University of Würzburg
Institute of Computer Science
Research Report Series

Optimization of the Self-Protecting Multipath for Deployment in Legacy Networks

Rüdiger Martin, Michael Menth, Ulrich Spörlein

Report No. 418 April 2007

University of Würzburg
Institute of Computer Science
Department of Distributed Systems
Am Hubland, 97074 Würzburg, Germany
{martin,menth,spoerlein}@informatik.uni-wuerzburg.de

Optimization of the Self-Protecting Multipath for Deployment in Legacy Networks

Rüdiger Martin, Michael Menth, Ulrich Spörlein
University of Würzburg
Institute of Computer Science
Department of Distributed Systems
Am Hubland, 97074 Würzburg, Germany
{martin,menth,spoerlein}@informatik.uni-wuerzburg.de

Abstract

The self-protecting multipath (SPM) is a simple protection switching mechanism that can be implemented, e.g., by MPLS. We present a linear program for the optimization of the SPM load balancing parameters to maximize the amount of transportable traffic with resilience requirements. This is needed to configure the SPM for the deployment in legacy networks. Our study shows that the SPM is very efficient in the sense that it can carry 50% - 200% more protected traffic than IP rerouting in sufficiently meshed networks. The investigation of the computation time and the memory consumption recommends the COIN LP (CLP) as preferred LP solver. The computation time of the program depends mainly on the number of links in the network and networks with up to 240 links can be optimized within one hour on a standard PC.

1 Introduction

Carrier grade networks require high availability which is often as high as 99.999% such that restoration or protection switching is required. Restoration sets up a new path after a failure while protection switching pre-establishes backup paths in advance. A typical restoration scheme is shortest path rerouting (SPR) in IP networks, which heals broken paths some time after a failure. A typical protection switching mechanism is the primary and backup path concept, where the traffic is switched onto the backup path as soon as the primary path does not work anymore. Protection switching or restoration mechanisms alone are not sufficient to maintain the full service availability during network failures. Then, the links carry the normal traffic together with the deviated traffic. As a consequence, the quality of service (QoS) can only be met if the links have enough capacity. This must be taken into account for network provisioning. If the link capacities are already given, the structure of the backup paths must be laid out in such a way that they have enough capacity for all relevant failure scenarios.

In this paper, we focus on the self-protecting multipath (SPM) which is a protection switching mechanism that has been proposed in previous work [1,2]. The SPM consists

This work was funded by Siemens AG, Munich, and by the Deutsche Forschungsgemeinschaft (DFG) under grant TR257/18-2. The authors alone are responsible for the content of the paper.

of several parallel paths between source and destination, and a load balancing function distributes the traffic over the working paths. The particularity of that concept is that the traffic may be spread over several paths both under normal networking conditions and in case of network failures. First, a multipath structure for the SPM is found and then, the load balancing function can be optimized. The contribution of this paper is a concise presentation of a linear program (LP) that optimizes the load balancing function of the SPM for network dimensioning in such a way that the amount of transportable traffic with resilience requirements is maximized. In addition, the complexity of the LP is investigated both theoretically and by empirical data. This is crucial for the assessment of the practical applicability of this optimization approach.

This paper is organized as follows. Section 2 gives an overview on protection switching techniques. Section 3 explains the LP for the optimization of the SPM load balancing functions and analyzes its complexity. Section 4 investigates the capacity gain for traffic with resilience requirements in networks using the SPM instead of simple IP rerouting; furthermore, computation time and memory consumption of the optimization program are studied by experimental data. Finally, the conclusion in Section 5 summarizes this work and gives an outlook on further research.

2 Overview on Resilience Mechanisms

In this section we give a short overview on various resilience mechanisms to contrast the SPM against other approaches.

2.1 Restoration Mechanisms

As mentioned before, restoration mechanisms take actions only after a network failure. They try to find new routes or set up explicit backup paths when the traffic cannot be forwarded anymore due to link or node failures. The disadvantage of such methods is obvious: they are slow. The re-convergence of the IP routing algorithm is a very simple and robust restoration mechanism [3, 4]. Another example are backup paths in MPLS that are set up after a network failure.

2.2 Protection Switching Mechanisms

The authors of [5] give a good overview on different protection switching mechanisms for MPLS.

2.2.1 End-to-End Protection with Primary and Backup Paths

Backup paths are set up simultaneously with primary paths and in case of a failure, the traffic is just shifted at the path ingress router of a broken primary path to the corresponding backup path. This is called end-to-end protection. It is faster than restoration methods but the signalling of the failure to the path ingress router takes time and traffic being already on the way is lost.

2.2.2 Fast Reroute Mechanisms

MPLS fast reroute (FRR) tackles the problem of lost traffic in case of end-to-end protection. Backup paths towards the destination are set up not only at the ingress router of the primary path but at almost every node of the path [6,7]. Then, a backup path is immediately available if the path breaks at some location. Currently, fast reroute mechanisms are also discussed for IP networks. Several solutions are being discussed but a preferred method is not yet established [8–11].

2.2.3 Self-Protecting Multipath

The self-protecting multipath (SPM) has been presented first in [1,2]. Its path layout consists of disjoint paths and the traffic is distributed over all of them according to a traffic distribution function (see Figure 1). If a single path fails, the traffic is redistributed over the working paths according to another traffic distribution function such that no traffic is lost. Thus, a specific traffic distribution function is required for every pattern of working paths.

