Path Generation Issues for Survivable Network Design*

Michał Pióro, Tomasz Śliwiński, Michał Zagożdżon,
Mateusz Dzida, and Włodzimierz Ogryczak

Warsaw Univ. of Technology, Faculty of Electronics & Info. Technology, Warsaw, Poland
(tsliwins,wogrycza)@ia.pw.edu.pl,
(mpp,mzagozdz,mdzida)@tele.pw.edu.pl

Abstract. Link dimensioning and routing problems in resilient network design are considered. Reliable network operation is ensured by means of flow restoration which is performed on preselected protection (backup) paths that can absorb traffic overflows from failed primary paths. Backup and primary flows use separated link capacities, and can be split among many paths. In the paper, two restoration models are considered. The first model assumes that once the backup path is assigned it must be used in every state in which the protected primary path fails while the second model allows different protection paths to be used in different network failure states. The problems are formulated as multiple commodity linear programming (LP) models using the link-path (L-P) notation and solved by the column generation technique. Consequent pricing models and algorithms are introduced. Computational efficiency of the presented approaches is analyzed.

1 Introduction

Large capacity of optical transmission systems in telecommunication networks can drastically influence the network performance in the case of network link failures. The implied increasing aggregation level of the traffic flows makes them more and more sensitive to such failures. For this reason, transport network operators are interested in providing reliable services, that thanks to fast recovery mechanisms are able to survive network failures. To provide reliable transport services, the transport network operator must foresee failures that can appear in the network, and accordingly design protection (backup) paths that can absorb the traffic overflows from failed primary paths that are used in the normal, failure-less state of network operation. This is referred to as *flow restoration*. In this paper, we investigate link dimensioning and flow routing design problems for resilient (i.e., robust to failures) networks that assume flow restoration of affected flows against the foreseen failures. Hence, the problem consists in searching for a configuration of paths for each operating state, such that the capacity installation cost resulting from realization of given traffic demands is minimized. Throughout the paper we assume that capacity of the primary flows is separated from the capacity used by the protection flows. There is a strong motivation for this assumption, stemming from practical aspects of the network operation. Having reserved capacity for primary flows

* The research was partially supported by the Dean's grant from the Faculty of Electronics and Information Technology, Warsaw University of Technology.

O. Gervasi et al. (Eds.): ICCSA 2008, Part II, LNCS 5073, pp. 820–835, 2008.

and establishing them, a network operator is in fact able to immediately bring back primary path flows once a failure is removed. Such capacity model is known as the case of *no stub-release*, since it does not allow protection flows to use the basic capacity that could be released due to failing primary paths.

As indicated above, we assume that network survivability is achieved by means of a flow restoration mechanism. We distinguish two problem variants, depending on the assumptions concerning restoration of affected primary flows. The first, called *state-independent flow restoration* assumes that once a backup path is assigned, it is used in every state in which the protected primary path fails. The second model, though, allows different protection paths to be used in different network failure states and is therefore called *state-dependent flow restoration*. Thus, in this paper we assume that each primary flow is protected by its backup paths (state-dependant or not), and bifurcation is permitted for each primary flow as well for its protection flows.

The considered resilient design problems are well recognized in the literature (see [1] and discussion there) and were considered in many research papers, for example in [2,3,4,5,6,7,8,9,10]. The two problems considered in this paper have a property that is common in resilient network design. Both problems are $\mathcal{NP}$-complete and cannot be formulated through LP formulations in a compact way. Still, they can be formulated using the Link-Path (L-P) notation (see [11,1]), but certainly not in a compact way. Hence, the use of L-P formulations is useful only in two situations: when all required paths can be used in the formulation, or when an efficient path (column) generation method can be proposed. Since the former solution is not scalable with growing number of nodes, we focus on the latter method.

Path generation (PG) is an application of the general column generation technique, developed to effectively solve large-scale LP problems of special structure (see [11]). The PG framework can be described as follows. Having given traffic demands and costs of link capacities, we determine preliminary sets of the candidate paths, containing, for each demand, at least one primary path together with its necessary backup paths. Using these sets we formulate an LP problem in the L-P notation. Having the LP problem in hand, we treat it with an LP solver and get the values of the dual variables related to the L-P problem constraints. The dual equalities allow us to verify if the current sets of candidate paths should be extended in order to get an optimal flow distribution in the sense of minimal capacity cost. If the current set of candidate paths for a specific demand does not contain a path (either primary or backup) that due to the dual constraints may be necessary, then the so-called *pricing problem* is formulated and the required path is obtained as a solution of this problem. The procedure consists of re-optimizing the LP problem and introducing new paths generated by the pricing problem, until no new path is necessary.

Application of the PG technique allows one to keep the lists of candidate paths very short with respect to the sets of all possible paths. Thanks to that, the corresponding LP problem can be solved very efficiently. Still, the efficiency of the overall technique depends on the efficiency of the path generation method resolving the pricing problems. It appears that for some network failure scenarios the pricing problem can become $\mathcal{NP}$-complete [6,5,7,8]. One of the investigated approaches is to solve pricing problems formulated as mixed-integer (MIP) programmes. Another approach assumes a special

variant of a shortest path algorithm, namely a label-setting algorithm [12], that allows to compute shortest paths with respect to additional resource constraint requirements. In our case these requirements are induced by backup paths used in the failure states in which the generated primary path fails. In any case, exact generation methods are sometimes inefficient, still in such cases it is often sufficient to use path lists produced by a heuristic method. Therefore, we also propose a heuristic for solving the considered PD problems. Its application makes PG quite easy and allows to obtain solutions of good quality for the overall problem. The heuristic is based on precomputed sets of potential primary paths. Then, the pricing problem consists in searching through these sets and selecting the best path for each demand. The presented approach assumes that the sets of candidate primary paths are initially determined as a solution of a K-shortest path algorithm.

The paper is organized as follows. In Section 2 we give mathematical formulations for the problems under consideration. Section 3 introduces the related pricing problems. There are also presented exact and approximate approaches to generate efficiently new columns (paths). The paper is summarized with the discussion on computational experiments (Section 4) and some concluding remarks (Section 5).

