Optimal Routing with Failure-Independent Path Protection

Thomas Stidsen
Informatics and Mathematical Modelling, Technical University of Denmark, Denmark

Bjørn Petersen, Simon Spoorendonk, and Martin Zachariasen
Department of Computer Science, University of Copenhagen, Denmark

Kasper Bonne Rasmussen
Department of Computer Science, ETH Zurich, Switzerland

Reliable communication has become crucial in today's information society. Modern communication networks are required to deliver reliable communication to their customers. Unfortunately, protection against network failures significantly hampers efficient utilization of network investments, because the associated routing problems become much harder. In this article we present a rigorous mathematical analysis of one of the most promising protection methods: Failure independent path protection. We present an LP model which is solved by column generation. The subproblem is proven to be strongly $\mathcal{NP}$-hard, but still solvable for medium sized networks through the use of specialized dynamic programming algorithms. This enables us to evaluate the performance of failure independent path protection for eight networks with up to 37 nodes and 57 links. The results indicate that only between 3% and 8% extra network capacity is necessary when compared with the capacity required by complete rerouting (which is the absolute lower bound for single link failure protection). © 2009 Wiley Periodicals, Inc. NETWORKS, Vol. 55(2), 125–137 2010

Keywords: failure independent path protection; column generation; protection capacity minimization

1. INTRODUCTION

Today's information society relies increasingly on advanced communication networks. This has led to massive investments in increased communication network capacity. To utilize these investments the network operators perform traffic engineering, i.e., they route communication to maximize the utilization of the capital invested in the communication network.

Received March 2007; accepted January 2009
Correspondence to: T. Stidsen; e-mail: tks@imm.dtudk
DOI 10.1002/net.20323
Published online 31 July 2009 in Wiley InterScience (www.interscience.wiley.com).
© 2009 Wiley Periodicals, Inc.

Most of the backbone networks which today carry long distance communication traffic use path based routing, i.e., a communication connection between two points in the network is established along one or more fixed paths. Despite the huge success of the packet switched Internet, path-based routed network technology will continue to be the dominant technique of backbone networks, because traffic engineering can be performed much more efficiently than in packet switched networks. Examples of such path switched network technologies are SDH/SONET or DWDM networks or circuit switched network technologies like PSTN/ISDN. Furthermore, the new Multi Path Label Switching (MPLS) [34] protocol enables packets to be routed on fixed paths.

The standard model of a path switched communication network is a directed graph $G = (V, A)$ consisting of a set of nodes V and a set of arcs A. The nodes correspond to telecommunication switches. The telecommunication switches route the communication signals through cables. We will assume that all cables enable bidirectional communication and therefore we will model a cable using two opposite arcs. We assume that a static communication demand is given which requires one-way communication between an origin node o_k and a terminating node d_k of volume ρ_k for a set of demands $k \in K$. For each demand k we should construct a single primary (or working) path from o_k to d_k, and all the required volume of traffic ρ_k should be sent over this primary path (i.e., traffic should be nonbifurcated).

Communication networks are increasingly required to be reliable. If we cannot trust our messages to reach the receiver, the use of a communication network is limited. Communication networks are prone to failures and many different types of failures can occur. Switches (nodes) can lose power, experience software and hardware failures, etc. Cables (arcs) can be cut by entrepreneurs or by natural disasters. For simplicity, in this article we will only consider single cable failures, i.e., simultaneous failure of the two opposite arcs which

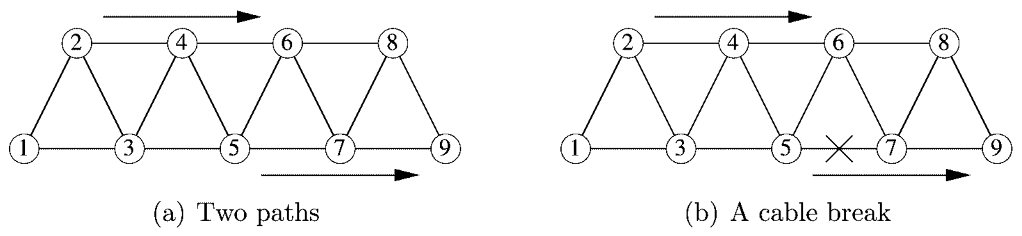


FIG. 1. Path switched routing. (a) Two paths (b) A cable break.

correspond to a cable. This is a well-known and widely used simplification [16,27].

Multiple cable failures can occur in networks, but are less probable. Several cables can fail, if, e.g., a switch fails or a single cable failure in a lower network layer results in multiple failures in the upper layers. These kind of network errors are of increasing importance but they also make network protection significantly harder. For example, the problem of finding failure independent paths is $\mathcal{NP}$-complete in the case of multiple cable failures [18].

When a cable fails, the network operator either has to repair the cable or re-route the failed paths around the failure. Because repairing a cable can take considerable time, rerouting is an interesting alternative. The main problem with rerouting is that enough capacity needs to be available on the remaining nonfailed cables to enable rerouting. Traffic engineering which takes into account the possibility of a cable failure becomes significantly more complex, but is again important in order to utilize network investments.

In this article we assume that traffic which is routed along one primary path is always rerouted along the same backup (or protection) path. Hence rerouted traffic is nonbifurcated. The cost function is simple: We assume that a linear cost term c_a for using capacity on arc a has to be paid. The required capacity of an arc is the maximum capacity required for all failure situations (the network should be able to accommodate necessary rerouting). The total cost of the network is the sum of costs over all arcs. It should be noted that in our model arcs have no capacity bounds—in contrast to the well-known multicommodity flow model [2].

In Figure 1a two paths are established, from node 2 to node 6 and from node 5 to node 9, both with a volume of 1, that is, $(o_1, d_1, \rho_1) = (2, 6, 1)$ and $(o_2, d_2, \rho_2) = (5, 9, 1)$. In Figure 1a—and all the other figures in this article—we have only drawn the bidirectional cables, and not the two corresponding arcs for each cable, in order not to complicate the figures unnecessarily. The necessary capacity of a cable corresponds to the sum of the necessary arc capacities for that cable. Given the paths chosen in Figure 1a an arc capacity of 1 is then required on the arcs $(2, 4)$, $(4, 6)$, $(5, 7)$, and $(7, 9)$, resulting in a total required Non-Failure (NF) network capacity of 4. In Figure 1b the cable between node 5 and node 7 fails resulting in the failure of arc $(5, 7)$ and arc $(7, 5)$. This results in a communication breakdown for the path from node 5 to node 9.