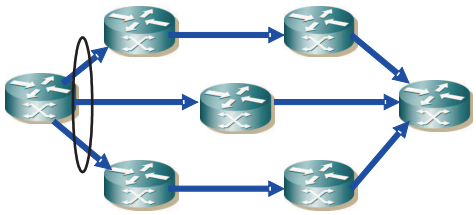


Figure 1: The SPM performs load balancing over disjoint paths according to a traffic distribution function which depends on the working paths.

2.3 Routing Optimization

The traffic matrix and the paths of the flows together determine the resource demands on the links. The layout of the paths may be optimized to minimize either the link utilization or the required network capacity. In the following, we address briefly different optimization objectives to distinguish our optimization problem from others.

2.3.1 Routing Optimization in Combination with Network Dimensioning

In not yet provisioned networks, the network capacity and the routing may be determined together. If failure scenarios are not taken into account, shortest path routing requires the least capacity. With resilience requirements, however, backup resources may be shared by different flows in different failure scenarios. Routing optimization can reduce the required network capacity considerably by maximizing the capacity sharing. This has been exemplified by [1] and [12].

2.3.2 Routing Optimization for Legacy Networks

In already provisioned networks or legacy networks, the capacity of the links is fixed. If the traffic matrix is given, the maximum link utilization in the network under failure-free conditions can be minimized by a suitable routing. This has been done for IP networks [13], for MPLS networks, and for hybrid networks [14]. If restoration or protection switching is applied, the target is the minimization of the maximum link utilization in any failure case. This has been done for IP networks [3, 4] and for MPLS networks [15]. Thereby, backup capacities may be shared by different flows and in different failure scenarios. The objective of this work is to optimize the SPM in such a way that the maximum link utilization in any protected failure scenario is minimized. This is equivalent to a maximization of the amount of transportable traffic with resilience requirements by scaling up the traffic matrix up to the point where traffic is lost in at least one failure scenario.

3 Optimization of the SPM for Deployment in Legacy Networks

The SPM consists of parallel paths over which the traffic is distributed according to a load balancing function. A suitable choice of the multipath layout and the optimization of the path failure specific load balancing function can minimize the maximum link utilization ρ_{max} in any protected failure scenario. First, we address the path layout, then we explain the linear program for the optimization of the load balancing functions, and finally, we analyze the complexity of the linear program.

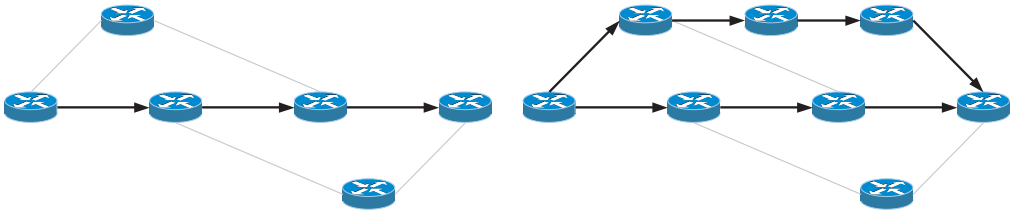
3.1 Path Layout

First we consider algorithms to find disjoint parallel paths and then we address the problem of SRLGs.

3.1.1 Algorithms for Disjoint Parallel Paths

The SPM consists of disjoint parallel paths such that the remaining paths are still working if a single path fails due to the failure of a single network element. Some network topologies do not allow to find disjoint paths, but we do not consider that case in this investigation and there are workarounds to cope with that problem.

A very intuitive method to find link or node disjoint paths in a network is based on the shortest path algorithm. The disjoint paths are obtained iteratively: once a shortest path between a pair of nodes is found, its links and interior nodes are removed from the topology. When no additional path can be found, the algorithm stops. This simple approach cannot always find disjoint paths (see Figure 2(a)) although a disjoint paths solution exist, or it may not always find the shortest disjoint paths (see Figure 2(b)). Bhandari's book [16] gives a good overview on different algorithms to find disjoint paths in networks and we use them in our software. In this work, we try to find at most 5 link and node disjoint paths for the path layout of the SPMs.



(a) It prohibits a another disjoint path.

(b) It increases the length of the shortest disjoint path.

Figure 2: Impact of the wrong selection of the first shortest path.

3.1.2 Adaptation to SRLGs

Shared risk link groups (SRLGs) are sets of links in a network that may fail simultaneously. Reasons may be, e.g., links on different wavelengths within a common fiber or links on different fibers within a common duct – they fail together in case of an electronic device failure or fiber cut. Another frequent reason for SRLGs are router failures. To work with SRLGs, the disjoint paths of SPMs should not contain links of the same SRLGs; otherwise, several paths of the SPM fail simultaneously and they do not protect each other anymore. Therefore, an adaptation of the paths layout to SRLGs must avoid links of common SRLGs on disjoint paths. This is a difficult NP-hard problem [17] which cannot be solved efficiently for general SRLGs. However, specific SRLGs can be respected efficiently, e.g. by node disjoint paths like in this work. The path layout for SPMs in case of SRLGs is not the focus of our work but rather the optimization of the path failure specific load balancing functions for SPMs in the next section.