2 The Survivability Models

2.1 Network Model

The considered network is modeled with a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ composed of a set of nodes $\mathcal{V}$ and a set of (directed) links $\mathcal{E}$ between the nodes. For ease of exposition, we assume that the graph does not contain loops nor parallel links, i.e., $\mathcal{E} \subseteq \mathcal{V}^2 \setminus \{(v, v) : v \in \mathcal{V}\}$. For any link $e \in \mathcal{E}$, its source node is denoted by u_e, whereas the target node is denoted by v_e, so that $e = (u_e, v_e)$. The cost of realizing one unit of demand on link $e \in \mathcal{E}$ is denoted by ξ_e.

The set $\mathcal{D} \subseteq \mathcal{V}^2$ represents (directed) point-to-point demands. For notational convenience, at most one demand between each ordered pair of nodes is assumed. The source and target of a demand $d \in \mathcal{D}$ are denoted by u_d and v_d, respectively, and assumed to be different from each other. The demand value of demand $d \in \mathcal{D}$ is given by $h_d > 0$.

Let $\mathcal{S} \subseteq 2^{\mathcal{E}}$ be the family of all considered failure states (failure scenario) where each state (also called failure situations) $s \in \mathcal{S}$ is identified with the set of its failing links. Family $\mathcal{S}$ is assumed to include the failure-less state $\mathcal{O}$ (formally equal to the empty subset of $\mathcal{E}$) in which all links are operational (state $\mathcal{O}$ is also called the *normal state*). It is assumed that links fail totally. The sets of surviving links in state $s \in \mathcal{S}$ are denoted by $\mathcal{E}^s \subseteq \mathcal{E}$. Similarly, $\bar{\mathcal{E}}^s = \mathcal{E} \setminus \mathcal{E}^s$ denotes the set of failing links in state $s \in \mathcal{S}$. Certainly, $\mathcal{E}^{\mathcal{O}} = \mathcal{E}$.

We observe that the node failures can be easily modeled by link failures. This is achieved by a suitable transformation of the network graph. We simply substitute each node $v \in \mathcal{V}$ that can fail by two nodes v' and v'' connected by a dummy link $e_v = (v', v'')$ with infinite capacity and zero cost. Also, in the transformed graph we terminate all original links $e \in \mathcal{E}$ such that $v_e = v$ at node v', and start all original links $e \in \mathcal{E}$ such that $u_e = v$ at node v''.

Let $\mathcal{P} := \bigcup_{d\in\mathcal{D}} \mathcal{P}_d$ be the set of all candidate (undirected) paths that can be used for realizing demand flows, where $\mathcal{P}_d$ is the set of candidate paths for demand $d \in \mathcal{D}$. We assume that paths do not contain loops so for each demand $d \in \mathcal{D}$ the set $\mathcal{P}_d$ is a subset of all simple paths from u_d to v_d. Note that the sets $\mathcal{P}_d, d \in \mathcal{D}$ are mutually disjoint (since we have assumed at most one demand per node-pair) and in consequence, each path $p \in \mathcal{P}$ can be identified with the set of the links it traverses, so that $p \subseteq \mathcal{E}$.

Notation $\mathcal{P}_d^s \subseteq \mathcal{P}_d$ refers to the set of all those candidate paths for demand $d \in \mathcal{D}$ that are available in state $s \in \mathcal{S}$, i.e., $\mathcal{P}_d^s := \{p \in \mathcal{P}_d \mid p \cap s = \emptyset\}$. Similarly, $\bar{\mathcal{P}}_d^s = \{p \in \mathcal{P}_d \mid p \cap s \neq \emptyset\}$ denotes the set of all candidate paths for demand $d \in \mathcal{D}$ that fail in state s, so that $\mathcal{P}_d^s \cup \bar{\mathcal{P}}_d^s = \mathcal{P}_d$. Furthermore, $\mathcal{P}_e \subseteq \mathcal{P}$ is the set of all paths containing link $e \in \mathcal{E}$, and $\mathcal{P}_e^s := \mathcal{P}_e \cap \left(\bigcup_{d\in\mathcal{D}} \mathcal{P}_d^s\right)$ denotes the set of all paths containing link $e \in \mathcal{E}$ that are available in state $s \in \mathcal{S}$.

For convenience, we also introduce the notation $\mathcal{S}_p \subseteq \mathcal{S}$ for the set of all states $s \in \mathcal{S}$ in which path $p \in \mathcal{P}$ is available, i.e., $\mathcal{S}_p = \{s \in \mathcal{S} \mid p \cap s = \emptyset\}$, and $\bar{\mathcal{S}}_p := \mathcal{S} \setminus \mathcal{S}_p$ is the set of all states $s \in \mathcal{S}$ where path $p \in \mathcal{P}$ fails, i.e., $\bar{\mathcal{S}}_p = \{s \in \mathcal{S} \mid p \cap s \neq \emptyset\}$. Similarly, $\mathcal{S}_e = \{s \in \mathcal{S} \mid e \notin s\}$ will denote the set of all states $s \in \mathcal{S}$ in which link $e \in \mathcal{E}$ is available, and $\bar{\mathcal{S}}_e = \mathcal{S} \setminus \mathcal{S}_e$— the set of all failure states in which it is failed.

For the restoration paths (called also protection or backup paths) we will use the following notation. For each demand $d \in \mathcal{D}$ and each path $p \in \mathcal{P}_d$, the set $\mathcal{Q}_p$, $\mathcal{Q}_p \subseteq \mathcal{P}_d$ denotes all candidate backup paths $q \in \mathcal{P}_d$ that never fail together with p. Hence, if $q \in \mathcal{Q}_p$ then for all $s \in \mathcal{S}$, $(p \cap s \neq \emptyset) \Rightarrow (q \cap s = \emptyset)$, i.e., paths p and q never fail simultaneously (and therefore are called *failure-disjoint*). Besides, $\mathcal{Q}_{pe} := \{q \in \mathcal{Q}_p \mid q \ni e\}$ denotes the set of all paths protecting path p that contain a particular link $e \in \mathcal{E}$.