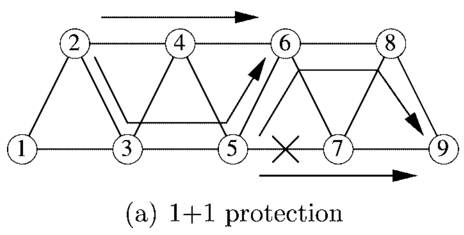
To protect communication against a cable failure, a rerouting strategy needs to be planned for each possible cable failure, i.e., a protection method needs to be installed. (Because rerouting methods protect against failures, we will use rerouting methods and protection methods interchangeably.) The importance of network reliability and the importance of minimizing network investments have resulted in a large number of rerouting methods. It is beyond the scope of this article to review these and we refer the reader to [16] for a recent and comprehensive survey. One of the promising methods is p-cycle protection. This is a clever extension of the well-known ring protection scheme, which significantly improves the capacity requirements necessary for protection [16,32]. Furthermore, the use of p-cycles enables fast protection of communication, as provided by ring protection. Despite these promising features, p-cycles have not (yet) achieved widespread application.

In this article we will consider traffic engineering optimization methods for the Failure Independent Path Protection (FIPP) method for path switched networks. In this protection method the backup path for a given demand is independent of the failure related to the primary path, that is, independent of which of the cables in the primary path have failed. This protection method is also called Shared Backup Path Protection in [16] or Global Backup Path Protection in [7].

The outline of the article is as follows. In Section 2 we give a brief description of various path protection methods. This leads us to focus on the FIPP method for which we give a mathematical model in Section 3. In the same section we also present a column generation algorithm to solve a relaxed model and discuss the computational complexity of the subproblem. In Section 4 we present a labeling algorithm for the subproblem. In Section 5 we then present and discuss the results when applying the column generation algorithm to a number of test cases. In Section 6 we discuss possible extensions and in Section 7 we draw our conclusions.

2. PATH PROTECTION METHODS

The classic path protection method employed in path switched networks is 1+1 protection. Figure 2a shows how the 1+1 protection method can be used to protect the path connections from Figure 1. In 1+1 protection, two cable disjoint paths (and hence arc disjoint paths) are established and actively used. If an arc fails on one path, the other path will survive and enable the receiving node to restore communication by just switching to the other incoming signal. This method is simple, there are well-defined standards, but the required network capacity is always at least twice the required



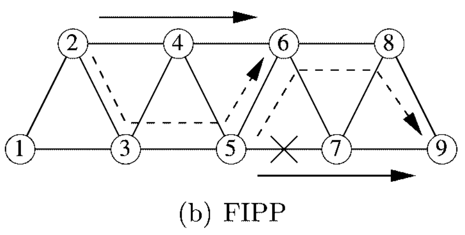


FIG. 2. Capacity sharing illustrated. (a) 1+1 protection (b) FIPP.

non-failure network capacity. The total network capacity required in the example in Figure 2a, assuming the same demands, is 10. Notice in particular that a capacity of 2 is required on arc $(5, 6)$.

2.1. *Comparing Path Protection Methods*

We now define two measures: Restoration Over Build (ROB) network capacity and Relative Restoration Over Build (RROB) network capacity.

ROB: The extra network capacity necessary to ensure protection, i.e., the network capacity for both routing and protection minus the NF network capacity, assuming shortest path routing. In the example from Figure 2a, ROB $= 10 - 4 = 6$.

RROB: The relative extra network capacity necessary to ensure protection, i.e., the ROB network capacity divided by the NF network capacity. In the example from Figure 2a, RROB $= \frac{10-4}{4} = 1.5$, meaning that 1+1 protection in this case costs 150% extra network capacity compared to the necessary non-failure network capacity.

The FIPP method is a slight variation of 1+1 protection: Instead of actively sending data packets on both paths, one path is designated the primary path and only when that path fails will the data packets be sent along the backup path. In Figure 2b the same two protected connections as in Figure 2a are shown, but now there is a primary path (full line) and a backup path (dashed line) for each path. But the required network capacity has decreased. The arc $(5, 6)$ now only needs a capacity of 1, because the backup paths are not being used at the same time. This concept is called sharing and is possible because we only guarantee protection against single cable failures and because the two primary paths are cable disjoint. For the FIPP method, the NF network capacity is again 4, but the ROB network capacity is now 9, which leads to an RROB network capacity of 1.25.

To utilize the path protection methods traffic engineering has to be performed in order to minimize the RROB network capacity. When working with 1+1 protection this is a well-studied problem for which there exist polynomial-time algorithms [5, 33]. This is not the case for the FIPP method. Because of the possibility of sharing the capacity for the backup paths, the best choice of primary path and backup path for each end-to-end demand node pair becomes interdependent.

A practical solution to the FIPP traffic engineering problem is studied in [24]. In order to simplify the problem, the dependency between different protected communication connections is ignored in [24]. Instead, the focus is on algorithms which can find pairs of disjoint paths, where the cost of backup paths is assumed to be some constant factor cheaper than for primary paths. Because of the sharing possibility it is reasonable that the capacity costs for each arc of the backup path are less than the capacity costs for each arc of the primary path. Even this simplified problem is $\mathcal{NP}$-hard [24] and a number of different heuristics are suggested to find good, but not necessarily optimal solutions to the problem. This line of research is continued in [23]. It should be emphasized that the cost model for backup paths used in [23, 24] is approximate. We quantify the exact relationship between costs for primary and backup paths in Section 3.1 and prove that the resulting optimization problem is strongly $\mathcal{NP}$-hard.

In [27] the full FIPP traffic engineering problem is considered. A column generation approach, similar to the approach in our article, is considered. The same mathematical model for the column generation master problem is formulated, but the subproblem is not formulated. This means that if an optimal solution is required, the full set of disjoint paths has to be pregenerated, and this is only feasible for small networks.

2.2. *Different Path Protection Methods*

The (FIPP) method is just one example of a path protection scheme, and there are a number of other methods. The different path protection methods all use one primary path, but protect the primary path in different ways. In Figure 3, which is (partly) taken from [7], six path protection methods are presented. If the path protection methods are only allowed to choose the backup path based on the failed cable, this list is complete (a number of additional variations exists, some of which are described in Section 2.3).

