, in various network situations. Finally, we conclude our work in Section 4.

2. Proposed new algorithms

In this section we present our new algorithms that realize the concept of PPPO. As we have discussed, the task of this new traffic engineering method is to re-route several already established LSPs in order to fit the new LSP demand into the network. Since this job can be done relatively frequently in comparison with, e.g., a global optimization of the network, it seems to be natural to limit the number of changeable paths. In [12] we investigated the simplest case when we allow to re-route only a single LSP. It is obvious that by increasing the limit on the number of LSPs to be re-routed, the problem itself becomes much more difficult, namely the size of state space and the complexity of problem formulation expands rapidly. In our study we decided not to specify this limit explicitly, but to adjust it based on the practical running time values. Thus, a realistic limit can be provided, considering that these algorithms should be applied in connection with real-time routing decisions.

As mentioned, PPPO is triggered by the failure of CSPF algorithm that occurs because of the lack of reservable bandwidth between the ingress and egress router of the new LSP demand. The basic process of PPPO is the following:

Step 0: $n \leftarrow 1$.
Step 1: Try to route the new LSP request by re-routing n previously established LSPs.

Step 2: If feasible solution is found STOP with success.
Step 3: If $n = \max\_n$ STOP with failure.
Step 4: $n \leftarrow n + 1$. Go to step 1.

Note that the upper limit on the number of re-routable LSPs (max_n) can be adjusted, enabling to control the running time of the algorithm in a sense. In case of a real-time application a more suitable implementation would be to build a time limit into the algorithm that is being continuously examined, thus when the particular running time exceeds this limit the algorithm stops with failure.

The core of the algorithm (see step 1) is such a method that is able to re-route one or more previously established LSPs in order to route the new LSP demand. In this study we propose two methods that realize this function. The first one called *integer linear programming based simultaneous re-routing* (ILP-SR) formulates the task of path selection for the new demand and the LSPs to be re-routed as an *integer linear program* (see details in Section 2.1). The main idea behind our second method called *Dijkstra's algorithm based recursive re-routing* (DA-RR) is that it tries to re-route LSPs one by one with the help of Dijkstra's shortest path algorithm (see Section 2.2). The two approaches are common in the sense that they do not focus on finding the best solution, their only purpose is to search for the first feasible solution, which gives their heuristic nature. As another common attribute, both methods provide the possibility of applying various link cost functions during the optimization.

It is obvious that increasing the number of LSPs to be re-routed, the number of different cases to be examined grows quickly, namely in case of n re-routable LSPs the number of possible LSP subsets that have to be checked is

$$\binom{|\mathbf{P}|}{n} = \frac{|\mathbf{P}|!}{n!(|\mathbf{P}| - n)!},$$

where $\mathbf{P}$ denotes the whole LSP set. Supposing a large set of already established LSPs and small n value, $\binom{|\mathbf{P}|}{n} \sim |\mathbf{P}|^n$ is valid. In other words, for a fixed LSP set $\mathbf{P}$ the number of instances to be examined is in exponential proportion with the number of LSPs to be re-routed n. Furthermore, we suppose that the number of already routed LSPs is in quadratic proportion with the network size, i.e., the number of network nodes. Thus, for example if $n = 3$, the size of the state space specified by the problem is proportional to the sixth power of the number of network nodes.

The examination of all possible LSP subsets would be a very time-consuming task, making the running time of the algorithm critical for larger network instances. In order to reduce the necessary running time, various *filters* can be applied, with the intention of facilitating and speeding up the examination of the various cases. Applying ILP-SR the task of filtering is to prevent the algorithm from formulating the routing problem of such LSP subsets that surely or probably cannot be re-routed. On the other hand, applying DA-RR—that re-routes the LSPs one by one—the filtering function has to decide which LSP is worth re-routing. Thus, many branches of the recursion can be eliminated. The detailed description of filters will be included in the presentation of the proposed routing methods (see Sections 2.1 and 2.2).

2.1. Integer linear programming based simultaneous re-routing

In this section we present our first proposal called *integer linear programming based simultaneous re-routing* (ILP-SR) method that realizes the core function of PPPO. The method consists of the following steps:

Step 0: Select first LSP subset (containing n LSPs).
Step 1: If the selected LSP subset can be filtered go to step 4.
Step 2: Try to route the new LSP demand and re-route the selected LSP subset with the help of ILP.
Step 3: If feasible solution is found STOP with success.
Step 4: If there is no more LSP subset STOP with failure.
Step 5: Select next LSP subset (containing n LSPs). Go to step 1.

The essence of the above method is step 2, where the routing problem of a given set of LSPs is formulated as an ILP and tried to be solved. As an important characteristic of this approach, the possible paths are searched simultaneously for the LSPs. Moreover, the main advantage of ILP-SR is that it surely finds a feasible solution if one exists (for a given LSP subset). Another important action is filtering performed in step 1, which contributes to the reduction of running time of algorithm. As we have discussed in Section 2, if the above method could not find solution for the current value of n, it can be increased up to a predefined limit (max_n) and the method can be restarted. The detailed ILP formulation, the subset selection and filtering proposals are presented in the next two sections.

2.1.1. ILP formulation

As it was presented in the previous section, the base of ILP-SR is such a function that intends to solve the routing problem of a set of LSPs with the help of ILP formulation. The input parameters of this function are the new LSP demand and one or more previously established LSPs that have to be re-routed. Although we did not explicitly limit the number of LSPs to be re-routed, to understand the methodology of ILP based routing we present the detailed formulation of three independent LSP routes in this section, which corresponds to the case when we re-route two previously established LSPs in order to fit in the new LSP demand.