2.2 State-Dependent Restoration – FD

Both restoration models considered in this paper assume protection of traffic against node or link failures using flow restoration. We also adopt a natural assumption that in the case of a failure of one or more links (multiple failure), only the flows affected by this failure are restored, and the flows not containing any of the failed links are not modified. This mechanism is sometimes referred to as "restricted reconfiguration" (see [1]). In the model considered in this section we assume that the failed flows are restored in a failure-dependent fashion, i.e., the flow restoration pattern depends on the particular failure state. We also assume that the failed flows are restored using protection capacity of links that is separated from the basic capacity on the links used by primary flows. As already mentioned, this is known as path restoration "without stub release". Hence, we note that separated protection capacity assumes that each link $e \in \mathcal{E}$ has its "primary" (basic) capacity, denoted by y'_e, used for realizing flows in the normal operating state (i.e., primary flows), and the protection capacity, y''_e, used for restoration of failed primary flows. The (primal) formulation of the state-dependent flow restoration (FD) design problem is as follows (alternative formulations can be found in [1]):

$$\text{minimize } F(\boldsymbol{y}) = \sum_{e\in\mathcal{E}} \xi_e(y'_e + y''_e) \tag{1a}$$

$$\sum_{p\in\mathcal{P}_d} x_p \geq h_d \quad d\in\mathcal{D} \tag{1b}$$

$$\sum_{p\in\mathcal{P}_e} x_p \leq y'_e \quad e\in\mathcal{E} \tag{1c}$$

$$\sum_{p\in\mathcal{P}_d^s} x_p \leq \sum_{q\in\mathcal{P}_d^s} z_{qs} \quad d\in\mathcal{D},\ s\in\mathcal{S}\setminus\{\mathcal{O}\} \tag{1d}$$

$$\sum_{q\in\mathcal{P}_e^s} z_{qs} \leq y''_e \quad e\in\mathcal{E},\ s\in\mathcal{S}_e\setminus\{\mathcal{O}\} \tag{1e}$$

$$\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\geq\mathbf{0}. \tag{1f}$$

Formulation (1) uses flow variables $x_p \geq 0$ for the primary paths $p \in \mathcal{P}$, and backup path flow variables z_{qs} for restoration paths $\mathcal{P}_d^s, d \in \mathcal{D}, s \in \mathcal{S}\setminus\{\mathcal{O}\}$. Link capacities, both basic and protection to be installed are modeled using variables y'_e and y''_e respectively.

At this point it is worth to mention the *Path Diversity* (PD) problem—a network design problem in which traffic is protected against node or link failures using overprovisioning. The concept behind PD is to route, for each demand $d \in \mathcal{D}$, possibly more traffic than specified by the demand value h_d in the failure-less state, and ensuring that at least a specified fraction of h_d survives any failure state in the considered failure scenario; in effect no flow restoration/rerouting has to be considered. This concept, called also *diversification*, was later extended [13] to develop a new survivability concept called *demand-wise shared protection* (DSP). The DSP was generalized [14] to the model where it is required that at least the amount of flow equal to h_d^s (where $h_d^s \leq h_d, s \in \mathcal{S}$ and $h_d^{\mathcal{O}} = h_d$) should survive in failure state $s \in \mathcal{S}$. The problem formulation is following:

$$\text{minimize} \sum_{e\in\mathcal{E}} \xi_e y_e \tag{2a}$$

$$h_d^s \leq \sum_{p\in\mathcal{P}_d^s} x_p \quad d\in\mathcal{D},\ s\in\mathcal{S} \tag{2b}$$

$$\sum_{p\in\mathcal{P}_e} x_p \leq y_e \quad e\in\mathcal{E}. \tag{2c}$$

As shown in Section 3 the pricing model for the above problem and, hence, the used algorithms, are essentially the same as for the FD problem, in spite of different model origin.

2.3 State-Independent Restoration – FI

Assume that restoration of a failed flow is state-independent, i.e., that restoration of a failed primary flow is performed in the same fashion in every failure state in which the restoration of this particular primary flow is called for. Now the corresponding primal problem FI reads as follows:

$$\text{minimize } F(\boldsymbol{y}) = \sum_{e\in\mathcal{E}} \xi_e(y'_e + y''_e) \tag{3a}$$

$$\sum_{p\in\mathcal{P}_d} x_p \geq h_d \quad d \in \mathcal{D} \tag{3b}$$

$$\sum_{p\in\mathcal{P}_e} x_p \leq y'_e \quad e \in \mathcal{E} \tag{3c}$$

$$x_p \leq \sum_{q\in\mathcal{Q}_p} z_{pq} \quad p \in \mathcal{P} \tag{3d}$$

$$\sum_{d\in\mathcal{D}} \sum_{p\in\bar{\mathcal{P}}_d^s} \sum_{q\in\mathcal{Q}_{pe}} z_{pq} \leq y''_e \quad e \in \mathcal{E},\ s \in \mathcal{S}_e \setminus \{\mathcal{O}\} \tag{3e}$$

$$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \geq \mathbf{0}, \tag{3f}$$

3 Solution Approach

The problems considered in this paper are formulated as non-compact LP models with the number of path-flow variables growing exponentially with the problem size. Such problems are treated by column (path) generation [15,16], as this does not require all the columns of the constraint matrix to be held in the computer memory (which is impossible for large problem instances). Instead, we only keep a subset of the variables (columns) and this can be seen as an approximation (restriction) of the original problem. The algorithm iteratively modifies this subset by introducing new variables in a way that improves the current optimal solution. At the end, the set contains all the variables (paths) necessary to construct the overall optimal solution which can use all possible paths in graph $\mathcal{G}$. Certainly, to make the calculation efficient we have to be able to effectively generate the paths required during the optimization. At each iteration we are interested in generating paths for which the corresponding dual constraint is most violated, as we can expect that this will improve the current optimal solution to the greatest possible extent. As shown in the next paragraphs, finding such a path corresponds to finding the shortest path in a graph according to specific criteria and is called *path generation*, the *pricing problem*, or *dual separation*.