2.2.1. Full Backup Path Protection (FBPP). Theoretically FBPP [25], see Figure 3a, is the most efficient path protection method. (This method is not included in [7].) Given a primary path, each cable which can fail on the primary path is protected by a unique backup path. There are no limitations regarding these backup paths, except they are, obviously, not allowed to use any of the two failed arcs in the cable which they protect. This gives the highest possible freedom in choosing the cheapest protection paths. All the other

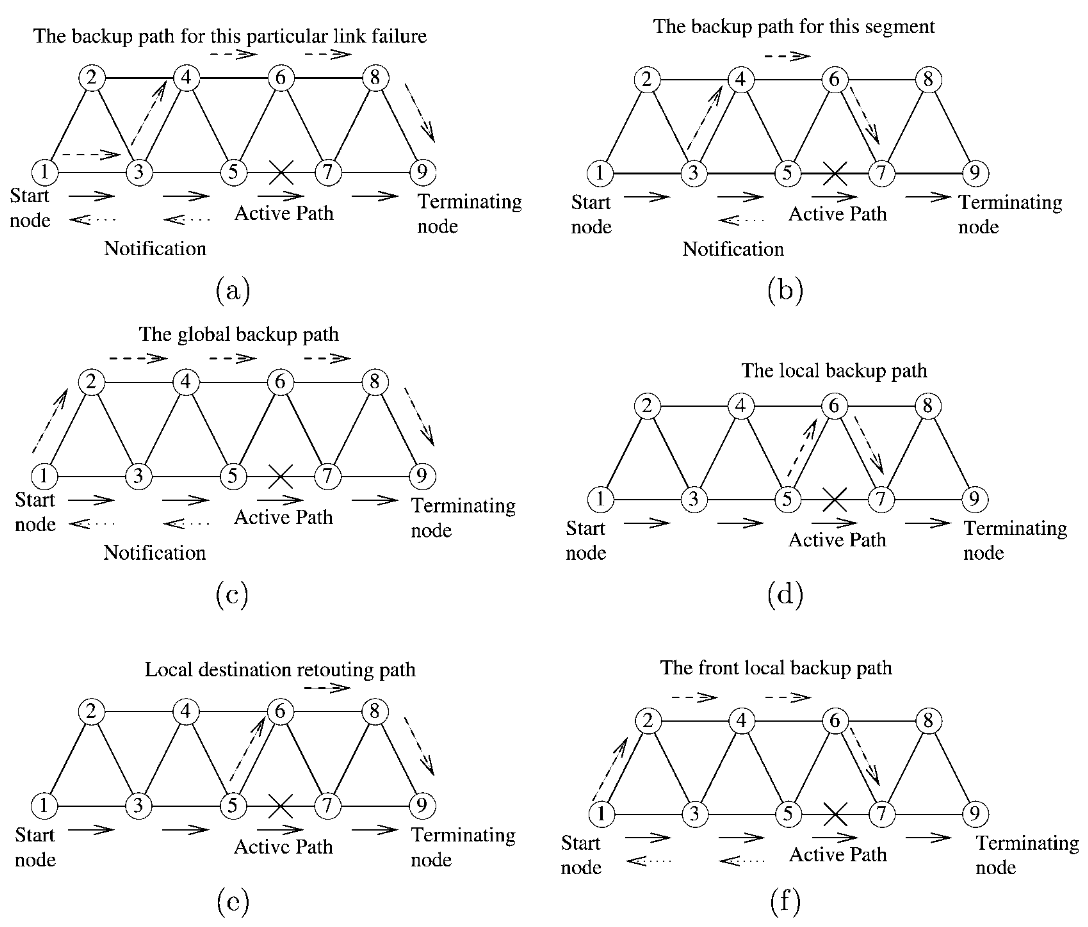


FIG. 3. Different path protection schemes. (a) Full backup path protection. (b) Segment backup path protection. (c) Failure independent path protection. (d) Local backup path protection. (e) Local destination rerouting. (f) Front dynamic backup path protection.

path protection methods are more restrictive in the choice of backup paths and hence more costly.

2.2.2. Segment Backup Path Protection (SEBPP). SEBPP, see Figure 3b, protects segments (sets of cables) of the primary path with the same backup path. Hence several cables in the same segment are forced to share backup paths.

2.2.3. FIPP. FIPP, see Figure 3c, limits the choice of backup path even further, such that only one backup path is allowed. This forces the backup path to be cable disjoint with the primary path.

2.2.4. Local Backup Path Protection (LBPP). LBPP [25], see Figure 3d, performs a local protection, i.e., the rerouting paths are required to lead from one node of the failed cable to the other node of the failed cable. This resembles the classical span protection, but in this case different reroute paths may be chosen for each connection.

2.2.5. Local Destination Rerouting (LDR). LDR [3], see Figure 3e, is a variation of local protection, where the connection paths are rerouted directly to the end node of the connection. LDR preserves the fast rerouting time of LBPP, but is more efficient regarding ROB network capacity.

2.2.6. Front Dynamic Backup Path Protection (FDBPP). FDBPP, see Figure 3f, is another variation of local protection, where the connection path is rerouted from the start node to the end node of the failed cable. To the best of our knowledge this type of protection has not been suggested anywhere else and is only included to make the list of path protection methods complete. We do not expect the FDBPP method to be implemented anywhere.

2.3. *Further Variations*

The description of the different path protection schemes is very simplified and a number of variations can be added. Here we briefly mention two of these.

Stub-release is a technique which can be applied to further lower the required network capacity. The idea is that in case of a failed cable, the unharmed parts of the primary path, which are no longer in use, are released and can be used for protection [26]. Stub-release can improve the capacity efficiency of all methods, with the exception of the

LBPP method, at the price of a more complicated protection scheme.

To speed up the recovery process, Haskin [17] proposed a protection scheme. The idea is to loop-back the communication signals at the switch just before the failed cable, to where the backup path starts. Haskin protection minimizes packet-loss, but requires more network capacity and cannot be used in LBPP or LDBPP.

2.4. Motivation for FIPP

Among the six different types of path protection methods described in Section 2.2, we only consider the FIPP method in this article.

FIPP is the only path protection method for which the protection action does not depend on which cable actually fails—it is failure independent. This makes FIPP the simplest of the path protection methods. Furthermore, the complex switching schemes take place at the start node of the connection path, which may be an advantage in future networks. It is not the most capacity efficient path protection method. The most efficient method is FBPP, but FBPP requires administration of a large number of backup paths. Furthermore, in Section 5 we demonstrate that the FIPP method is indeed a very efficient protection method when optimal routing is performed.

The main disadvantage with the FIPP method is the relatively long restoration time, i.e., the time it takes to restore communication. This is because of the notification time—which is the backward communication time between the node which observes the failure and the node from which the connection paths originates. We have illustrated the notification time by dotted arrows in Figure 3 for the path protection methods for which this is necessary. For a more complete discussion of restoration time, we refer to [7].

3. LP MODEL AND COLUMN GENERATION APPROACH

In this section we start by defining the FIPP optimization problem formally. Then we present an LP model for a relaxed version of the FIPP optimization problem, the so-called fractional FIPP optimization problem. The LP model has an exponential number of variables, and hence we solve it using column generation. In Section 3.1 we describe the associated pricing problem (or subproblem). A MIP model for solving the subproblem is given in Section 3.2, and in Section 3.3 we show that the subproblem is in fact strongly $\mathcal{NP}$-hard. Finally, in Section 3.4 we summarize our column generation algorithm.