3.2 Optimization of the Load Balancing Functions

The objective of this section is the optimization of the path failure specific load balancing functions for SPMs. First, we explain our notation of path concepts, then we introduce implications of failure scenarios, and finally, we propose two simple heuristics and an exact optimization for the load balancing functions to minimize the maximum link utilization of all protected failure scenarios.

3.2.1 Notation of Network Concepts

We introduce some basic notation from linear algebra that we use to model links, traffic aggregates, single paths, and multipaths.

Let $\mathbb{X}$ be a set of elements, then $\mathbb{X}^n$ is the set of all n-dimensional vectors and $\mathbb{X}^{n\times m}$ the is set of all $n \times m$-matrices with components taken from $\mathbb{X}$. Vectors $\mathbf{x} \in \mathbb{X}^n$ and matrices $\mathbf{X} \in \mathbb{X}^{n\times m}$ are written bold and their components are written as $\mathbf{x} = \begin{pmatrix} x_0 \\ \vdots \\ x_{n-1} \end{pmatrix}$ and $\mathbf{X} = \begin{pmatrix} x_{0,0} & \cdots & x_{0,m-1} \\ \vdots & & \vdots \\ x_{n-1,0} & \cdots & x_{n-1,m-1} \end{pmatrix}$. The scalar multiplication $c\mathbf{v}$ and the transpose operator $^\top$ are defined as usual. The scalar product of two n-dimensional vectors $\mathbf{u}$ and $\mathbf{v}$ is written with the help of matrix multiplication $\mathbf{u}^\top\mathbf{v} = \sum_{i=1}^{n} u_i \cdot v_i$. Binary operators $\circ \in \{+, -, \cdot\}$ are applied

component-wise, i.e. $\mathbf{u} \circ \mathbf{v} = (u_0 \circ v_0, \ldots, u_{n-1} \circ v_{n-1})^\top$. The same holds for relational operators $\circ \in \{<, \leq, =, \geq, >\}$, i.e. $\mathbf{u} \circ \mathbf{v}$ equals $\forall\ 0 \leq i < n\colon u_i \circ v_i$. For simplicity reasons we define special vectors $\mathbf{0} = (0, \ldots, 0)^\top$ and $\mathbf{1} = (1, \ldots, 1)^\top$ with context specific dimensions.

A network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ consists of $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ unidirectional links. The links are represented as unit vectors $\mathbf{e_i} \in \{0,1\}^m$, i.e. $(e_i)_j = 1$ if $i = j$, and $(e_i)_j = 0$ if $i \neq j$ for $0 \leq i, j < m$. We denote traffic aggregates between routers $\mathbf{v_i} \in \mathcal{V}$ and $\mathbf{v_j} \in \mathcal{V}$ by $d = (i, j)$ and the set of all aggregates by $\mathcal{D} = \{(i, j) : 0 \leq i, j < n \text{ and } i \neq j\}$. A single path p between two distinct nodes is a set of contiguous links represented by a link vector $\mathbf{p} \in \{0,1\}^m$. The basic structure of an SPM for a traffic aggregate d is a multipath $\mathbf{P_d}$ that consists of k_d paths $\mathbf{p_d^i}$ for $0 \leq i < k_d$ that are link and possibly also node disjoint except for their source and destination nodes. It is represented by a vector of single paths $\mathbf{P_d} = (\mathbf{p_d^0}, ..., \mathbf{p_d^{k_d-1}})$. Thus, a multipath is described by a matrix $\mathbf{P_d} \in \{0,1\}^{k_d \times m}$.

3.2.2 Implications of Failure Scenarios

A failure scenario s is given by a set of failing links and nodes. The set of protected failure scenarios $\mathcal{S}$ contains all outage cases including the normal working case for which the SPM should protect the traffic from being lost. The failure indication function $\phi(\mathbf{p}, s)$ yields 1 if a path p is affected by a failure scenario s; otherwise, it yields 0. The failure symptom of a multipath $\mathbf{P_d}$ is the vector $\mathbf{f_d(s)} = \left(\phi(\mathbf{p_d^0}, \mathbf{s}), ..., \phi(\mathbf{p_d^{k_d-1}}, \mathbf{s})\right)^\top$ and indicates its failed single paths in case of failure scenario s. Thus, with a failure symptom of $\mathbf{f_d} = \mathbf{0}$, all paths are working while for $\mathbf{f_d} = \mathbf{1}$ connectivity cannot be maintained. In this work, we take the protection of all single link or node failures into account such that at most one single path of an SPM multipath fails. The set of all different failure symptoms for the SPM $\mathbf{P_d}$ is denoted by $\mathcal{F}_d = \{\mathbf{f_d(s)} : s \in \mathcal{S}\}$.

Normally, all traffic aggregates $d \in \mathcal{D}$ are active. If routers fail, some demands disappear which leads to a traffic reduction that is expressed by the failure scenario specific set of aggregates $\mathcal{D}_s$.

- *No Traffic Reduction (NTR):* We assume hypothetically that failed routers lose only their transport capability for transit flows but they are still able to generate traffic. Therefore, we have $\mathcal{D}_s = \mathcal{D}$.
- *Source Traffic Reduction (STR):* If a certain router fails, all traffic aggregates with this source node disappear.
- *Full Traffic Reduction (FTR):* We assume that traffic aggregates with failed source *or* destination are stalled.