The network is modeled by a directed graph $\vec{G} = (V, E)$, where the set V of vertices and the set E of edges represent the routers and physical connections, respectively. Since we route bandwidth guaranteed traffic flows (LSPs), a capacity function $capacity(e)$, $e \in E$ specifies the total amount of reservable bandwidth per physical link. In our case three pairs of nodes $s_1, t_1 \in V$, $s_2, t_2 \in V$ and $s_3, t_3 \in V$ are given, representing the ingress and egress nodes of the new LSP demand and the two LSPs to be re-routed. Since the LSP demands have generally different bandwidth requirements, we have to classify the edges into subsets. Subsets E_1, E_2 and E_3 of E consist of the edges that have enough reservable capacity for the first, second and third LSP, respectively. Since an edge can be in more of these subsets at the same time, it is necessary to examine whether this edge has enough unreserved capacity to accommodate more LSPs at the same time. Edges in E_{12} ($\subseteq E_1 \cap E_2$) have the proper amount of reservable bandwidth to fit in the first and second LSPs simultaneously. Similarly, subsets E_{13} and E_{23} are given to represent the edges having enough capacity for the given LSP-pairs. Edges that are able to accommodate all three LSPs at the same time are the elements of subset E_{123} ($\subseteq E_1 \cap E_2 \cap E_3$). Introducing these notations, our task can be formulated in the following way. We search for three paths P_1, P_2 and P_3 from s_1 to t_1, from s_2 to t_2 and from s_3 to t_3, respectively, taking into consideration the following constraints:

- $P_1 \subseteq E_1$, $P_2 \subseteq E_2$ and $P_3 \subseteq E_3$,
- $P_1 \cap P_2 \subseteq E_{12}$, $P_1 \cap P_3 \subseteq E_{13}$ and $P_2 \cap P_3 \subseteq E_{23}$,
- $P_1 \cap P_2 \cap P_3 \subseteq E_{123}$.

These constraint conditions come directly from the above discussed system of subsets: an LSP can use such edges that have enough unreserved capacity for its bandwidth requirement, two LSPs can jointly use links width sufficient bandwidth for both of them, and all three LSPs can cross a given edge only if it has the proper amount of bandwidth to accommodate all of them. We can compose the following integer linear program in order to solve the above detailed routing problem. Vector variables $\underline{x}_1$, $\underline{x}_2$ and $\underline{x}_3$ show which edges are in use by the first, second and third LSP, respectively. For instance, x_1^e indicates whether the first LSP uses the particular edge e, or not. In this way we search for the *integer* numbers $x_1^e \in \{0,1\} : e \in E_1$, $x_2^e \in \{0,1\} : e \in E_2$ and $x_3^e \in \{0,1\} : e \in E_3$ taking into account the following constraints:

$$\sum_{e \in \rho(v) \cap E_1} x_1^e - \sum_{e \in \delta(v) \cap E_1} x_1^e = \delta_{v,t_1} - \delta_{v,s_1} \quad \forall v \in V, \tag{1a}$$

$$\sum_{e \in \rho(v) \cap E_2} x_2^e - \sum_{e \in \delta(v) \cap E_2} x_2^e = \delta_{v,t_2} - \delta_{v,s_2} \quad \forall v \in V, \tag{1b}$$

$$\sum_{e\in\rho(v)\cap E_3} x_3^e - \sum_{e\in\delta(v)\cap E_3} x_3^e = \delta_{v,t_3} - \delta_{v,s_3} \quad \forall v \in V, \tag{1c}$$

$$x_1^e + x_2^e \leqslant 1 \quad \forall e \in (E_1 \cap E_2) \setminus E_{12}, \tag{1d}$$

$$x_1^e + x_3^e \leqslant 1 \quad \forall e \in (E_1 \cap E_3) \setminus E_{13}, \tag{1e}$$

$$x_2^e + x_3^e \leqslant 1 \quad \forall e \in (E_2 \cap E_3) \setminus E_{23}, \tag{1f}$$

$$x_1^e + x_2^e + x_3^e \leqslant 2 \quad \forall e \in (E_1 \cap E_2 \cap E_3) \setminus E_{123}, \tag{1g}$$

where $\rho(v)$ and $\delta(v)$ are the sets of incoming and outgoing edges of node v, and the function $\delta_{i,j}$ is the Kronecker symbol, which is 1 or 0 according to whether i is equal to j or not.

We can extend this problem to an optimization problem by assigning cost functions c_1, c_2 and $c_3 : E \Rightarrow \mathbb{R}^+$ to the edges. Supposing that we use additive metric, the cost of path P_i is the sum of the corresponding link costs along path P_i. We search for the paths that satisfy the above detailed constraints and minimize the sum cost of the three paths. Using the above ILP notations, our task is to find the integer vectors $\underline{x}_1$, $\underline{x}_2$ and $\underline{x}_3$ that satisfy (1a)–(1g) and approach

$$\min \sum_{e\in E_1} c_1(e)x_1^e + \sum_{e\in E_2} c_2(e)x_2^e + \sum_{e\in E_3} c_3(e)x_3^e. \tag{2a}$$

The formulation of two paths is a bit simpler (see [12]), while routing four paths simultaneously—besides one additional equation per nodes corresponding to (1a)–(1c)—requires essentially more cases to be differentiated in order to examine the possible joint use of a given edge.

Generally, any ILP software package can be applied to solve this program. These solver packages are typically based on the so-called *branch and bound* technique (see, e.g., [21]) in order to find the optimal solution of the above detailed integer linear program. During simulations we used the `lp_solve` software package [4] to solve the ILP.

2.1.2. Select and filter LSP subset

In step 2 of ILP-SR an integer linear program is composed and tried to be solved. This action is the most time-consuming step of ILP-SR, thus decreasing the number of its executions could result in significant improvement in overall running time. For this purpose so-called *filters* can be applied. The feature of these filters is that their running time is negligible compared to the ILP formulation and solution, while using them many LSP subsets can be ignored during optimization, which results in considerable running time reduction.

Our first proposal called *routable filter* (RoF) is a *non-heuristic filter*, which means that it examines necessary conditions and consequently precludes only such situations that are surely impossible to be solved. It checks whether the new LSP demand could be routed if we tear down a given LSP subset. This can be simply performed with the help of the (constrained) Dijkstra's algorithm. It avoids the further investigation of such LSP subsets that do not facilitate the placement of the new LSP demand into the network.