Given is a directed graph $G = (V, A)$ with nodes V and arcs A. For each failure situation $s \in S$ we have a set of failed arcs $F_s \subseteq A$. There is a cost c_a for using one unit of capacity of an arc a. We further assume to know a static set of demand node pairs for which protected connections using the FIPP method should be established. A directed connection between an origin node o_k and a terminating node d_k with a volume of ρ_k should be established for each demand $k \in K$. The optimization objective is to minimize the cost of the required capacity when applying the FIPP method to protect the established connections. This means that for each demand a pair of directed failure disjoint paths needs to be found: A primary path p^{pri} and a backup path p^{bac}, both connecting node o_k to node d_k. Such a pair of failure disjoint paths is denoted a path pair $\pi = (p^{\text{pri}}, p^{\text{bac}})$. The objective in the FIPP problem is to find a path pair for each demand $k \in K$, such that the total cost of the capacity required is minimized. Note that the capacity required by an arc is the maximum capacity required taken over all failure situations.

Given these definitions we are ready to present an LP model for the fractional FIPP optimization problem. In this problem we allow more than one path pair to accommodate the flow required by a demand. Let P_k be the set of path pairs that can satisfy demand k, that is, the set of primary/backup paths that connect origin node o_k with terminating node d_k. Let $P_k(a) \subseteq P_k$ be the subset of path pairs for which the primary path uses arc $a \in A$. Similarly, let $P_k(a,s) \subseteq P_k$ be the set of path pairs for which the primary path fails and the backup path uses arc $a \in A \setminus F_s$ in failure situation $s \in S$. Finally, let variable λ_π^k denote the amount of communication flow through path pair $\pi \in P_k$, and let variable θ_a denote the capacity required for arc $a \in A$.

FIPP

minimize:

$$\sum_{a \in A} c_a \cdot \theta_a \tag{1}$$

subject to:

$$\sum_{\pi \in P_k} \lambda_\pi^k \geq \rho_k \quad \forall\, k \in K \tag{2}$$

$$\sum_{k \in K} \sum_{\pi \in P_k(a)} \lambda_\pi^k + \sum_{k \in K} \sum_{\pi \in P_k(a,s)} \lambda_\pi^k \leq \theta_a \quad \forall\, s \in S, a \in A \setminus F_s \tag{3}$$

$$\lambda_\pi^k, \theta_a \in R_+$$

The objective function is given by (1) and it is the cost of the summed network capacity. The demand constraint (2) ensures that enough capacity is established on the path pairs. The capacity constraint (3) ensures that enough capacity is allocated to route the communication on each arc a in each failure situation s which does not disrupt the arc.

The problem with this LP model is that the number of path pairs grows exponentially with the network size, and hence the complete model can only be solved for small network sizes. Instead, we will use a column generation algorithm such that only a subset of the path pairs is generated. The optimization subproblem to generate new path pairs with negative reduced costs is given in Section 3.1, and in Section 3.4 the column generation algorithm is given.

It is clear that the fractional FIPP optimization problem is a relaxation of the original FIPP optimization problem which is $\mathcal{NP}$-hard [30]. The hardness of the fractional FIPP optimization problem on the other hand is still an open problem. The fractional FIPP optimization problem can however be used for lower bounding in a branch-and-price algorithm when solving the original FIPP optimization problem. The bound can be weak, because the bound of the relaxed FIPP model is equivalent to the bound of the relaxed FBPP model, in certain cases. In other words, the quality of the bound of the relaxed FIPP model strongly depends on the network topology and the communication demand.

3.1. Subproblem: Quadratic Cost Disjoint Path Problem

For the FIPP master problem let $\alpha_k \geq 0, k \in K$, be the dual variables associated with the (negated version of) constraint (2), and let $\beta_a^s \geq 0$, $s \in S$, $a \in A \setminus F_s$, be the dual variables associated with constraint (3). Our task is to decide if there exists a pair of primary and backup paths $\pi = (p^{\text{pri}}, p^{\text{bac}})$ from some origin node o_k to some terminating node d_k with negative reduced cost for some $k \in K$.

The reduced cost of a pair of paths $(p^{\text{pri}}, p^{\text{bac}})$ is computed as follows. The cost of an arc $a \in p^{\text{pri}}$ is $\sum_{s\in S} \beta_a^s$, while the cost of an arc $a \in p^{\text{bac}}$ is $\sum_{s\in S: F_s \cap p^{\text{pri}} \neq \emptyset} \beta_a^s$. Note the asymmetry in the definition of arc costs in primary and secondary paths: For an arc on the primary path the cost is the sum taken over all failure situations, while for an arc on the backup path the sum is only taken over the failure situations that affect an arc on the primary path. The total reduced cost of $(p^{\text{pri}}, p^{\text{bac}})$ is now

$$-\alpha_k + \overbrace{\sum_{a\in p^{\text{pri}}} \sum_{s\in S} \beta_a^s}^{\text{primary path cost}} + \overbrace{\sum_{a\in p^{\text{bac}}} \sum_{s\in S: F_s \cap p^{\text{pri}} \neq \emptyset} \beta_a^s}^{\text{backup path cost}}$$

The Quadratic Cost Disjoint Path Problem (QCDPP) is to compute a pair of paths $\pi = (p^{\text{pri}}, p^{\text{bac}})$ with minimum total cost. The name of the problem comes from the fact there is a pairwise (or quadratic) dependence on the cost of the backup path as a function of the primary path. Since the dual variables β_a^s are non-negative, there clearly exists an optimal solution where both the primary path p^{pri} and the backup path p^{bac} are simple. Hence, in the following we require that the paths p^{pri} and p^{bac} are simple and arc disjoint.

3.2. MIP Model for QCDPP

A primary path is defined by the binary variables x_a for all $a \in A$ and a backup path is defined by the binary variables y_a for all $a \in A$. We define the sets $\delta^+(i)$ as the arcs going out of node $i \in V$ and $\delta^-(i)$ as the set of arcs going into node $i \in V$. We again use the set of failed arcs F_s and define the cardinality of the set as $|F_s|$, i.e., the number of arcs which fail in situation $s \in S$. The binary variables u_s for all $s \in S$ detect whether the primary path is interrupted by failure s and the binary variables v_s for all $s \in S$ detect whether the backup path is interrupted by failure s. Furthermore, the auxiliary variables z_s^a for all $s \in S$ and all $a \in A$ detect if the primary path is interrupted by failure s at the same time as the backup path uses arc a.