3.1 Pricing Models

Pricing for FD Models. Let $\lambda_d, \pi_e^s, \sigma_d^s$ be the dual variables corresponding to constraints (1b), (1d) and (1e), respectively (note that the dual variable corresponding to constraint (1c) equals ξ_e in the optimal solution). Then, the dual problem to (1) reads as follows:

$$\text{maximize } W(\boldsymbol{\lambda}) = \sum_{d\in\mathcal{D}} h_d \lambda_d \tag{4a}$$

$$\sum_{s\in\mathcal{S}_e\setminus\{\mathcal{O}\}} \pi_e^s = \xi_e \quad e \in \mathcal{E} \tag{4b}$$

$$\sigma_d^s \leq \sum_{e\in q} \pi_e^s \quad d \in \mathcal{D},\ s \in \mathcal{S} \setminus \{\mathcal{O}\},\ q \in \mathcal{P}_d^s \tag{4c}$$

$$\lambda_d \leq \sum_{e\in p} \xi_e + \sum_{s\in\bar{\mathcal{S}}_p} \sigma_d^s \quad d \in \mathcal{D},\ p \in \mathcal{P}_d \tag{4d}$$

$$\boldsymbol{\lambda}, \boldsymbol{\pi}, \boldsymbol{\sigma} \geq \mathbf{0}. \tag{4e}$$

It follows from formulation (4) that if for the current optimal dual variables $\boldsymbol{\sigma}^*$ and $\boldsymbol{\pi}^*$ we are able to find, for some $d \in \mathcal{D}$ and a failure state $s \in \mathcal{S} \setminus \{\mathcal{O}\}$, a protection path $q_d^s \notin \mathcal{P}_d^s$ from node u_d to node v_d with

$$\sum_{e \in q_d^s} {\pi^*}_e^s < (\sigma_d^s)^* \tag{5}$$

then adding this path to the set $\mathcal{P}_d^s$ can potentially decrease the optimal objective (1a). Note that it is easy to generate a shortest protection paths—we just need to use state-dependent link weights equal to π_e^s, $e \in \mathcal{E}$ in the shortest path algorithm. Let q_d^s denote a shortest path for demand $d \in \mathcal{D}$ in situation $s \in \mathcal{S} \setminus \{\mathcal{O}\}$, and its length by r_d^s. Clearly, $r_d^s \leq (\sigma_d^s)^*$. Further, when considering the problem of generating a shortest primary path for demand $d \in \mathcal{D}$ we need to find a path p^* such that the following condition holds

$$\sum_{e \in p^*} \xi_e + \sum_{s \in \bar{\mathcal{S}}_{p^*}} r_d^s < \lambda_d^*. \tag{6}$$

Note that in inequality (6), r_d^s can be substituted with $(\sigma_d^s)^*$ and the procedure will still be valid. As noted above, the pricing problem associated with backup paths is solvable in polynomial time as it can be solved using shortest path algorithm. This is not always the case when the pricing of primary paths is concerned. We comment on the related issues below.

It has been shown in [17,9] (see also [10]) that the path generation problem is solvable in polynomial time in the case of single link (or node) failures. To see this assume only single link failures, i.e., $\mathcal{S} \subseteq \{\{e\} \mid e \in \mathcal{E}\} \cup \{\mathcal{O}\}$, and observe that for demand $d \in \mathcal{D}$ the condition (6) can be rewritten as follows:

$$\sum_{e \in p} (\xi_e + r_d^{\{e\}}) < \lambda_d^*. \tag{7}$$

The right-hand side depends only on the demand, and the left-hand side is nonnegative. Hence, for each demand $d \in \mathcal{D}$, violation of dual constraint (4d) can be tested by searching for a shortest path between the end-nodes of d with respect to the demand-dependent link weights $\gamma_d^e := \xi_e + {r^*}_d^{\{e\}}$ using for example the Dijkstra algorithm, and comparing its length to the value of λ_d^* (if link e does not fail at all we assume $r_d^{\{e\}} = 0$). If the shortest path fulfills condition (7), then the corresponding path variable can be added to the dual formulation (4) (and the corresponding flow variable to the primal formulation (1)) to reduce the domain of the dual problem; otherwise we know that no path for this demand violates its dual constraint for the current set of optimal dual variables.

A general case of a failure state admits multiple link (node) failures; the set of links that fail simultaneously is sometimes called "shared risk link group" (SRLG), and the set of links and nodes that fail simultaneously—"shared risk resource group" (SRRG), see [18]. For this case the path generation problem for FD is difficult.

As for the PD problem, the corresponding dual problem augmented with an auxiliary dual variable Λ_d, $d \in \mathcal{D}$ reads as follows:

$$\text{minimize } W(\boldsymbol{\lambda}) = \sum_{d\in\mathcal{D}} \sum_{s\in\mathcal{S}} h_d^s \lambda_d^s \tag{8a}$$

$$\Lambda_d = \sum_{s\in\mathcal{S}} \lambda_d^s \quad d \in \mathcal{D} \tag{8b}$$

$$\Lambda_d \leq \sum_{e\in p} \xi_e + \sum_{s\in\bar{\mathcal{S}}_p} \lambda_d^s \quad d \in \mathcal{D},\ p \in \mathcal{P}_d, \tag{8c}$$

where λ_d^s is dual variable corresponding to constraint (2b). It follows from formulation (8) that if for current optimal dual variables $\boldsymbol{\lambda}^*$ and $\boldsymbol{\Lambda}^*$ we are able to find for some $d \in \mathcal{D}$ a path $p \notin \mathcal{P}_d$ from node u_d to node v_d with

$$\sum_{e\in p} \xi_e + \sum_{s\in\bar{\mathcal{S}}_p} (\lambda_d^s)^* < \Lambda_d^* \tag{9}$$

then adding this path to set $\mathcal{P}_d$ can decrease the optimal objective (2a). Note that condition (9) has the form equivalent to (6), and in effect the two pricing problems can be considered as essentially the same.