Unfortunately, the path subset selection method of the ILP based algorithm cannot ensure any relationship between the LSPs to be re-routed. It can happen that the elements of the selected LSP subset (to be re-routed) are really far from each other in the network. In this case we may say that their routing is practically independent. For this kind of events the ILP formulation and solution is obviously useless. This is the idea behind the second filter called *relationship filter* (ReF), which tries to make the relationship between LSPs measurable. Based on this measure, the examination of such subsets where one or more LSPs are 'not in relationship' with the others can be avoided. ReF is *heuristic filter* as it checks such condition that probably get fulfilled if the examined case has a suitable solution, which means that it sometimes precludes also such situation that would be solvable. Although, this may cause the reduction of success probability of optimization, the approach can provide significant decrease in running time.

We decided to specify this relationship measure with the help of the absolute shortest path lengths determined purely by the network topology. As a starting step of this filter, we store in a table the minimal path length values between each node-pair. In order to specify the relationship between, e.g., LSPs A and B we examine the minimal length of such paths between the ingress and egress node of LSP A that use at least one edge of the current

path of LSP B. For a particular edge of LSP B, we have to sum the possible shortest path lengths between the end-points of the given edge and the ingress–egress pair of LSP A (dashed line in Fig. 1) with the help of the above specified minimal path length table. This sum increased by 1 (the length of the particular edge) indicates the minimal length of such a path that fulfills the above conditions. Repeating this examination for all edges of LSP B the overall minimum length can be determined. This minimal length is compared with the absolute shortest path length for LSP A, and the difference is considered as the measure of $B \to A$ where the arrow represents the relationship between the two LSPs. Obviously, this relationship is not symmetric, namely $(B \to A) \not\Rightarrow (A \to B)$.

This filter can be made parametric in the following way: restrict the relationship $B \to A$ only to such cases when the measure of relationship between two LSPs is at most p. This means that in case of $p = 0$ only an absolute shortest path crossing an edge of the other LSP can indicate relationship. On the other hand, when setting parameter p to, e.g., 3, relatively far LSPs can become 'relatives', since in this case B and A will be in relation, even if the minimal path for LSP A that crosses the current path of LSP B has three more hops compared to the absolute shortest path of LSP A.

For instance, we can see in Fig. 1 that LSP A—whose absolute shortest path is three hop long—has a three hop long path crossing one edge of LSP B, thus the measure of their relationship is 0. This means that the teardown (and re-routing) of LSP B could help the re-routing of LSP A in some cases.

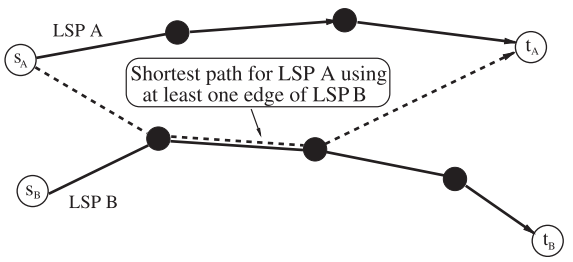


Fig. 1. Relationship filter.

Note that examining whether $B \to A$ does not need any information on the current path of LSP A, thus this relationship can be extended to the new LSP demand, too. In this way, when analyzing a particular LSP subset, the first step is to find such an LSP X that can help the establishment of the new LSP demand N, namely for which $X \to N$ holds true.

In summary, this filter reduces the problem instances that need ILP formulation (and solution) only to such cases when a directed relationship system can be determined for the LSP subset. For example, $B \to A \to N$ indicates three LSPs (including the new demand as well) that could be (re-)routed simultaneously: the system of their relationships shows that the teardown of LSP A can help the establishment of the new LSP demand, while the re-routing of LSP B can facilitate the replacement of LSP A.

An important feature of this filter is that its implementation is very simple, it can be realized with the help of the pre-computed shortest path length table and simple lookup operations.

2.2. Dijkstra's algorithm based recursive re-routing

In the previous section we presented an ILP based method aiming at finding paths for more LSPs as a possible realization of the core function of PPPO algorithm. In this section we detail our other proposal to solve this task, namely the *Dijkstra's algorithm based recursive re-routing* (DA-RR) algorithm. The main difference between the proposals is that while ILP-SR routes the LSPs simultaneously, DA-RR selects path only for one LSP at a time. The principal idea behind this method is the following: find an LSP whose teardown releases the sufficient amount of bandwidth for setting up the new LSP demand, route the new LSP demand, then consider the torn down LSP as a new LSP demand, and try to route this 'new LSP demand' in the same way.

This idea can be implemented with a help of a recursive procedure. The pseudo code of this procedure called 'darr' can be seen in Fig. 2. As we have mentioned, the need for PPPO is indicated by the failure of CSPF. Moreover, resulting from the basic structure of algorithm, on a particular

```
procedure darr(LSP_demand, re-routing_level) {
  if re-routing_level = n {
    if dijkstra(LSP_demand) return success
    else return failure
  }
  else foreach old_LSP {
    new_LSP_demand=teardown(old_LSP)
    if dijkstra(LSP_demand) {
      if darr(new_LSP_demand, re-routing_level + 1) return success
      else teardown(LSP_demand)
    }
    else restore(old_LSP)
  }
  return failure
}
```

Fig. 2. Dijkstra's algorithm based recursive re-routing.

optimization level n, it is sure that the new LSP demand cannot be set up by re-routing at most $n-1$ LSPs (see steps of PPPO). Consequently, when the level of recursion reaches the current optimization level n, the procedure tries to route the actual LSP demand without allowing the re-routing of any further LSPs. If we are not yet at level n, the procedure tears down the previously established (referred as 'old') LSPs in a cycle, and checks if the actual LSP demand became routable. If we succeed in routing the actual LSP demand, the re-routed LSP is considered as a new LSP demand in the following, and the procedure is called recursively by also increasing the number of re-routed LSPs. If the procedure does not find an 'old' LSP that both enables the set up of the actual LSP demand and can be re-routed itself, it returns to the lower level (in recursion) with failure. The 'dijkstra' function in the procedure performs a bandwidth constrained path selection by pruning the links with unsufficient amount of bandwidth. It returns with 'success' if the routing was successful, and with 'failure' otherwise. The 'teardown' function releases the bandwidth reserved by the LSP and returns the LSP as a new LSP demand, while the 'restore' function is in charge of restoring the original path and reservations of the given LSP.