We use FTR for the computation of the results in this paper, but we considered all options for network dimensioning in [18] and analyzed their impact.

3.2.3 The Load Balancing Function and Simple Heuristics

There is one SPM for each traffic aggregate $d \in \mathcal{D}$. This SPM has a load balancing function to distribute the traffic over its k_d different paths. If certain paths fail, which is indicated by the symptom $\mathbf{f_d(s)}$, the load balancing function shifts the traffic to the remaining working paths. Thus, the SPM needs a load balancing function $\mathbf{l_d^f}$ for each symptom $\mathbf{f} \in \mathcal{F}_d$ that results from any protected failure scenarios $s \in \mathcal{S}$. Since the load balancing function $\mathbf{l_d^f} \in (\mathbb{R}_0^+)^{k_d}$ describes a distribution, it must obey

$$\mathbf{1}^\top \mathbf{l_d^f} = 1. \tag{1}$$

Furthermore, failed paths must not be used, i.e.

$$\mathbf{f}^\top \mathbf{l_d^f} = 0. \tag{2}$$

A simple example for load balancing function is equal load balancing over all working paths, i.e., $\mathbf{l_d^f} = \frac{1}{\mathbf{1}^\top(\mathbf{1}-\mathbf{f})} \cdot (\mathbf{1}-\mathbf{f})$. Another relatively simple option is balancing the load over the partial paths $\mathbf{p_d^i}$ indirectly proportionally to their length $(\mathbf{1}^\top \mathbf{p_d^i})$. This can be computed by $(l_d^f)_i = \frac{1-f_i}{\mathbf{1}^\top \mathbf{p_d^i}} / \left(\sum_{0 \le j < k_d} \frac{1-f_j}{\mathbf{1}^\top \mathbf{p_d^j}} \right)$. Both heuristics require a lot of backup capacity [2]. Therefore, optimization of the load balancing function is required.

3.2.4 Optimization of the Load Balancing Function

The optimization configures the load balancing functions in such a way that the maximum link utilization ρ_{max} is minimal in any failure scenario $s \in \mathcal{S}$ for given link capacities and a given traffic matrix.

The traffic rate associated with each traffic aggregate $d \in \mathcal{D}$ is given by $c(d)$ and corresponds to an entry in the traffic matrix. We describe the network capacity by a bandwidth vector $\mathbf{b} \in (\mathbb{R}_0^+)^m$, which carries a capacity value for each link. Similarly, the vector indicating the traffic rates on all links, which are induced by a specific SPM $\mathbf{P_d}$ and a specific failure symptom $f \in \mathcal{F}_d$, is calculated by $\mathbf{P_d} \cdot \mathbf{l_d^f} \cdot c(d)$.

We now formulate constraints for the traffic transport over the network in all protected scenarios $s \in \mathcal{S}$ under the side constraint that all links have a maximum utilization of ρ_{max}. In packet switched networks, resources are not physically bound to traffic aggregates. If traffic is rerouted due to a local outage, the released resources can be immediately reused for the transport of other traffic. Under this assumption, the capacity constraints are

$$\forall s \in \mathcal{S} : \sum_{d \in \mathcal{D}_s} \mathbf{P_d} \cdot \mathbf{l_d^{f_d(s)}} \cdot c(d) \le \mathbf{b} \cdot \rho_{max}. \tag{3}$$

In [2, 18], we have also proposed constraints that apply when capacity cannot be reused, but we have investigated them in the context of network dimensioning.

The objective of the optimization is the minimization of the maximum link utilization ρ_{max}. The free variables, which must be set in the optimization process, are the load balancing functions $\forall d \in \mathcal{D}\ \forall \mathbf{f} \in \mathcal{F}_d : \mathbf{l_d^f} \in (\mathbb{R}_0^+)^{k_d}$ and the maximum link utilization ρ_{max} itself. The following constraints must be respected in the optimization process to obtain valid load balancing functions and to avoid overload on the links.

- **(C0)**: Equation (1) assures that the load balancing function is a distribution.
- **(C1)**: Equation (2) assures that failed paths will not be used.
- **(C2)**: Equation (3) assures that the bandwidth suffices to carry the traffic in all protected failure scenarios.

3.3 Analysis of the Linear Program Complexity

We estimate the number of free variables and the number of constraints of the LP depending on the network size since they influence its computation time and memory consumption.

3.3.1 Number of Free Variables

The maximum link utilization ρ_{max} is just a single free variable. The consideration of the load balancing functions $\mathbf{l}_d^{f_d(s)}$ is more complex. One SPM exists for each traffic aggregate $d \in \mathcal{D}$ and for each SPM a load balancing function $\mathbf{l_d^f}$ is needed for every SPM failure symptom $f \in \mathcal{F}_d$. A load balancing vector has an entry for each of the k_d paths of the SPM. There is one load balancing vector for each SPM failure symptom. We take all single link and node failures into account in addition to the working scenario, so we have exactly $|\mathcal{F}_d| = k_d + 1$ different failure symptoms. We use a full traffic matrix in our study, thus, the number of traffic aggregates is $|\mathcal{D}| = n \cdot (n-1)$. We denote the average number of outgoing links per node by the average node degree deg_{avg} which can be calculated by $deg_{avg} = \frac{m}{n}$. The average number of disjoint paths for all SPMs is given by $k^* = \frac{1}{|\mathcal{D}|} \cdot \sum_{d \in \mathcal{D}} k_d$ and it is smaller than the average node degree deg_{avg}. Taking this into account, the overall number of free variables is $\sum_{d \in \mathcal{D}} k_d \cdot (k_d + 1) \approx n \cdot (n-1) \cdot k^* \cdot (k^* + 1) \leq m^2$. Thus, the number of free variables scales quadratically with the number of links in the network.