QCDPP

minimize:

$$c^k_{\text{reduced}} = -\alpha_k + \overbrace{\sum_{a\in A} \sum_{s\in S} \beta_a^s \cdot x_a}^{\text{primary path cost}} + \overbrace{\sum_{a\in A} \sum_{s\in S} \beta_a^s \cdot z_s^a}^{\text{backup path cost}} \quad (4)$$

subject to:

$$\sum_{a\in\delta^+(i)} x_a - \sum_{a\in\delta^-(i)} x_a = \begin{cases} 1 & i = o_k \\ -1 & i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i \in V \quad (5)$$

$$\sum_{a\in\delta^+(i)} y_a - \sum_{a\in\delta^-(i)} y_a = \begin{cases} 1 & i = o_k \\ -1 & i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i \in V \quad (6)$$

$$|F_s| \cdot u_s \geq \sum_{a\in F_s} x_a \quad \forall\, s \in S \quad (7)$$

$$|F_s| \cdot v_s \geq \sum_{a\in F_s} y_a \quad \forall\, s \in S \quad (8)$$

$$u_s + v_s \leq 1 \quad \forall\, s \in S \quad (9)$$

$$z_s^a \geq u_s + y_a - 1 \quad \forall\, s \in S, a \in A \quad (10)$$

$$x_a, y_a, u_s, v_s \in \{0,1\}, \quad z_a^s \in [0,1] \quad (11)$$

The objective function (4) is the reduced cost c^k_{reduced} of the two disjoint paths. The first double sum calculates the cost of the primary path. The second double sum then calculates the cost of the backup path. Notice that each arc a in the backup path only costs β_a^s in situation s if the primary path is disrupted in failure situation s. This is detected by the variable z_s^a. Finally the dual value α_k from constraint (2) is subtracted to calculate the corresponding reduced cost. Both the primary path variables x and the backup path variables y are constrained to form paths by constraints (5) and (6), respectively. The path disruption variables, u for the primary path and v for the backup path, are set by constraints (7) and (8) respectively. Variables u and v are then used in constraint (9) to ensure failure disjointness of the paths. In constraint (10) the auxiliary variable z_s^a is forced to the value 1 if the primary path is disrupted in situation s and the backup path uses the arc a. Finally the domains of the variables are given by constraint (11).

We consider two variants of failure situations: In the single arc failure variant there is one failure situation for each arc in A. In the single link failure variant there is one failure situation for each pair of opposite arcs, i.e., when the corresponding undirected link is broken.

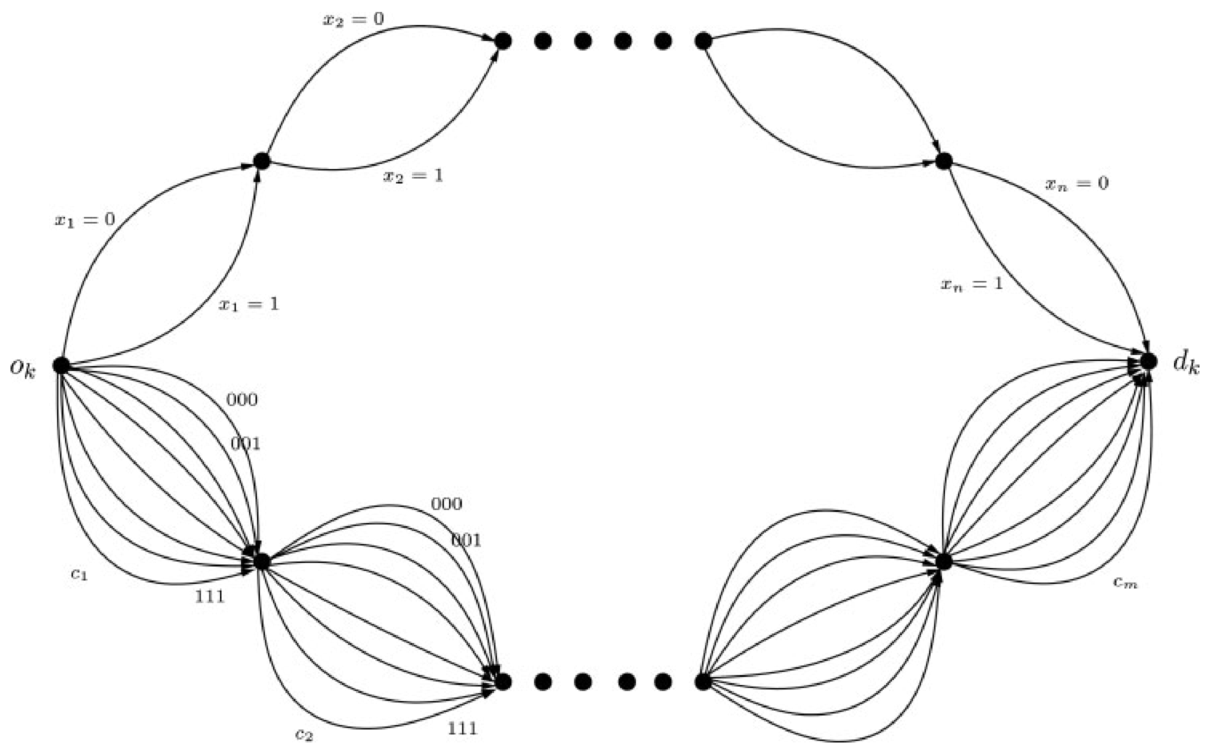


FIG. 4. Graph construction for $\mathcal{NP}$-completeness proof.

In Section 3.3 it is proved that the sub-problem above is $\mathcal{NP}$-hard. However, if instead the primary paths were precalculated and the task was to find the best backup paths, the subproblem would be a simple shortest path problem.

3.3. $\mathcal{NP}$-hardness of QCDPP

We now prove that QCDPP is strongly $\mathcal{NP}$-hard for the single arc and single link failure variants. First we present the proof for the single arc variant and then we indicate how this leads to an $\mathcal{NP}$-hardness proof for the single link variant. In the single arc variant the set of failure situations S is identical to the set of arcs A. The decision version of QCDPP with single arc failures is formally defined as follows (where the constant term $-\alpha_k$ in the objective function of QCDPP is ignored).

INSTANCE: Directed graph $G = (V, A)$, pairwise (integer and non-negative) costs β_a^f for all ordered pairs of arcs $(a, f) \in A \times A$, origin node $o_k \in V$, terminating node $d_k \in V$ and integer C.

QUESTION: Does there exist a pair of simple arc disjoint paths $\pi = (p^{\text{pri}}, p^{\text{bac}})$ from o_k to d_k in G such that

$$\sum_{a \in p^{\text{pri}}} \sum_{f \in A} \beta_a^f + \sum_{a \in p^{\text{bac}}} \sum_{f \in p^{\text{pri}}} \beta_a^f \leq C \text{ ?}$$

We prove that this problem is $\mathcal{NP}$-complete by reduction from 3-SATISFIABILITY (3SAT) [15]. It is obvious that the decision version of QCDPP is in $\mathcal{NP}$, since given $\pi = (p^{\text{pri}}, p^{\text{bac}})$ we can compute the corresponding cost and compare it with C in polynomial time.