2.2.1. Filtering

As we have discussed earlier, the basic conception of ILP-SR does not ensure any relationship between the LSPs to be re-routed, e.g., it can happen that they are 'far' from each other which means that they practically cannot affect the routing of each other. On the contrary, DA-RR implicitly guarantees that the LSPs to be re-routed are in close relation in all examined cases. The necessary condition of the origination of a new branch in recursion is that the teardown of the new LSP to be re-routed has to enable the setting up of the current LSP demand. In this way, every examined situation includes the possibility of a successful optimization. Therefore, no further filters are used for DA-RR.

3. Numerical results

In order to investigate the attainable performance improvement resulting from the application of PPPO, we performed various simulation experiments. Analyzing the numerical results we focused on two principal issues: (1) long-term effects of PPPO on the throughput of network, and (2) limits of real-time applicability of the proposed

methods considering the network size. In the remainder of the section the applied simulation environment is presented, then the performance evaluation of the proposed algorithms is given, including the description of the performed simulation scenarios.

3.1. Simulation environment

The analytical investigation of different routing algorithms (see, e.g., [20]) is generally a very complex and difficult task. Considering the examined optimization problem it could be hard to find a tractable analytical approach that can be applied to obtain practical results. Thus, we decided to apply a discrete event driven simulator, which enables the investigation of the dynamic operation of the network. Since MPLS is generally used in backbone networks, we considered macro-flow-level simulation as favorable approach.

The topology of network has a large influence on the behavior of routing algorithms, so it was a key challenge to characterize the networks that are relevant for our investigations. We decided to generate topologies with the *Random Topology Generator* method [9,10]. This method takes into consideration the characteristics of real communication networks, and enables to adjust such topology parameters as number of nodes and nodal degree. In order to get results from different network configurations, we varied the number of nodes from 10 to 50, and the average nodal degree from 2.5 to 4. The links were generally 2.5 Mbps (OC-48) connections. During simulations we did not focus on network element failures, consequently the topologies were permanent in a particular scenario.

For the lack of real traffic statistics, we generated the traffic demands randomly. Since typically every node of a backbone has its own access area, we considered each node as a potential ingress and egress router for LSPs. Thus, we generated traffic demands for all node-pairs based on a uniform distribution, and the number of LSPs per node-pair was generally set to three on average.

In order to analyze the dynamical operation of the network, we examined the behavior and effect of the algorithms in the time domain. We decided to model the arrival of new LSP demands as a Poisson process with rate λ, which is the general approach in case of traffic flows with bandwidth requirements. Since we model aggregated micro-flows in our environment, some long-tailed distribution seemed suitable to model the connection times. Thus, we decided to apply the Weibull distribution with shape parameter $a < 1$ for this purpose. This method is derived from the exponential distribution, the difference is that the emphasis is shifted in the direction of larger holding times, in other words the probability of longer connections is higher.

Considering the bandwidth requirements of LSPs, we applied the common rule that defines the maximum possible bandwidth of a traffic demand as the 10% of the average link capacity. Besides, during a particular simulation scenario we adjusted the total volume of bandwidth requested in the network by adjusting the upper limit of LSP bandwidth. Within this interval, the distribution of the bandwidth values was uniform. In this way, the total network load and the blocking probability values could be controlled in order to set such network states that were interesting for us.

The general arrangement of our simulations was the following. Each scenario started with an "empty" network without established LSPs. During the simulation two phases were differentiated: (1) the initial so called *transient* phase, when the average number of established LSPs and the network load increases rapidly; and (2) the second so called *stationary* phase, when the network throughput is more or less smooth. The length of the transient phase was determined in such way that the necessary time to reach the expected value of the number of established LSPs—that is the product of arrival rate and holding time parameters in non-blocking case—was doubled. Thus, the number of arriving LSP requests was the threefold of the expected number of LSPs in general. On the other hand, during the stationary phase the number of new LSP arrivals was generally the twentyfold of the average number of established LSPs. Obviously, in order to ignore the initial transient data, the measurements were done purely in the stationary phase. To get reliable results this process was repeated until the statistical error of

results decreased below 2% in case of confidence level of 95%. On the other hand, in case of practical running time values of algorithms the requirements on confidence were looser, since in that case we only focused on order of magnitude differences.

In our opinion operators generally endeavor to manage as much traffic as possible, therefore we identified the *total network throughput* as the main performance metric in numerical comparisons. The current network throughput value consists of the sum of reserved bandwidths by all established LSPs at a given moment. We consider this value on average through the stationary phase. Another important measure is the *blocking probability*, which shows the ratio of non-routable traffic demands and all traffic demands. Since the proposed PPPO method is triggered by the failure of CSPF, the operation of optimization algorithms can be examined only when a part of the arriving LSP demands get blocked. On the other hand, too high blocking value is not realistic considering that if the majority of the demands fails to be set up then probably the network has to be re-dimensioned. Furthermore, we examined also special cases of traffic pattern distortion—since one can benefit from the re-routing feature of our optimization method in such situations—when the experienced blocking probability can be higher than in case of normal operation.

3.2. Performance evaluation

In this section we describe the performed simulation scenarios, furthermore, present and analyze our simulation results. In the first scenario, the practical running time values of algorithms are examined, laying emphasis on the effect of various filters. In the second example we investigate the behavior of our optimization algorithms in such situation where though the traffic is well balanced, the network is lightly overloaded. After that we try to explore what happens when the traffic of a particular network node suddenly increases for some reason. Finally, we simulate the situation when the focus of traffic volume moves from one part of a network to another. As we mentioned, both ILP-SR and DA-RR (as well as CSPF) enable the use of any additive type link metric. Since we focus on highly loaded network situations during investigation of the effect of PPPO, we used the discrete link cost method (see Section 1) with parameters $C = 100$, $\alpha = 1.0$ and minimum link utilization level of 0 in our experiments [23].