3.3.2 Number of Constraints

We calculate the number of constraints resulting from (C0), (C1), and (C2) of the previous section. Both (C0) and (C1) require for each path failure specific load balancing function one constraint such that we get $n_{C0} = n_{C1} = \sum_{d \in \mathcal{D}} (k_d + 1) \approx n \cdot m$ different equations. Constraint type (C2) requires an equation for each link and for each protected failure scenario, i.e. for the working scenario and all single link and node failures. Therefore, the number of constraints for (C2) is exactly $n_{C2} = m \cdot (1 + m + n)$. Thus, the overall number of constraints is roughly $m^2 + 3 \cdot m \cdot n + m$. Hence, the number of constraints also scales about quadratically with the number of links in the network.

4 Results

In this section, we show first the efficiency of the SPM as protection switching mechanism. Then, we illustrate the computation time and the memory requirements of the above

described optimization algorithm for four different LP solving approaches and illustrate the dependency of the computation time on the network structure size.

4.1 Efficiency of the SPM as a Protection Switching Algorithm

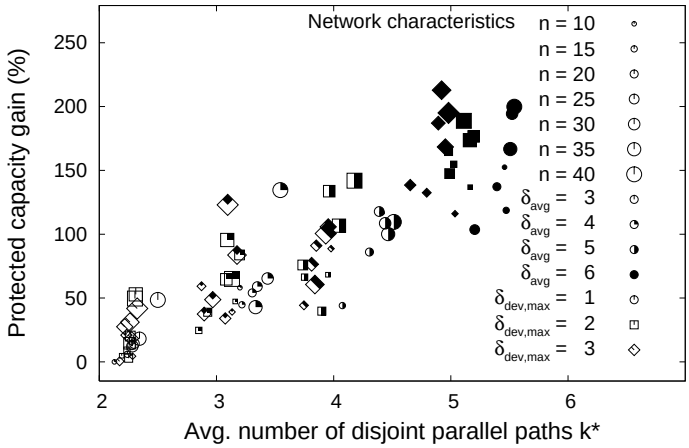


Figure 3: Protected transmission gain of the SPM compared to SPR for random networks depending on their average number of average parallel paths.

We show by means of a multitude of sample networks that the SPM is a very efficient protection switching mechanism. The degree of a network node is the number of its outgoing links. We construct sample networks for which we control the number of nodes $n \in \{10, 15, 20, 25, 30, 35, 40\}$, the average node degree $deg_{avg} \in \{3, 4, 5, 6\}$, and the deviation of the individual node degree from the average node degree $deg_{max} \in \{1, 2, 3\}$. We use the algorithm of [2] for the construction of these networks since we cannot control these parameters rigidly with the commonly used topology generators [19–23]. We sampled 5 networks for each of the 84 different network characteristics and tested altogether 420 different networks.

We consider the maximum link utilization of a network in all single link and router failure scenarios $s \in \mathcal{S}$ and compare it for SPM $\left(\rho_{max}^{SPM}\right)$ and shortest path rerouting $\left(\rho_{max}^{SPR}\right)$ based on the hop count metric. We define the protected capacity gain $\gamma = \rho_{max}^{SPR}/\rho_{max}^{SPM}$ to express how much more traffic can be transported by SPM than by SPR at the same maximum link utilization in the network. Figure 3 shows the protected capacity gain for these networks under the assumption of a homogenous traffic matrix and homogeneous link bandwidths, i.e. the entries of the traffic matrix are all the same and all links of a network have the same bandwidth. Each point in the figure stands for the average result of the 5 sample networks with the same characteristics. The shape and the size of the points determines the network characteristics, the corresponding x-coordinates indicate the average number of disjoint paths k^* for the SPMs in networks, and the y-coordinates show the protected capacity gain of the SPM. The figure reveals

an obvious trend: the protected capacity gain of the SPM increases significantly with an increasing number of disjoint parallel paths k^* in the networks. Networks with the same average node degree deg_{avg} are obviously clustered since the average node degree deg_{avg} and k^* are strongly correlated. Networks with a small deviation deg_{dev}^{max} regarding their average node degree (circles) have a larger k^* than those with a large deg_{dev}^{max} (diamonds). Large networks lead to a slightly larger protected capacity gain than small networks, however, this trend is not so obvious. After all, the SPM is quite efficient since it can carry 50% to 200% more protected traffic than SPR in sufficiently meshed networks.

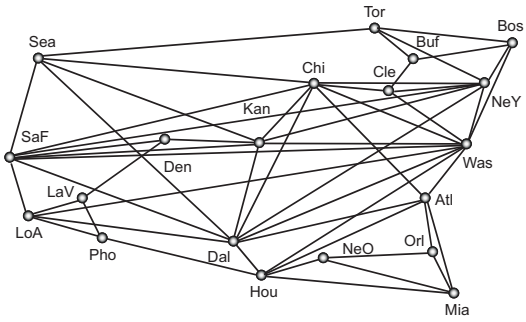


Figure 4: Topology of the Labnet03 network.