Pricing for FI Models. Let λ_d, π_e^s be the dual variables corresponding to constraints (3b) and (3e), respectively (again, the dual variable corresponding to constraint (3c) equals ξ_e in the optimal solution). Then, the dual problem to (3) reads:

$$\text{maximize } W(\boldsymbol{\lambda}) = \sum_{d\in\mathcal{D}} h_d \lambda_d \tag{10a}$$

$$\sum_{s\in\mathcal{S}_e\setminus\{\mathcal{O}\}} \pi_e^s = \xi_e \quad e \in \mathcal{E} \tag{10b}$$

$$\lambda_d \leq \sum_{e\in p} \xi_e + \sum_{e\in q} \Big(\sum_{s\in\bar{\mathcal{S}}_p} \pi_e^s \Big) \quad d \in \mathcal{D},\ p \in \mathcal{P}_d,\ q \in \mathcal{Q}_p \tag{10c}$$

$$\boldsymbol{\lambda}, \boldsymbol{\pi} \geq \mathbf{0}. \tag{10d}$$

Observe that in this case the pricing problem is always difficult, i.e., also in single link failure scenarios (see [8]). Path generation consists, for each $d \in \mathcal{D}$, in finding a pair of paths $(p, q), p \in \mathcal{P}_d, q \in \mathcal{Q}_p$ with the smallest length defined by the right-hand side of (10c). It follows that the link metrics for calculating the length of the basic path $p \in \mathcal{P}_d$ are equal to the true unit link costs ξ_e, while the link metrics for calculating the length of the backup path $q \in \mathcal{Q}_p$ are given by $\sum_{s\in\bar{\mathcal{S}}_p} \pi_e^s$.

The pricing problem for FI consists in finding, for each demand $d \in \mathcal{D}$, a pair of failure-disjoint paths $r = (p, q)$ from u_d to v_d minimizing

$$\langle r \rangle = \sum_{e\in p} \xi_e + \sum_{e\in q} \Big(\sum_{s\in\bar{\mathcal{S}}_p} \pi_e^s \Big). \tag{11}$$

3.2 Pricing Algorithms

In the following section we analyze three algorithms for the path generation: two exact (MIP and label-setting), and one approximate (primary path precomputation).

MIP Models. For FD problem with multiple link failures, the pricing problem (which is: for each demand $d \in \mathcal{D}$ find a path p from u_d to v_d minimizing $\sum_{e\in p}\xi_e + \sum_{s\in\bar{\mathcal{S}}_p} r^{*\{s\}}_d$) can be formulated as the following mixed-binary problem in binary variables $\boldsymbol{x} = (x_e : e \in \mathcal{E})$ and continuous variables $\boldsymbol{Y} = (Y^s : s \in \mathcal{S})$:

$$\text{minimize } F(\boldsymbol{x}, \boldsymbol{Y}) = \sum_{e\in\mathcal{E}} \xi_e x_e + \sum_{s\in\mathcal{S}} r^s_d Y^s \tag{12a}$$

$$\sum_{e\in\delta^+(v)} x_e - \sum_{e\in\delta^-(v)} x_e = 0 \quad v \in \mathcal{V}\setminus\{u_d, v_d\} \tag{12b}$$

$$\sum_{e\in\delta^+(u_d)} x_e - \sum_{e\in\delta^-(u_d)} x_e = 1 \tag{12c}$$

$$Y^s \geq x_e \quad s \in \mathcal{S},\ e \in \bar{\mathcal{E}}^s \tag{12d}$$

For the FI problem, the pricing problem (minimizing (11)) can be formulated as a mixed-binary problem in binary variables $\boldsymbol{x}' = (x'_e : e \in \mathcal{E})$ and $\boldsymbol{x}'' = (x''_e : e \in \mathcal{E})$, and continuous variables $\boldsymbol{Y} = (Y^s : s \in \mathcal{S})$ and $\bar{\boldsymbol{Z}} = (\bar{Z}^s_e : s \in \mathcal{S}, e \in \mathcal{E})$:

$$\text{minimize } F(\boldsymbol{x}, \bar{\boldsymbol{Z}}) = \sum_{e\in\mathcal{E}} \xi_e x'_e + \sum_{e\in\mathcal{E}}\sum_{s\in\mathcal{S}} \pi^s_e \bar{Z}^s_e \tag{13a}$$

$$\sum_{e\in\delta^+(v)} x'_e - \sum_{e\in\delta^-(v)} x'_e = 0 \quad v \in \mathcal{V}\setminus\{u_d, v_d\} \tag{13b}$$

$$\sum_{e\in\delta^+(u_d)} x'_e - \sum_{e\in\delta^-(u_d)} x'_e = 1 \tag{13c}$$

$$\sum_{e\in\delta^+(v)} x''_e - \sum_{e\in\delta^-(v)} x''_e = 0 \quad v \in \mathcal{V}\setminus\{u_d, v_d\} \tag{13d}$$

$$\sum_{e\in\delta^+(u_d)} x''_e - \sum_{e\in\delta^-(u_d)} x''_e = 1 \tag{13e}$$

$$Y^s \geq x'_e \quad s \in \mathcal{S},\ e \in \bar{\mathcal{E}}^s \tag{13f}$$

$$\bar{Z}^s_e \geq 0,\ \bar{Z}^s_e \geq x''_e + Y^s - 1 \quad s \in \mathcal{S},\ e \in \mathcal{E}. \tag{13g}$$

In the above formulation it is important to assume that $\pi^s_e = +\infty$ for $e \in \mathcal{E}^s$.