Let (U, C) be an instance of 3SAT, where $U = (x_1, x_2, \ldots, x_n)$ is a finite set of n variables and $C = (c_1, c_2, \ldots, c_m)$ is a set of clauses where $|c_i| = 3$, $i = 1, \ldots, m$. We assume without loss of generality that each variable appears in at least one clause.

On the basis of the 3SAT instance we create an instance of the QCDPP with the structure illustrated in Figure 4. The graph consists of two chains of arcs — the so-called top chain and the bottom chain. Two node disjoint paths from o_k to d_k must necessarily have the property that one of the paths travels through the top chain while the other travels through the bottom chain. By assigning costs appropriately, we will force the primary path to use the bottom chain and the backup path to use the top chain.

We will first assume that we seek two node disjoint paths from o_k to d_k in this graph. Later we describe how we can modify the graph so that the paths become arc disjoint. Furthermore, the graph that is shown is a directed multigraph, and later we also describe how this graph can be transformed into an ordinary directed graph.

The arcs in the top chain are denoted variable arcs, while the arcs in the bottom chain are denoted clause arcs. For each clause $c_i \in C$ we have eight parallel arcs, one for each combination of assignments for the three literals; these assignments are denoted 000, 001, 010 etc. As an example, for the clause $(x_1 \vee x_2 \vee \bar{x}_3)$ the assignment 011 means that $x_1 = 0, x_2 = 1$ and $x_3 = 0$. Note that an assignment different from 000 corresponds to a satisfied clause. Similarly, we have two variable

TABLE 1. Costs β_a^f associated with variable arcs a for clause f being equal to $(x_1 \vee x_2 \vee \bar{x}_3)$.

Assignment	$x_1 = 0$	$x_1 = 1$	$x_2 = 0$	$x_2 = 1$	$x_3 = 0$	$x_3 = 1$
000	0	1	0	1	1	0
001	0	1	0	1	0	1
010	0	1	1	0	1	0
011	0	1	1	0	0	1
100	1	0	0	1	1	0
101	1	0	0	1	0	1
110	1	0	1	0	1	0
S111	1	0	1	0	0	1

arcs for each variable x_j, one arc for $x_j = 0$ and one arc for $x_j = 1$.

We will now assign pairwise costs β_a^f for all ordered pairs of arcs $(a,f) \in A \times A$. We set $\beta_a^f = 0$ for all $(a,f) \in A \times A$ except from the following pairs:

- For a clause arc a corresponding to the assignment 000 we have $\beta_a^{f'} = 1$ for one arbitrary variable arc f' (say, the arc corresponding to $x_1 = 0$). This means that if the arc a is used by a primary path from o_k to d_k then the cost of a is $\sum_{f \in A} \beta_a^f = 1$.
- For a variable arc a and clause arc f, if the variable assignment given by arc a does not match the clause assignment given by arc f, then $\beta_a^f = 1$. As an example, the variable arc a corresponding to $x_3 = 1$ has $\beta_a^f = 1$ for the arc f corresponding to the clause $(x_1 \vee x_2 \vee \bar{x}_3)$ with assignment 011. In Table 1 an extended example on how costs are assigned for variable arcs is given.

  Since we assume that each variable appears in at least one clause, each variable arc a has cost at least 1 as a primary arc, since there will be at least one clause assignment that does not match with the variable assignment given by a.

Finally, we set $C = 0$ in the QCDPP instance. Now we prove that we have a YES-instance for QCDPP if and only if we have a YES-instance for 3SAT.

Consider a YES-instance for QCDPP, that is, an instance with zero cost. Such an instance must have a primary path p^{pri} following the clause arcs from o_k to d_k, since the variable arcs have positive costs as primary path arcs. Consequently, the backup path p^{bac} must follow the variable arcs from o_k to d_k. Since the total cost of the solution $\pi = (p^{\text{pri}}, p^{\text{bac}})$ is zero, all arcs of the path p^{pri} correspond to clauses being satisfied (i.e., are different from the clause assignments 000 which have cost 1 as primary path arcs). Also, since the total cost of $\pi = (p^{\text{pri}}, p^{\text{bac}})$ is zero, the variable arcs followed by p^{bac} match the assignments in the clause arcs. Therefore, assigning the variables $x_j, j = 1, \dots, n$, to the values indicated by the path p^{bac} gives a satisfying assignment for the 3SAT-instance.

For the other direction, consider a YES-instance for 3SAT. By letting p^{bac} follow the variable arcs in the QCDPP instance as given by a satisfying 3SAT-assignment, and letting p^{pri} follow the clause arcs corresponding to the 3SAT-assignment, we obtain a solution to QCDPP of total cost zero.

By splitting each node in the graph (apart from o_k and d_k) – that is, replacing the node with an arc (u, v), and connecting all in-coming arcs to u and all out-going arcs to v — we force the paths to be arc disjoint. Furthermore, the multigraph can be transformed into an ordinary directed graph G by replacing each arc in the multigraph by a sequence of two arcs, and assigning pairwise costs appropriately. Thus we have the following:

Theorem 1. *The decision version of QCDPP when reduced to single* arc *failures is $\mathcal{NP}$-complete even when all pairwise costs are 0 or 1 (and only distinct pairs of arcs can have nonzero costs).*

Consider the directed graph G resulting from the above construction. If, for each arc in G, we add an arc in the opposite direction we obtain a graph G', where bidirectional communication is feasible for each underlying link. Consider the single link failure variant of QCDPP for the graph G', where the costs are assigned as in the construction above, but where the β_a^f costs are replaced with β_a^l costs (where l corresponds to a link). Since the primary and backup paths in G' should be simple, no backward arcs in G' will ever be used, and therefore we obtain the following:

Theorem 2. *The decision version of QCDPP when reduced to single* link *failures is $\mathcal{NP}$-complete even when all pairwise costs are 0 or 1 (and nonoverlapping pairs of arcs and links can have nonzero costs).*

3.4. Column Generation Algorithm

Given the LP model in the introduction of Section 3 we can now apply column generation to solve the model. The subproblem described in Section 3.2 is either solved using a MIP solver or by using the labeling algorithm described in Section 4. Below we briefly describe the column generation algorithm.

In the column generation algorithm (Fig. 5) we first initialize α_k and β_a^s with artificial values: $\alpha_k = \sum_{a \in A} c_a$ and

```
Initialize $\alpha_k$,$\beta_a^s$
$k = 1$
do
    $k' := k$
    do
        SOLVE QCDPP($k$,$\alpha_k$,$\beta_a^s$)
        $k := k + 1$
    while $c_{reduced}^{p,k} \geq 0$ and $k' \neq k$
    Update set of path pairs
    SOLVE FIPP with new set of path pairs
    Update $\alpha_k$,$\beta_a^s$
while $k' \neq k$
```

FIG. 5. Column Generation algorithm.