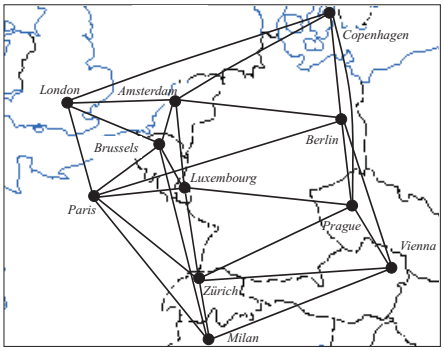


Figure 5: Topology of the COST239 network.

Finally, we consider two real networks whose topologies are depicted in Figure 4 and Figure 5. The first one is an experimental network from the project KING [24] while the second one comes with a provisioned links and a real traffic matrix [25, 26]. The SPM leads to 61.5% more protected capacity in the Labnet03 and to 138% more in the COST239 network compared to SPR rerouting. If we take into account the link capacities and the traffic matrix of the COST239 network, the SPM achieves even 109%

more protected capacity than SPR rerouting. Thus, SPR is inadequate if the traffic matrix and the capacity provisioning do not fit well together. In contrast, SPM copes well with that situation.

4.2 Experimental Runtime Analysis of the Optimization Algorithm

We study the optimization algorithm regarding its computation time and memory consumption since both have a tremendous impact on its feasibility in practice. First, we give a short introduction to linear programs (LP) and an overview on the tested solvers. Then, we report on the memory consumption and the computation time of the different LP solvers that were required to calculate the numerical results above.

4.2.1 Linear Programs and LP Solvers

The solutions of LPs may consist of rational numbers, they may be restricted to integer solutions, then the problems are called integer (linear) programs (IP, ILP), or they may be partly restricted to integer solutions, then the problems are called mixed integer (linear) programs (MIP, MILP)) [27]. ILPs or MILPs are NP-complete problems. Fortunately, our LP formulation has a rational solution. Therefore, it can be used by the Simplex algorithm or by interior point methods (IPMs). Simplex is quite fast in general but it has an exponential runtime in the worst case. In contrast, IPMs run in polynomial time [28] but they are more complex. We implemented the above LP with the following four different free available LP solvers.

- **GLPK**: The GNU Linear Programming Kit [29] which offers both a Simplex and an IPM based solver, but we show only the results for the Simplex option since it is faster.
- **BPMPD**: The BPMPD solver which is based on IPMs [30].
- **CLP**: The COmputational INfrastructure for Operations Research (COIN-OR) solver which is also called CLP solver [31].
- **LPSolve**: The LPSolve solver [32].

Due to license issues, we avoided commercial standard software. We used the SuSE 9.1 operating system on an Intel Pentium 4 with a CPU of 3.20 GHz and 2 GB RAM to produce the following results.

4.2.2 Computation Time of Different LP Solvers

We optimized all networks with each of the four different LP solvers and measured the computation time. Figure 6 shows the average computation time for each solver depending on the network size in links. The almost straight lines in the double-logarithmic plot show that the computation time increases polynomially with the number of links. The computation time differs clearly among the solvers. CLP can solve even the largest

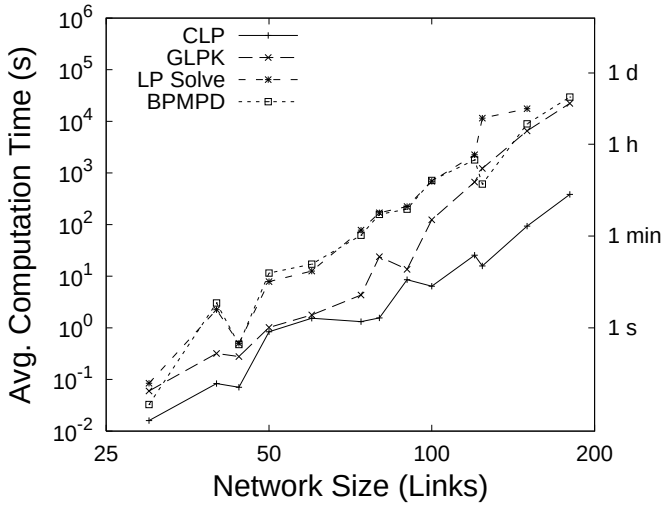


Figure 6: Average computation time of the GLPK, BPMPD, CLP, and LPSolve for the optimization of different random networks depending on the network size in links only.

networks within an hour for which the other products take longer than a day. Therefore, we recommend the CLP solver for the implementation of the optimization program.