3.2.1. Computational efficiency and filters

As we have discussed, an important task of this study is to investigate the possible upper limit of the number of LSPs to be re-routed. Thus, we examined the running time of the algorithms at various network sizes and traffic situations. Since PPPO is intended to be used in real-time applications, we considered 10 min as the upper limit of acceptable running time. As we have discussed, the running time can highly depend on the efficiency of filtering, thus the further task of this section is to examine the obtained performance gain when applying filters.

In the first simulation scenario we investigated how filtering reduces the number of LSP subsets to be examined, and consequently the running time. Since the relationship filter (ReF) is heuristic, also the possible degradation in long-term performance had to be examined. The experiments were focused on the case when maximum two LSPs are allowed to be re-routed by ILP-SR (referred as 'ILP-SR 2'). The results of measurements performed on 10-node networks with average nodal degree of 4 are summarized in Table 1. We differentiated two *worst cases*: (1) the new LSP demand can be established only by re-routing exactly the maximum number of re-routable LSPs, or (2) the new LSP demand cannot be set up at all. The rational behind this approach is that the practical running time could be critical in these two cases. In accordance with this, two values can be found per cell in the table corresponding to these two cases, respectively. The simulations were carried out on a Sun Ultra Enterprise 420R machine with Ultra II 450 Mhz processor and 1 GByte memory. The tendencies are very clear. Applying purely the routable filter (RoF) the whole state space could be reduced to 15.4% of the original size, which means that the ILP formulation can be avoided in about 85% of all cases (note that RoF is always in use). Using ReF with $p = 1$ (recall that parameter p

Table 1
Effect of relationship filter (ReF)

Algorithm/ measure	Examined cases [%]	Running time [s]	Throughput [%]
ILP-SR 2	0.9	8.2	108.14
ReF $p = 0$	1.5	13.8	
ILP-SR 2	3.9	35.6	108.21
ReF $p = 1$	7.6	69.0	
ILP-SR 2	5.9	49.1	108.27
w/o ReF	15.4	148.9	
CSPF	n.a.	<0.1	100.0
w/o opt.			

Table 2
Running time of algorithms [s]

Algorithm/ network size	10 nodes	15 nodes	20 nodes	30 nodes	50 nodes
DA-RR 1	<0.1	<0.1	<0.1	0.6	5.5
	<0.1	<0.1	<0.1	0.6	5.3
ILP-SR 1	<0.1	<0.1	1.8	12.4	52.2
	<0.1	<0.1	2.1	10.3	62.3
DA-RR 2	1.1	8.3	34.8	554.5	n.a.
	0.7	6.7	45.7	556.0	n.a.
ILP-SR 2	8.2	42.2	389.4	n.a.	n.a.
	13.8	87.1	588.2	n.a.	n.a.
DA-RR 3	10.3	58.4	n.a.	n.a.	n.a.
	7.8	44.2	n.a.	n.a.	n.a.
ILP-SR 3	523.4	n.a.	n.a.	n.a.	n.a.
	2751.3	n.a.	n.a.	n.a.	n.a.

specifies the measure of relationship, see Section 2.1.2) the number of examined cases is below 8%. Moreover, setting p to 0, only 1.5% of the original cases has to be checked. These proportions can be observed in the measured running times, too. Although running time without any filter is not available since the RoF—as it checks necessary condition—is always applied, the effect of relationship filter is well noticeable, namely the 1–2 min running time can be reduced to about 10 s. Beside the running time we should also investigate the long-term throughput. In the performed experiments we did not realize relevant performance degradation when applying ReF, as we expected. Consequently—because of the significant reduction in running time—we only consider the fastest version, namely the 'ReF $p = 0$' filtering in the following examinations.

In the next scenario we analyzed the practical running time values in case of different PPPO algorithms and various network sizes. As we can see in Table 2, there is an order of magnitude difference between the running times of DA-RR and ILP-SR methods at each level. We can also notice that allowing to re-route a single LSP with DA-RR (referred as 'DA-RR 1') needs minimal running time even in case of a 50-node network, where the ILP based method finishes in a minute on average. Considering ten minutes as a realistic time limit, we may say that if two LSPs are allowed to be re-routed, DA-RR can be applied up to 30 nodes. 'ILP-SR 2' applying the relationship filter ('Ref 0') can be used up to network sizes of 20 nodes. The necessary running time of 'DA-RR 3' in case of blocking is unacceptably high in case of a 20-node network; the algorithm reaches the limit of its applicability at about 17–18 nodes. Unfortunately, re-routing three LSPs with ILP-SR may result in almost one hour blocking time even if the relationship filter is applied (for 10-node networks). Thus, it proves to be unfeasible for real time situations. However, it might be used in extremely small networks, or if the number of LSPs is low.

3.2.2. Normal operation with symmetric traffic pattern

In this simulation scenario we considered a network with roughly symmetric traffic, namely the intensities of new LSP requests were nearly the same between the various nodes. As it had been introduced previously, the average network load was adjusted by changing the expected value of LSP bandwidth requests. The simulation was performed on a 20-node random network topology (see Fig. 3b) having average nodal degree of 4. As we can see in Fig. 3a, the difference between the algorithms is minimal in case of average LSP bandwidth of 68 Mbps. This is not surprising since the blocking probability of a new demand is very low at this point. It is clearly visible that increasing the load the benefit from the use of PPPO