Label-Setting. The pricing problems for FD and FI can be approached in a way described in [8]. The idea is as follows. Consider a demand $d \in \mathcal{D}$ and a path $p \in \mathcal{P}_d$ between nodes u_d and v_d. The length of path p with respect to weights ξ_e ($e \in \mathcal{E}$) will be denoted by $\|p\| := \sum_{e\in p} \xi_e$. For fixed dual variables, the notation $\langle p\rangle$ will stand for the dual length of path p, and in case of FD, can be expressed as follows (9):

$$\langle p\rangle = \sum_{e\in p} \xi_e + \sum_{s\in\bar{\mathcal{S}}_p} r^s_d \tag{14}$$

and in case of FI:

$$\langle p \rangle = \sum_{e \in p} \xi_e + \sum_{e \in q} \Big(\sum_{s \in \bar{\mathcal{S}}_p} (\pi_e^s)^* \Big). \tag{15}$$

We say that path $p \in \mathcal{P}_d$ *dominates* another path $p' \in \mathcal{P}_d$ if $\|p\| \leq \|p'\|$ and $\bar{\mathcal{S}}_p \subseteq \bar{\mathcal{S}}_p$, i.e., when path p is not longer than path p' and when the set of failure states that affect path p is a subset (not necessarily proper) of the set of failure states affecting path p'. It is obvious that if path p dominates path p' then $\langle p \rangle \leq \langle p' \rangle$. Consequently, in order to generate a new path p for demand d (i.e., a path with minimum value of $\langle p \rangle$ with respect to the current optimal dual variables) it is sufficient to generate the set $\mathcal{U}(d)$ of minimal paths with respect to the relation of domination defined above. That is, no path from $\mathcal{U}(d)$ dominates another path from this set.

Such a set $\mathcal{U}(d)$ of all non-dominated paths can be generated by means of a label-setting algorithm for *shortest-path problems with resource constraints* (SPPRC) [12]. This can be done in pseudo-polynomial time, i.e., in time which is polynomial with respect to the size of the network graph, and exponential with respect to the number of failure states $|\mathcal{S}|$ ("resources" are the failure states in this case).

The label-setting algorithm works by creating a set of paths originating at node u_d. Each path is represented by a label placed in its last node. The label contains information about the already visited nodes, as well as the utilized resources. The path can be extended by a link incident to its last node, but only if it does not result in a cycle or in a violation of the resource constraints. In the worst case the number of different paths between the nodes u_d and v_d can grow exponentially with the number of resources. In many cases, however, we are able to apply some dominance rules to decrease the number of the resulting paths. This can be done not only in the target node v_d but also in the intermediate nodes. It is easy to show that if path p' is dominated with respect to the rules defined above, then any extension of p' would be finally dominated in v_d as well.

As an extension of the SPPRC algorithm applied in [8], we have also introduced path length limitation—an important contribution to the reduction of the size of set $\mathcal{U}(d)$. The extension is based on the observation that excessively long paths are useless as they cannot improve the current solution, or the solutions they represent are known to be worse than some already known solutions. For example, a simple path length restriction for FD can be constructed from (6): $\sum_{e \in p} \xi_e < \lambda_d^*$. Also, knowledge of some path p' representing a feasible solution can help to tighten the path length restriction. In such a case, for FD we are only interested in finding a path p satisfying $\sum_{e \in p} \xi_e < \sum_{e \in p'} \xi_e + \sum_{s \in \bar{\mathcal{S}}_{p'}} r_d^s$. Similar constraints can be derived for problem FI as well. Applying path length limitation to pricing in problem FI results in significant (up to 3 orders of magnitude in our experiments) reduction of the pricing time.

Primary Path Precomputation. Usefulness of the pricing algorithms presented before may sometimes be limited if applied for large network instances. Therefore, we consider a heuristic solution based on large predefined sets of allowable primary paths that are kept aside from the current problem formulation (using smaller sets of candidate primary paths $\mathcal{P}_d$). Having sets $\mathcal{R}_d$, $d \in \mathcal{D}$ of the allowable primary paths defined in advance, we can reduce the pricing problem to simple enumeration of the primary paths contained in these (large) sets.

Table 1. Network characteristics

	Network					
	Pdh	Newyork	Ta1	France	Norway	Cost266
Nodes	11	16	24	25	27	37
Links	34	49	51	45	51	57
Demands	24	120	163	300	351	666

For problem FD, we simply compute the lengths of the paths in each $\mathcal{R}_d$ according to the left hand side of formula (6), where r_d^s are the appropriate shortest path lengths computed previously for the current optimal dual variables. For each $d \in \mathcal{D}$ the shortest path in set $\mathcal{R}_d$ should be added to the problem (i.e., to set $\mathcal{P}_d$) if its length is strictly less than λ_d^*.

For problem FI, we enumerate, for each $d \in \mathcal{D}$, the allowable primary paths p in the set $\mathcal{R}_d$, and for each of them we determine the shortest backup path q computed as a shortest path with respect to link weights π_e^s, $s \in \bar{\mathcal{S}}_p$, i.e., for the states in which the considered primary path p fails. Obviously, the resulting backup paths should be state-disjoined with respect to the protected primary paths. We assure this property simply by removing links belonging to the protected primary path p. As a result, a pair $r = (p, q)$ for each $p \in \mathcal{R}_d$ is formed. Finally, a pair with the smallest total length defined by the right-hand side of (10c) is found, and added to the formulation of problem (3), as long as its length is strictly smaller than the value of λ_d^*.

As shown by numerical experiments presented in Section 4, this approach seems to be reasonable, often allowing for the globally optimal solution of the considered problem. Still, the effectiveness of the proposed simplified method heavily depends on the construction of the sets of allowable primary paths $\mathcal{R}_d$. Sets $\mathcal{R}_d$ should contain only "good" candidate primary paths and should be kept as small as possible. Having routing lists that contain "good" paths (sufficient for resolving the problem in hand) the resolution process would be simplified a lot. It seems that a reasonable approach for arriving at lists of paths which have a good chance to be sufficient for solving the problem can be based on the computation of K-shortest paths with respect to unit link costs ξ_e, $e \in \mathcal{E}$. Hence, using these costs as link weights, we can construct the lists of paths that are most likely to be used in the globally optimal solution. This follows from the observation that in the simplified LP dimensioning problems without failure states, an optimal solutions consists in realizing the demands along the shortest paths computed with respect to unit link costs ξ_e, $e \in \mathcal{E}$. Algorithms for finding K-shortest path are well known and can be found in [19,20,21,22,23,24].