4.2.3 Memory Consumption of Different LP Solvers

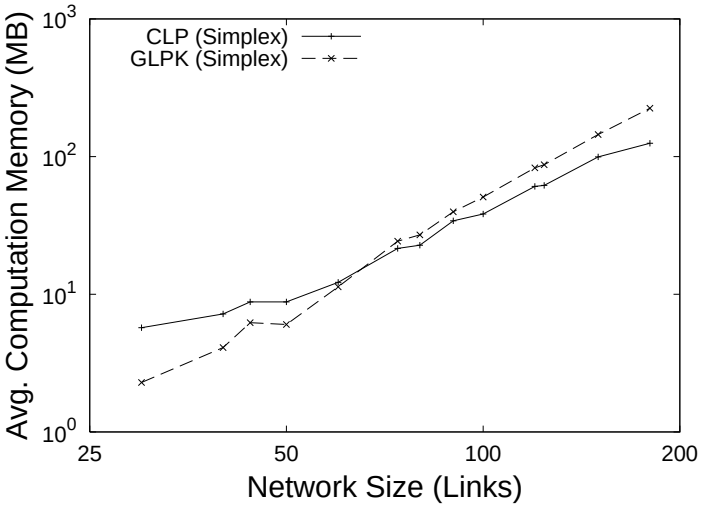


Figure 7: Average memory consumption of the CLP and GLPK solvers for the optimization of different random networks depending on the network size in links only.

Figure 7 shows the average memory memory consumption of the optimization program depending on the network size in links for the CLP and the GLPK since it seems to be the second fastest LP solver. Again, the straight lines show a polynomial growth of the memory consumption with the number of links in the network. For small networks the program size of CLP is significantly larger than the one of GLPK, but for large networks the relation is vice-versa. Hence, CLP is the suitable LP solver for our optimization problem both from a computation time and memory consumption point of view.

4.2.4 Detailed Analysis of the Computation Time for the CLP

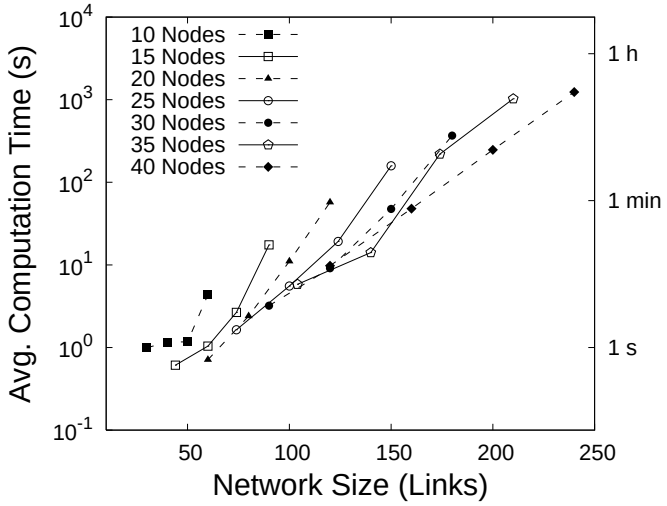


Figure 8: Average computation time of the CLP for the optimization of different random networks depending on the network size in links and nodes.

Figure 8 shows the average computation time of the optimizations using the CLP depending on network size in links and in nodes. The number of links m in the network has a clearly larger impact on the computation time than the number of nodes n which confirms our theoretical findings in Section 3.3. If networks have the same size in terms of links but not in terms of nodes, it takes more time to optimize the SPM for the networks with fewer nodes. Those networks have a larger average node degree $deg_{avg} = \frac{m}{n}$ than the others and thereby a larger average number k^* of disjoint parallel paths per source-destination pair. We used the approximation $k_d \approx k^* \approx \frac{m}{n}$ for the analysis of the program complexity, but it is more accurate for large deg_{avg}. Thus, the complexity of linear programs for networks with smaller deg_{avg} is overestimated compared to those with larger deg_{avg} and the same number of links. Therefore, they run faster.

5 Conclusion

In this paper we have reviewed several protection switching mechanisms and, in particular, the self-protecting multipath (SPM). Its structure is composed of disjoint paths that can be calculated by a shortest disjoint paths algorithm. The traffic is distributed over these paths according to a load balancing function that can be optimized in such a way that the maximum link utilization of all links is minimized in all protected failure scenarios. This minimization is equivalent to a maximization of the protected transport capacity of the network. We formulated the optimization algorithm for the load balancing functions as a linear program (LP).

We performed a numerical study based on random and existing networks and took into account the protection of all single link and node failures. We showed that the SPM is a very efficient protection switching mechanism since the SPM outperforms standard IP rerouting based on shortest paths: 50% - 200% more protected traffic can be carried if sufficiently many disjoint paths can be found in a network. We first analyzed the complexity of the LP theoretically and then illustrated its computation time and memory consumption experimentally. The program complexity is dominated by the number of links in the network and both the computation time and the memory consumption scale polynomially. We studied several LP solvers and the COIN LP (CLP) proved to be the most suitable solver since it was both the fastest one and the one with the least memory consumption.

After all, the SPM is a capacity-efficient and simple protection switching mechanism and, therefore, its application in practice is of interest. It is well applicable in small and medium size networks due to the moderate computation time and memory demand of the optimization program which is required for its configuration. However, the configuration of the SPM in large networks requires a fast heuristic algorithm which is one of our current research issues.

Acknowledgment

The authors would like to thank Prof. Tran-Gia for the stimulating environment which was a prerequisite for that work.

References

[1] M. Menth, A. Reifert, and J. Milbrandt, "Self-Protecting Multipaths - A Simple and Resource-Efficient Protection Switching Mechanism for MPLS Networks," in 3^{rd} *IFIP-TC6 Networking Conference (Networking)*, (Athens, Greece), pp. 526 – 537, May 2004.

[2] M. Menth, *Efficient Admission Control and Routing in Resilient Communication Networks*. PhD thesis, University of Würzburg, Faculty of Computer Science, Am Hubland, July 2004.