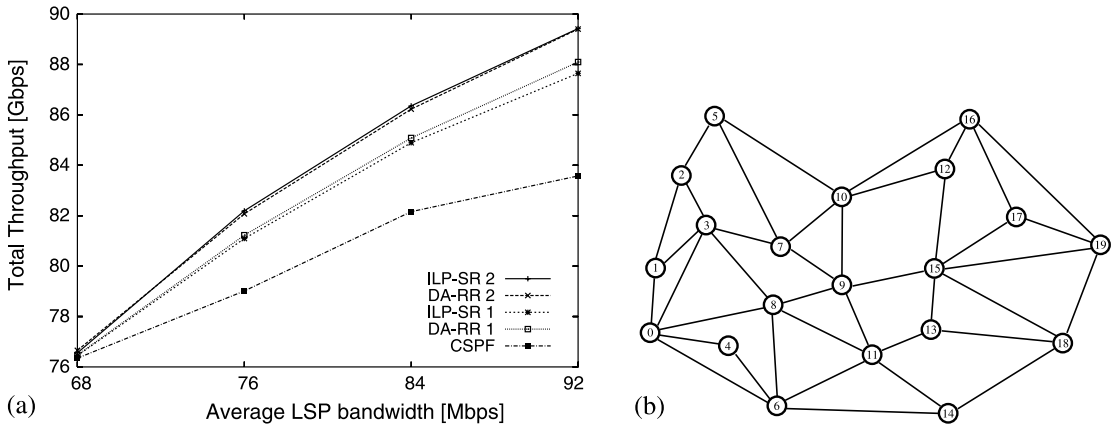


Fig. 3. Evenly distributed LSP requests. (a) Throughput change in traffic domain; (b) network topology.

increases, too. At about 84 Mbps average LSP bandwidth, by re-routing a single LSP the performance improvement is about 4%. Moreover, allowing the re-routing of two LSPs, the attainable gain in total network throughput reaches 6%. By applying ILP-SR as the core function of PPPO, even higher network utilization can be achieved in some cases, however, the average improvement is similar. The reason for the different results of approaches could be that ILP-SR sometimes succeed in re-routing such LSP subset that DA-RR cannot re-route due to its heuristic nature. On the other hand, it can also happen that ILP-SR filtering (ReF) ignores such LSP subset that DA-RR could re-route. However, if the running time becomes critical—because of the network size, or the unexpectedly large number of LSPs, etc.—we propose to use DA-SR, as it can provide quite good result with moderate running time.

One can realize that the difference between CSPF and DA-RR 1 methods is much larger then the difference between DA-RR 2 and DA-RR 1. This tendency is valid for ILP-SR, as well. Therefore at 84 Mbps average LSP bandwidth value we investigated the distribution of LSP re-routing cases (see Table 3). After CSPF failure, re-routing a single LSP enables the routing of the new LSP demand in about 63% of the cases, while re-routing two LSPs increases the routing probability by only 23–24%, which agrees with our above observation.

Table 3
Ratio of optimization and blocking cases [%]

Algorithm/ opt. case	Re-routing 1 LSP	Re-routing 2 LSPs	Blocking
ILP-SR 2	63.32	23.71	12.97
DA-RR 2	62.76	23.03	14.21

3.2.3. Increased traffic volume in a particular network node

In order to get a deeper insight into the behavior of various optimization algorithms, it is worth testing them on special situations. In our point of view, a special situation means that the traffic pattern differs significantly from the original one that the network was planned for. This might cause that some parts of the network become overloaded, while other parts remain underutilized. The reason for this change in traffic demands can be various. In this section we investigate the case when the incoming and outgoing traffic volume of a particular network node increases unexpectedly due to some special event in the access area of the node, e.g., an earthquake.

We decided to choose a 13-node random network (shown in Fig. 4b) with 3.54 links per node on average. After the initial transient phase the network operates normally for 40 time units. In this phase the traffic is balanced, the blocking probability of a new LSP demand is very low, about 1–2%. After this stationary state—in order

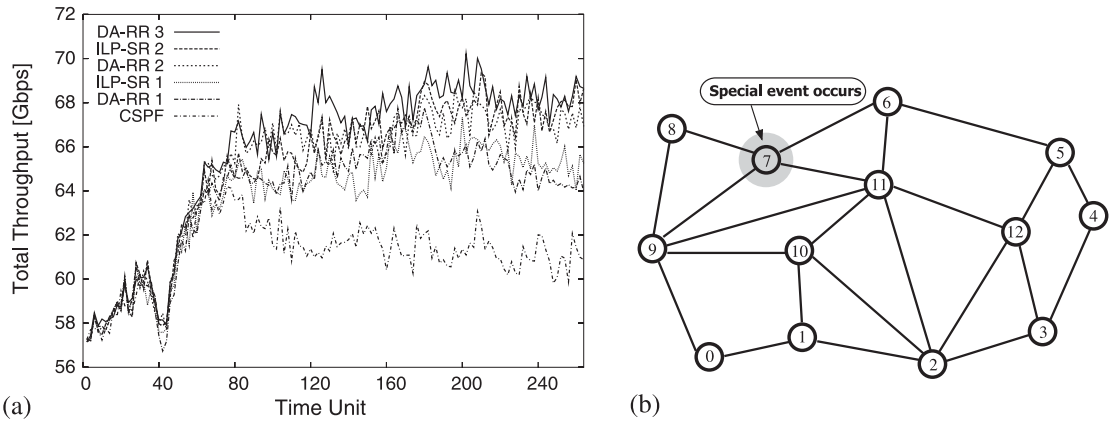


Fig. 4. Effect of special event occurred in the 40th time unit. (a) Throughput change in time domain; (b) network topology.

to simulate a special event—we increased the LSP arrival intensities by 200% from and towards node 7, while the intensities between other LERs remained the original. We can observe in Fig. 4a that this change results in a significant growth in the overall throughput. However, the blocking probability increases, too. Applying PPPO a part of the demands blocked by CSPF can be routed. Consequently, much higher network throughput can be achieved. In Table 4 the average throughput values are shown concerning the [80,260] time interval. When the re-routing of a single LSP is allowed, about 6% improvement can be reached. By re-routing an additional LSP the performance gain increases further by 3%, resulting in an average 109% throughput compared to the pure CSPF method. Allowing to re-route 3 LSPs in DA-RR the throughput increases again by 1.5%.