4 Computational Experiments

In our computational investigations we conducted a series of performance tests of the discussed methods. For this purpose we selected a set of six network instances (cf. Table (1)). Network topologies and demand volumes were taken from SNDLib library [25]. The unit link costs ξ_e, $e \in \mathcal{E}$ were generated randomly since not all the network

Table 2. FD – double link failures

Network	Path generation	Iterations	Columns	Time [s]		
				master	pricing	total
Pdh	MIP	53	1253	1	1	2
	Label-setting	63	1057	1	0	1
	Precomputed	52	1226	1	0	1
Newyork	MIP	88	10101	61	15	76
	Label-setting	78	7785	49	3	52
	Precomputed	82	9965	56	0	56
Ta1	MIP	69	13964	39	19	58
	Label-setting	62	12427	36	4	40
	Precomputed	60	13124	32	3	29
France	MIP	34	12583	32	28	60
	Label-setting	32	10762	29	4	33
	Precomputed	32	12531	36	0	36
Norway	MIP	39	29277	323	43	366
	Label-setting	34	21741	301	53	354
	Precomputed	40	28363	329	1	330
Cost266	MIP	33	44752	394	85	479
	Label-setting	34	34714	350	104	454
	Precomputed	34	44850	424	4	428

instances contained in SNDlib contained this information. Also, we assumed the presence of a demand requirement between each pair of nodes. Regarding failure states, we tested the two following scenarios: all single link failures, and all double link failures (for FD only). In the second case the number of failure states was equal to the number of states in the single link failure scenario, i.e., equal to the number of links (plus 1 for the normal state). The second scenario was constructed by randomly choosing the second failing links in the considered state and marking it to be failed. Clearly, this operation should be performed carefully as the network graph must be at least two-connected in all failure states.

In our numerical experiments we considered the three following computational scenarios: state-dependant restoration (FD) with single link failures, state-dependant restoration (FD) with double link failures, and state-independent restoration (FI) for single link failures. Initial problems (master problems) were formulated using L-P notation, and solved within our own path generation (PG) framework. Optimal values of dual multipliers were obtained using the LP solver of CPLEX 9.1. In order to resolve the pricing problems, we used all the three presented algorithms: exact MIP pricing, exact pricing based on the shortest path computations using label-setting algorithm, and the heuristic for precomputing the sets of allowable primary paths. (In the primary path precomputation we decided to pre-generate 5 primary shortest paths for each demand computed, with respect to link weights $\xi_e,\ e \in \mathcal{E}$.

In the case of problem FD with single link failures, the pricing problem was solved using a more general version of label-setting algorithm designed for the multiple failures case. Moreover, in both failure scenarios for FD, at each iteration of the PG

Table 3. FD – single link failures

Network	Path generation	Iterations	Columns	Time [s]		
				master	pricing	total
Pdh	MIP	162	1212	1	4	5
	Label-setting	115	1032	1	0	1
	Precomputed	149	1246	1	0	1
Newyork	MIP	87	7706	34	17	51
	Label-setting	74	6144	33	2	35
	Precomputed	75	7612	42	0	42
Ta1	MIP	112	10166	21	30	51
	Label-setting	128	9017	23	6	29
	Precomputed	133	9810	24	1	25
France	MIP	38	8483	23	17	40
	Label-setting	41	7687	20	3	23
	Precomputed	36	8469	22	0	22
Norway	MIP	71	19766	190	33	223
	Label-setting	84	19079	244	17	261
	Precomputed	53	19193	174	2	176
Cost266	MIP	36	31122	278	48	326
	Label-setting	35	28398	334	30	364
	Precomputed	34	31597	266	3	269

Table 4. FI – single link failures

Network	Path generation	Iterations	Columns	Time [s]		
				master	pricing	total
Pdh	MIP	121	682	8	112	120
	Label-setting	124	822	1	0	1
	Precomputed	157	814	1	1	2
Newyork	MIP	62	2887	152	660	812
	Label-setting	57	2867	25	1	26
	Precomputed	66	3079	126	1	127
Ta1	MIP	74	3385	122	1455	1577
	Label-setting	101	3913	22	2	24
	Precomputed	122	4484	95	10	105
France	MIP	14	1957	11	297	308
	Label-setting	16	2294	5	2	7
	Precomputed	18	2696	22	1	23
Norway	MIP	27	4453	420	2044	2464
	Label-setting	29	4419	105	7	112
	Precomputed	30	4665	323	5	328
Cost266	MIP	9	4012	67	900	967
	Label-setting	12	5465	52	5	57
	Precomputed	13	5569	146	2	148

algorithm at most one primary path and at most one backup path for each state and for each demand were generated. In problem FI, at each iteration at most one primary-backup path pair for each demand was generated. All the experiments were performed on a PC computer with a Pentium 4 (2.8GHz) processor.

Tables 2, 3 and 4 present results for the considered network instances, while Table 5 contains objective function values for the three pricing algorithms. First of all, we notice that the pricing problem (path generation) does not seem to affect considerably the overall performance, as the overall time spent on solving the master problems has much bigger impact on the total computation time. When comparing directly the pricing problem solution times, a very good performance of the primary path precomputation method in all the tested cases is observed. What is more important, it gives either optimal solutions or excellent approximations. The label-setting algorithm shows its strength for the FI problem, where it performs similarly to the heuristic algorithm, at the same time assuring optimal solutions. The approach utilizing MIPs for pricing is quite efficient for the FD problem but fails for the FI problems.

Table 5. Objective values

FD – double link failures						
Path gen.	Pdh	Newyork	Ta1	France	Norway	Cost266
MIP	62805.2	16508.6	$8.78397e7$	$4.15572e6$	120837	$1.78992e7$
Label-setting	62805.2	16508.6	$8.78397e7$	$4.15572e6$	120837	$1.78992e7$
Precomputed	62805.2	16508.6	$8.78534e7$	$4.15584e6$	120837	$1.79049e7$
FD – single link failures						
Path gen.	Pdh	Newyork	Ta1	France	Norway	Cost266
MIP	57564.3	15692.4	$8.15358e7$	$3.67687e6$	108303	$1.56683e7$
Label-setting	57564.3	15692.4	$8.15358e7$	$3.67687e6$	108303	$1.56683e7$
Precomputed	57564.3	15692.4	$8.15388e7$	$3.67687e6$	108305	$1.56683e7$
FI – single link failures						
Path gen.	Pdh	Newyork	Ta1	France	Norway	Cost266
MIP	57564.3	15692.4	$8.15494e7$	$3.67687e6$	108313	$1.56686e7$
Label-setting	57564.3	15692.4	$8.15494e7$	$3.67687e6$	108313	$1.56686e7$
Precomputed	57564.3	15692.4	$8.15545e7$	$3.67687e6$	108317	$1.56686e7$

5 Conclusions

The paper discusses the problem of link dimensioning and flow optimization in resilient networks with two different flow restoration schemes to be used in case of network failures, namely state-dependant restoration and state-independent restoration. The two resulting problems are specified in the L-P formulation.