[3] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot, "IGP Link Weight Assignment for Transient Link Failures," in 18^{th} *International Teletraffic Congress (ITC)*, (Berlin), Sept. 2003.

[4] B. Fortz and M. Thorup, "Robust Optimization of OSPF/IS-IS Weights," in *International Network Optimization Conference (INOC)*, (Paris, France), pp. 225–230, Oct. 2003.

[5] A. Autenrieth and A. Kirstädter, "Engineering End-to-End IP Resilience Using Resilience-Differentiated QoS," *IEEE Communications Magazine*, vol. 40, pp. 50–57, Jan. 2002.

[6] P. Pan, G. Swallow, and A. Atlas, "RFC4090: Fast Reroute Extensions to RSVP-TE for LSP Tunnels," May 2005.

[7] H. Saito and M. Yoshida, "An Optimal Recovery LSP Assignment Scheme for MPLS Fast Reroute," in *International Telecommunication Network Strategy and Planning Symposium (Networks)*, pp. 229–234, June 2002.

[8] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Fast Recovery from Link Failures Using Resilient Routing Layers," in *IEEE Symposium on Computers and Communications (ISCC)*, (Cartagena, Spain), June 2005.

[9] M. Menth and R. Martin, "Network Resilience through Multi-Topology Routing," in *The* 5^{th} *International Workshop on Design of Reliable Communication Networks*, (Island of Ischia (Naples), Italy), pp. 271 – 277, Oct. 2005.

[10] M. Shand and S. Bryant, "IP Fast Reroute Framework." `http://www.ietf.org/internet-drafts/draft-ietf-rtgwg-ipfrr-framework-06.txt`, Oct. 2006.

[11] A. Atlas and A. Zinin, "Basic Specification for IP Fast-Reroute: Loop-Free Alternates." `http://www.ietf.org/internet-drafts/draft-ietf-rtgwg-ipfrr-spec-base-05.txt`, July 2005.

[12] K. Murakami and H. S. Kim, "Optimal Capacity and Flow Assignment for Self–Healing ATM Networks Based on Line and End-to-End Restoration," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 207–221, Apr. 1998.

[13] B. Fortz, J. Rexford, and M. Thorup, "Traffic Engineering with Traditional IP Routing Protocols," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 118–124, 2002.

[14] S. Köhler and A. Binzenhöfer, "MPLS Traffic Engineering in OSPF Networks - A Combined Approach," in 18^{th} *International Teletraffic Congress (ITC)*, (Berlin, Germany), Sept. 2003.

[15] M. Pióro and D. Medhi, *Routing, Flow and Capacity Design in Communication and Computer Networks*. Morgan and Kaufman, June 2004.

[16] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing.* Norwell, MA, USA: Kluwer Academic Publishers, 1999.

[17] J. Q. Hu, "Diverse Routing in Optical Mesh Networks," *IEEE/ACM Transactions on Networking*, vol. 51, no. 3, pp. 489 – 494, 2003.

[18] M. Menth, J. Milbrandt, and A. Reifert, "Sensitivity of Backup Capacity Requirements to Traffic Distribution and Resilience Constraints," in 1^{st} *Conference on Next Generation Internet Design and Engineering (NGI)*, (Rome, Italy), Apr. 2005.

[19] B. M. Waxman, "Routing of Multipoint Connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[20] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A Quantitative Comparison of Graph-Based Models for Internet Topology," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 770–783, 1997.

[21] C. Jin, Q. Chen, and S. Jamin, "´Inet: Internet Topology Generator," Tech. Rep. CSE-TR-433-00, Department of EECS, University of Michigan, USA, 2000.

[22] A. Medina, I. Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generation," in *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, (Cinncinnati, Ohio, USA), Aug. 2001.

[23] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topology Generators: Degree-Based vs. Structural," in *ACM SIGCOMM*, Aug. 2002.

[24] C. Hoogendoorn, K. Schrodi, M. Huber, C. Winkler, and J. Charzinski, "Towards Carrier-Grade Next Generation Networks," in *International Conference on Communication Technology (ICCT)*, (Beijing, China), April 2003.

[25] P. Batchelor et al., "Ultra High Capacity Optical Transmission Networks. Final Report of Action COST 239," Jan. 1999.

[26] H.-P. C. Mauz, *Vergleich der Effizienz von Schutzstrategien in photonischen Transportnetzen.* PhD thesis, ETH Zürich, 2003.

[27] Optimization Technology Center of Northwestern University and Argonne National Laboratory, "Linear Programming Frequently Asked Questions." `http://www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html`, Sept. 2005.

[28] N. Karmakar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica*, vol. 4, pp. 373 – 396, 1984.

[29] The Free Software Foundation, "The GNU Linear Programming Kit (GLPK) Version 4.8." `http://www.gnu.org/software/glpk/glpk.html`, 2005.

[30] C. Mészáros, "BPMPD Web Site." http://www.sztaki.hu/ meszaros/bpmpd/, 1998.

[31] J. Forrest, "The COIN-OR Linear Program Solver (CLP)." `http://www.coin-or.org`, 2005.

[32] M. Berkelaar, K. Eikland, and P. Notebaert, "Web Site for lp_solve." `http://groups.yahoo.com/group/lp\_solve`, 2005.