In summary, we can say that by allowing to re-route several LSPs in order to fit in a new LSP demand, a significantly higher network utilization can be reached. However, increasing the number of LSPs that are allowed to be re-routed, the degree of performance improvement decreases, in accordance with our previous observations (see Section 3.2.2). Since ILP-SR and DA-RR performed very similarly, and considering that in this kind of situation the time limits of the algorithm would be significantly lower because of the more frequent change of traffic demands, the use of DA-RR seems to be more favorable.

3.2.4. Shift in traffic focus

In this section we examine another situation when the traffic pattern differs from the original one, namely we move the focus of traffic volume from one part of a network to another. With the help of this simulation the effect of a badly dimensioned network could be analyzed, furthermore, this approach could enable the modeling of the daily profile changes in a continent-wide network. In order to simulate this event we examined a network that consists of two regions. The nodes from 0 to 5 correspond to first Region A, while the remaining nodes correspond to Region B (see Fig. 5b). The majority of traffic demands is generated inside the regions, namely the amount of such traffic that connects the two regions is only 10% of

Table 4
Average network throughput in the interval [80,260]

Measure/algorithm	CSPF	DA-RR 1	ILP-SR 1	DA-RR 2	ILP-SR 2	DA-RR 3
Throughput [Gbps]	61.52	65.26	65.12	66.83	66.91	67.83
Rel. throughput [%]	100.0	106.07	105.85	108.64	108.75	110.25

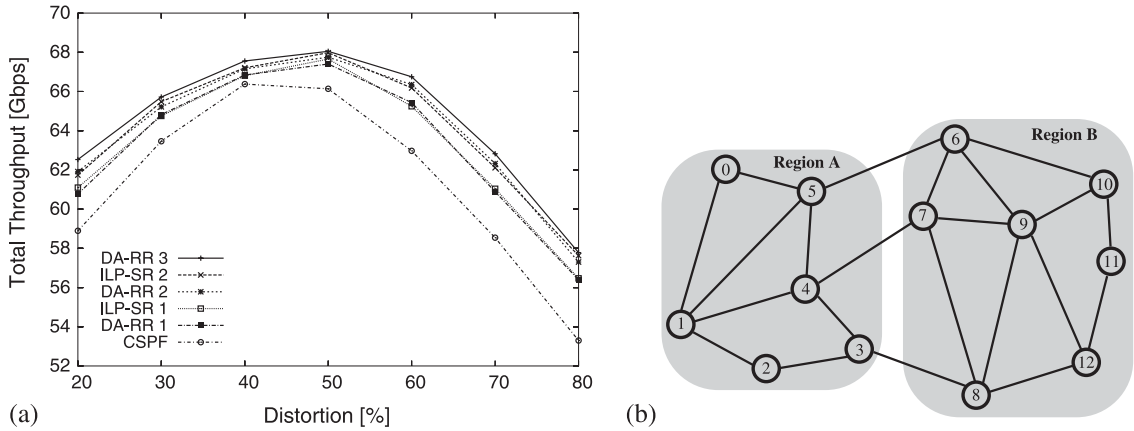


Fig. 5. Moving the focus of traffic between two network regions. (a) Throughput change in topology domain; (b) network topology.

the total traffic of the whole network. During the simulation we distort the traffic pattern in such way that in Region A we multiply the traffic intensities by $\delta/0.5$ and in Region B by $(1-\delta)/0.5$, where δ changes from 20% to 80%. Thus, in case of $\delta = 50\%$ the traffic matrix is the original, while towards the borders the traffic is concentrated almost only into either of the two regions.

As we can see in Fig. 5a, the network is the most balanced when $\delta = 40$. The reason for this is that Region B consists of one more node and three more links compared to Region A, consequently it can accommodate basically larger volume of traffic without significant blocking. Moving the focus of traffic to Region A or Region B the performance of pure CSPF method is reduced. By applying PPPO to complete CSPF the flexibility of routing can be increased, in consequence significantly higher average throughput level can be achieved on the long term. The available performance improvement is about 5–10% in the relevant interval.

4. Conclusion

In MPLS networks CSPF algorithms provide the simplest, automated method for setting up bandwidth constrained label switched paths. Another widely discussed path computation method is global path optimization that is generally performed in a central traffic engineering tool. While CSPF concentrates on a single LSP at a time, in case of global optimization all LSPs are considered for optimization. In this work we discussed and evaluated a third option, namely the problem of partially optimizing few already established label switched paths, in order to fit in a new LSP demand that could not be established by the simple CSPF algorithm. In our new method, beside requiring that the number of affected LSPs should be limited, we also aimed at limiting the running time of optimization. We argued that the proposed PPPO method can be best utilized in a dynamic environment, where the set of established paths continuously changes in the network. We identified the total throughput as the key network performance indicator. Thus, PPPO aims at improving the total network throughput, by giving another chance for LSPs that faced CSPF failure.

We presented two basically different algorithms. The first one uses heuristic filters to select such LSP subsets that are probably re-routable. Once a candidate LSP subset is identified, an integer linear programming based solution decides if the new LSP demand and the selected old ones can be established simultaneously. The filtering phase is critical in reducing the running time of this algorithm, since the solution of the ILP formulation is very time consuming. The second algorithm is based on the Dijkstra's shortest path algorithm, which can basically provide relatively faster operation. This second proposal aims at discovering

the state space by a recursively called procedure. It also limits the number of investigated cases by requiring that only such LSPs can be members of the re-routed LSP subset that enable the setup of the new LSP or the re-routing of another old LSP.

The proposed two methods are general in the sense that both are theoretically applicable for re-routing any number of LSPs. However, our simulation experiments showed that the complexity of the problem allows the investigation of re-routing 1, 2 or 3 LSPs. Applying our PPPO method for slightly overloaded networks, the performance gain, namely the total network throughput increase was 4–10% compared to the traditional CSPF method. Furthermore, we realized that our recursive algorithm provided quite similar results as the ILP based method, but with significantly lower running time.