$\beta_a^s = \frac{c_a}{|S|}$ (where S is the set of failure situations). This means that it is always profitable to include a path pair of primary and backup paths for each demand k. After entering the main loop, promising path pairs are found based on the current values of α_k and β_a^s. The resulting paths are then added to the set of path pairs and the master problem is solved with the new set of path pairs. This process continues until no negative reduced-cost path pair for any demand can be found.

4. LABELING ALGORITHM FOR THE QCDPP SUBPROBLEM

The QCDPP can be formulated as a Shortest Path Problem with Resource Constraints (SPPRC). The SPPRC is a common subproblem in many graph-based problems when using a column generation based algorithm, e.g., the Vehicle Routing Problem with Time Windows [21, 22] and the Crew Pairing Problem [9]. In the following we briefly define the SPPRC, discuss complexity issues and the application of recent developments within this area, and describe the basic labeling algorithm. Last we will present the reformulation of the QCDPP into an SPPRC.

The SPPRC can be stated as: Given a directed graph $G' = (V', A')$ with nodes V' and arcs A', and a set of resources R. For each node $i \in V'$ and arc $(i,j) \in A'$ there is associated a cost. Furthermore, there is a weight for each resource $r \in R$ by which r is accumulated when visiting node i or traversing arc (i,j), as well as a lower and upper limit on r. The resource weight is often defined as a constant but can be determined by a function of the previous visited nodes, arcs, and resource values. The accumulated amount of r must respect the lower and upper limits when arriving at i or when traversing (i,j). The increase in resource consumption and cost of a path when extended along an arc is defined by a function, sometimes denoted a resource extension function [19]. The objective is to find a minimum cost path from an origin node $o \in V'$ to a destination node $d \in V'$, where the resources satisfy the limits for all resources $r \in R$. In many cases it suffices to have the limits of the resources only at the nodes; in these cases the limits on the arcs can be made nonbinding.

The SPPRC is $\mathcal{NP}$-hard in the weak sense when the number of resources is a constant and can be solved with dynamic programming-based labeling algorithms in pseudo-polynomial time. An extension of the SPPRC is the node elementary version; the elementary shortest path problem with resource constraints (ESPPRC) where paths must be simple. The constraint of being elementary can be enforced with the use of a binary resource for each node to indicate if the node is visited on the path. Thus ESPPRC can be solved as an SPPRC. The ESPPRC is strongly $\mathcal{NP}$-hard, see [12]. However, if G' does not contain negative cost cycles the additional resources can be disregarded since a least cost path that is simple will always exist; therefore, the problem can be solved in pseudo-polynomial time.

Although the reformulation (see details below) of the QCDPP into a SPPRC leads to a graph with no negative cost cycles, the number of resources amounts to one binary resource per failure scenario, i.e., one per two arcs in G for the single link failure case in the QCDPP. That is, the number of resources in the SPPRC depends on the input to the QCDPP. Hence the complexity of the labeling algorithm is exponential when regarding the reformulation of the QCDPP. Also, it is important to note that the reformulation of the QCDPP into a SPPRC results in a nonconstant extension function where the cost of the arcs on the backup path depend on the failure scenarios that are affected by the primary path.

4.1. Related Approaches

A comprehensive overview of work related to SPPRC is outside the scope of this article, but we will briefly discuss some recent results. For further details on mathematical models and solution methods we refer the reader to the survey of Irnich and Desaulniers [20].

Dynamic programming based methods denoted labeling algorithms are to date the most dominant approaches for solving the SPPRC. However, Carlyle et al. [8] present a Lagrangian relaxation based method where the resource constraints are relaxed to a shortest path problem solvable in polynomial time. The approach is applicable for problems with no negative cost cycles and shows good results when few resources are considered. However, because of the nature of the nonconstant extension function on the arc costs in our reformulation this approach is not directly applicable; also we consider a large number of resources which may limit the effect of the Lagrangian relaxation.

Dumistrescu and Boland [13] present an improved preprocessing for the SPPRC (with no negative cost cycles) and embed it into a labeling algorithm. They present resource lower bound calculations using Lagrangian relaxation, hence solving a shortest path problem. Again this approach is not applicable in our case because of the arc cost extension function in our reformulation. Furthermore, this approach has very limited use when only considering binary resources, which is indeed the case for our reformulation, since the resource bounds are already very tight.

Feillet et al. [14] address the ESPPRC and propose to consider unreachable nodes instead of visited nodes with the binary resources. The unreachability of a node is determined based on limits on other resources. In our context this would correspond to deciding if a failure scenario cannot be triggered. However, this is difficult to decide without actually visiting the arcs of the scenario, since triggering a scenario does not directly depend on other resources but on the topology of the graph. Therefore, the unreachability concept cannot readily be used in our case.

A very successful labeling algorithm by Righini and Salani [28] showed how a significant speedup can be gained by using a bi-directional approach. That is, based on a monotone resource (e.g., the number of nodes on the path) a breaking point is chosen (e.g., when half the nodes have been visited) and the labeling algorithm is run from both sides. By splicing paths starting at the origin node o with a reverse path coming from the destination node d one can construct a full path. For

```
$Q := \emptyset$
$Q' := \emptyset$
Initialize label $L_o$
ENQUEUE$(Q, L_o)$
while $Q$ is not empty
  $L :=$ DEQUEUE$(Q)$
  for each node $i \in$ FEASIBLE EXTENSION$(L)$
    $L_i :=$ EXTEND LABEL$(L, i)$
    if $i = d$
      then ENQUEUE$(Q', L_i)$
      else ENQUEUE$(Q, L_i)$
  REMOVE DOMINATED$(Q)$
return $Q'$
```

FIG. 6. Pseudo-code for labeling algorithm.

this method to work all extension functions must be reversible which unfortunately is not the case for our objective function.

Boland et al. [6] and Righini and Salani [29] independently proposed to relax the state-space of the labeling algorithm such that only a subset of resources is considered to begin with. Any violated resource is then added iteratively until a feasible path has been found. By the construction of the graph and the definition of the objective function used in our reformulation, it is doubtful that this approach would perform satisfactorily since relaxing resources would yield zero weight arcs in the associated backup path, making it necessary to add resources until all feasible backup paths are covered.

4.2. *The General Labeling Algorithm*

In a labeling algorithm the labels represent partial paths that are extended (using the extension functions) in all feasible directions from the origin node o. Each label L (a vector with $|R| + 1$ components) stores the cost of the partial path $T_{\text{cost}}(L)$ and the current value $T_r(L)$ of each resource $r \in R$. To avoid enumerating all feasible paths in G', only Pareto-optimal labels (i.e., labels that are not proved to be dominated by other labels) are kept during the execution of the algorithm. When using non-decreasing extension functions (which is the case for the reformulation of QCDPP), the label dominance criterion can be stated as follows.