Three appropriate path generation methods, necessary to make the L-P formulation really useful, are discussed: exact approach based on Mixed Integer Programming, an adaptation of the label-setting algorithm, and a heuristic algorithm making use of large precomputed lists of primary paths. As shown by the numerical experiments (Section 4), together the presented algorithms provide an efficient means for paths generation for the

considered types of problems. In fact, only application of the MIP pricing for problem FI appears inefficient. (This is probably caused by the complexity of problem (13) incorporating large number of variables and constraints.)

We emphasize that in practice the simplified pricing based on the precomputation of K-shortest primary paths is very efficient and sufficient. In most cases, the values of the optimal objective function are correct. Still, this approach, as compared to the exact label-setting algorithm, does not provide a significant performance gain. This is due to the impact of the solution time of the master problem which is much longer when compared to the time needed to generate new paths. Our analysis shows that the presented path generation algorithms enable efficient resolution of the two considered resilient network design problems for networks of realistic size.

Acknowledgment

The authors are indebted to Dritan Nace, Sebastian Orlowski, and Thomas Stidsen for valuable and enlightening discussions on the paper.

References

1. Pióro, M., Medhi, D.: Routing, Flow, and Capacity Design in Communication and Computer Networks. Morgan Kaufman, San Francisco (2004)
2. Minoux, M., Serrault, J.Y.: Subgradient optimization and large scale programming: an application to optimum multicommodity network synthesis with security constraints. R.A.I.R.O. Operations Research 15(2) (1981)
3. Dahl, G., Stoer, M.: A cutting plane algorithm for multicommodity survivable network design problems. INFORMS Journal on Computing 10, 1–11 (1998)
4. Wessäly, R.: Dimensioning Survivable Capacitated NETworks. PhD thesis, Technische Universität Berlin (2000)
5. Hu, J.: Diverse routing in optical mesh networks. IEEE Trans. Com. 51(3), 489–494 (2003)
6. Maurras, J.F., Vanier, S.: Network synthesis under survivability constraints. 4OR (2), 52–67 (2004)
7. Coudert, D., Datta, P., Perennes, S., Rivano, H., Voge, M.E.: Complexity and approximability issues of shared risk resource group. Technical report, Technical report 5859, INRIA (2006)
8. Stidsen, T., Petersen, B., Rasmussen, K., Spoorendonk, S., Zachariasen, M., Rambach, F., Kiese, M.: Optimal routing with single backup path protection. In: International Network Optimization Conference INOC 2007, Spa, Belgium (2007)
9. Bashllari, A., Nace, D., Rourdin, E., Klopfenstein, O.: The MMF rerouting computation problem. In: International Network Optimization Conference INOC 2007, Spa, Belgium (2007)
10. Orlowski, S., Pióro, M.: On the complexity of column geneartion in survivable network design. Technical report, Zuse Institut Berlin and Warsaw University of Technology (2007)
11. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Englewood Cliffs (1993)
12. Irnich, S., Desaulniers, G.: Shortest path problems with resource constraints. In: Desaulniers, G., Desrosier, J., Solomon, M. (eds.) Column Generation, pp. 33–65. Springer, Heidelberg (2005)

13. Koster, A., Zymolka, A., Jäger, M., Hülsermann, R.: Demand-wise shared protection for meshed optical networks. Journal of Network and Systems Management 13(1), 35–55 (2005)
14. Wessäly, R., Orlowski, S., Zymolka, A., Koster, A., Gruber, C.: Demand-wise shared protection revisited: A new model for survivable network design. In: Proc. INOC 2005, Lisbon, pp. 100–105 (2005)
15. Dantzig, G.B., Wolfe, P.: The decomposition algorithm for linear programming. Operations Research 8, 101–111 (1960)
16. Desrosiers, J., Luebbecke, M.: A primer in column generation. In: Desaulniers, G., Desrosier, J., Solomon, M. (eds.) column generation, pp. 1–32. Springer, Heidelberg (2005)
17. Orlowski, S.: Local and global restoration of node and link failures in telecommunication networks. M.sc. thesis, Technische Universität Berlin (2003), `http://www.zib.de/orlowski/`
18. Strand, J., Chiu, A.L., Tkach, R.: Issues for routing in the optical layer. IEEE Communications Magazine, 81–87 (2001)
19. Yen, J.Y.: Finding the k shortest loopless paths in a network. Management Science 17, 712–716 (1971)
20. McCallum, C.J.: An algorithm for finding the k shortest paths in a network. Bell Laboratories Technical Memorandum TM73-1713-9 (1973)
21. Eppstein, D.: Finding the k shortest paths. In: 35^{th} IEEE Symposium on Foundations of Computer Science, pp. 154–165 (1994)
22. Dreyfus, S.E.: An appraisal of some shortest-path algorithms. Operations Research 17, 395–412 (1999)
23. Jiménez, V.M., Marzal, A.: Computing the k shortest paths: A new algorithm and an experimental comparison. In: Proceedings of 3^{rd} Annual Workshop on Algorithmic Engineering, London (1999)
24. Pascoal, M.M.B., Eugénia, M., Captivo, V., Climaco, J.C.N.: An algorithm for ranking quickest simple paths. Computers and Operations Research 32, 509–520 (2005)
25. Orlowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0–Survivable Network Design Library. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium (2007), `http://sndlib.zib.de`