As a future work, we would like to carry out experiments on real ISP topologies and to apply more realistic traffic models including LSP arrival processes, bandwidth distributions, number of traffic classes, etc. The recent topology and flow characterization works of CAIDA [5] could provide input for this activity. Another work item could be the development and analysis of novel filtering heuristics to improve the running time and performance of the optimization algorithms.

Acknowledgements

The authors wish to thank Áron Szentesi and Alpár Jüttner for the valuable discussions.

References

[1] G. Apostolopoulos, R. Williams, S. Kamat, R. Guerin, A. Orda, A. Przygienda, QoS routing mechanisms and OSPF extensions, Internet Engineering Task Force, Request For Comments 2676, August 1999.

[2] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, O. Waarts, On-line machine scheduling with applications to load balancing and virtual circuit routing, in: Proceedings of 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 623–631.

[3] P. Aukia, M. Kodialam, P.V. Koppol, T.V. Lakshman, H. Sarin, B. Suter, RATES: a server for MPLS traffic engineering, IEEE Network 14 (2) (2000) 34–41.

[4] M. Berkelaar, J. Dirks, `lp_solve 2.2`, `ftp://ftp.es.ele.tue.nl/pub/lp_solve`.

[5] CAIDA: Cooperative Association for Internet Data Analysis, The CAIDA Web Site, `http://www.caida.org`, 2003.

[6] Ca*net 4 Plans Customer Control of Lambdas, The Cook Report on Internet, `http://www.cookreport.com/10.10.shtml`, January 2002 (10.10).

[7] J. Forslöw, I. Jarrett, P. Moran, B. Szviatovszki, Management solutions for IP networks, Ericsson Review 77 (1) (2000) 42–52.

[8] B. Fortz, M. Thorup, Internet traffic engineering by optimizing OSPF weights, IEEE INFOCOM, March 2000.

[9] B.G. Józsa, D. Orincsay, Random graph generator (for telecommunication networks), Technical Report, Ericsson Research, TrafficLab, Budapest, Hungary, 1999.

[10] B.G. Józsa, Z. Király, G. Magyar, Á. Szentesi, An efficient algorithm for global path optimization in MPLS networks, Optimization and Engineering 2 (3) (2001) 321–347.

[11] B.G. Józsa, M. Makai, On the solution of reroute sequence planning problem in MPLS networks, Computer Networks 42 (2) (2003) 199–210.

[12] A. Jüttner, B. Szviatovszki, Á. Szentesi, D. Orincsay, J. Harmatos, On-demand optimization of label switched paths in MPLS networks, ICCCN 2000, Las Vegas, 2000.

[13] C. Liljenstolpe, Offline traffic engineering in a large ISP setting, Internet Engineering Task Force, Internet Draft, November 2001, Work in Progress.

[14] Q. Ma, P. Steenkiste, H. Zhang, Routing high-bandwidth traffic in max–min fair share networks, ACM SIGCOMM, August 1996, pp. 206–217.

[15] M. Kodialam, T.V. Lakshman, Minimum interference routing with applications to MPLS traffic engineering, in: Proceedings of IEEE INFOCOM 2000, Tel Aviv, Israel, March 2000.

[16] K. Kompella, D. Awduche, Notes on path computation in constraint-based routing, Internet Engineering Task Force, Internet Draft, September 2000, Work in Progress.

[17] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, C. Diot, IGP link weight assignment for transient link failures, Sprint ATL Technical Report TR02-ATL071000, July 2002.

[18] D. Orincsay, J. Harmatos, On-demand optimization of protected LSP-tunnels in MPLS networks, IFIP 2001, Budapest, June 2001.

[19] M. Pióro, P. Gajowniczek, Simulated allocation: a suboptimal solution to the multicommodity flow problems, in: 11th UK Teletraffic Symposium, 1994.

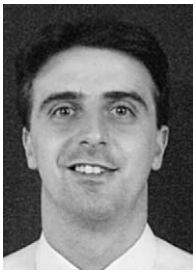
[20] S. Plotkin, Competitive routing of virtual circuits in ATM networks, IEEE Journal of Selected Areas in Communications 13 (6) (1995) 1128–1136.

[21] A. Schrijver, Theory of Linear and Integer Programming, Wiley-Interscience Series in Discrete Mathematics, Wiley, London, 1986.

[22] A. Shaikh, J. Rexford, K.G. Shin, Load-sensitive routing of long-lived IP flows, ACM SIGCOMM, 1999, pp. 215–226.

[23] A. Shaikh, J. Rexford, K.G. Shin, Evaluating the overheads of source-directed quality-of-service routing, IEEE/ACM Transactions on Networking 9 (2) (2001) 162–176.

[24] B. Szviatovszki, Á. Szentesi, A. Jüttner, Minimizing rerouting in MPLS networks with preemption-aware constraint-based routing, in: 2001 International Symposium of Performance Evaluation of Computer and Telecommunication Systems, 2001, pp. 249–261.
[25] D. Wang, Network planning and analysis tools for MPLS networks, Slide presentation given at the IW-MPLS'98 Conference, November 1998.
[26] Z. Wang, J. Crowcroft, Quality-of-service routing for supporting multimedia applications, IEEE Journal on Selected Areas in Communications 14 (7) (1996) 1234–1288.



Dániel Orincsay received his M.Sc. degree in 2000 at the Budapest University of Technology and Economics, where he is currently working on his Ph.D. Parallelly, he also works as a research fellow at Traffic Analysis and Network Performance Laboratory of Ericsson Research Hungary. His main interests are traffic engineering and performance optimization in high speed networks.

Balázs Szviatovszki received his M.Sc. degree in 1996 at the Budapest University of Technology and Economics, where he is currently finalizing his Ph.D. Meanwhile, he also works as a research fellow at Ericsson Research, Budapest, Hungary, at the Traffic Analysis and Network Performance Laboratory. His main interests are traffic engineering and performance optimization problems in IP networks.

Géza Böhm was graduated in computer science in 2002 at the Budapest University of Technology and Economics. Although he was specialized in telecommunication and traffic engineering, actually he is working for a CAD software company. His task is to find connection between the building industry and architectural applications.