Proposition 1 ([10]). *Let L and L′ be two labels representing partial paths ending at the same node. Label L dominates label L′ (which can be discarded) if*

$$T_{\text{cost}}(L) \leq T_{\text{cost}}(L')$$
$$T_r(L) \leq T_r(L') \quad \forall r \in R.$$

When equality holds for all label components, one of the two labels must be kept. Figure 6 summarizes the concept of a labeling algorithm. The initial state is represented by the label L_o at the starting node. This label is enqueued on a priority queue Q that keeps track of all unprocessed labels. The algorithm runs until all labels have been processed. In each iteration the next label L from Q is dequeued. The set of nodes (Feasible Extension(L)) that are feasible extensions of the partial path represented by L, with regard to connectivity and resource limits, is determined. L is extended to these nodes using the resource extension functions (implemented in Extend Label(L, i)) to create the new label L_i for node i. If the extended label L_i is extended to the end node d it is stored as a solution in the queue Q'; otherwise L_i is enqueued on Q for future processing. Then Q is cleaned for dominated labels so only Pareto-optimal labels remain.

4.3. *Transformation and Solution of Subproblem*

Next, we consider the transformation of the QCDPP stated as (4)-(11) into a SPPRC Recall the graph $G = (V, A)$ for the QCDPP where a minimum cost primary and backup path pair must be found from $o_k \in V$ to $d_k \in V$ over all $k \in K$. Let $V' = \{i' : i \in V\}$ be a copy of all nodes in V and let $A' = \{(j', i') : (i, j) \in A, i', j' \in V'\}$ be a reversed version of all arcs in A connecting the nodes in V'. Also let $A''_k = \{(d_k, d'_k)\}$ be the arc connecting $d_k \in A$ and $d'_k \in A'$ for demand pair k. The transformed graph for the kth demand pair is then $G'_k = (V \cup V', A \cup A' \cup A''_k)$ where a primary path will be sought in the first part of the graph with nodes V, then by the arc (d_k, d'_k) the search is switched to the other part of the graph consisting of the nodes V' where a reverse backup path is found. G'_k is illustrated in Figure 7.

For each failure situation $s \in S$ it must be ensured that no arcs from F_s are used on the backup path if any of the arcs in F_s were used on the primary path. A binary resource is added

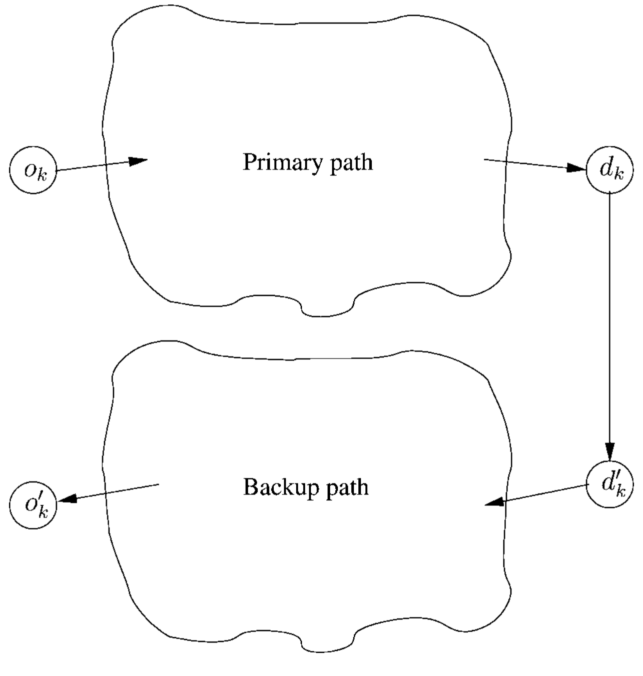


FIG. 7. The transformed graph for the kth demand pair. The backup path part of the graph is a reversion of the primary path part, i.e., the path found is a forward directed primary path and a reversed backup path.

TABLE 2. Tested networks and their characteristics.

Network	Nodes	Links	Avg. Node degree	No. demands
Cost239 [4]	11	26	4.73	55
Europe	13	21	3.23	78
Newyork [1]	16	49	6.12	120
Ta1 [1]	24	51	4.25	276
FranceSND [1]	25	45	3.60	300
Norway [1]	27	51	3.77	351
USA [11]	28	45	3.21	378
Cost266 [1]	37	57	3.08	666

for each failure situation $s \in S$. Hence, the set of resources has size $|S|$. Let a label L consist of $1 + |S|$ components. $T_{\text{cost}}(L)$ is the cost of the path and $T_s(L)$ for $s \in S$ is the bit value of the failure situation resources. $T_s(L)$ will be set to one if the failure scenario s is triggered on the primary path, and resource limits are enforced on the arcs when extending labels. The upper bound for resource $s \in S$ when extending a label on arc a' is 0 for $a' \in A' \wedge a \in F_s$ and 1 otherwise. That is, a label L cannot be extended on arc $(i', j') \in A'$ with $(j, i) \in F_s$ for $s \in S$ on the backup path if arc $a \in F_s$ is used on the primary path, i.e., the resource value $T_s(L) = 1$ and the upper bound for s on (i', j') is 0. Hence, in Figure 6 the end node of a is not in the set Feasible Extension(L). Recall that the cost of the backup path depends on the arcs used on the primary path and that $\beta_a^s \geq 0$ and $\alpha_k \leq 0$. The extension along an arc $a = (i, j)$ of a label L (implemented in Extend Label(L, i)) representing a partial path ending at node i proceeds as follows to create a new label L' (ending at node j):

$$T_{\text{cost}}(L') = T_{\text{cost}}(L) + \begin{cases} \sum_{s \in S} \beta_a^s & a \in A \\ \sum_{s \in S: T_s(L)=1} \beta_{(j,i)}^s & a = (i', j') \in A' \\ -\alpha_k & a \in A_k'' \end{cases}$$

$$T_s(L') = \begin{cases} 1 & a \in F_s \\ \max\{T_s(L), 0\} & \text{otherwise} \end{cases} \quad s \in S$$

Both extension functions are nondecreasing, and so the dominance criterion of Proposition 1 can be applied in the labeling algorithm. For the kth pricing problem a path represented by label L ending in o_k' has the cost:

$$c_{\text{reduced}}^k = -\alpha_k + \overbrace{\sum_{a \in A(L)} \sum_{s \in S} \beta_a^s}^{\text{primary path cost}} + \overbrace{\sum_{a=(i',j') \in A'(L)} \sum_{s \in S: T_s(L)=1} \beta_{(j,i)}^s}^{\text{backup path cost}} \tag{12}$$

where $A(L)$ and $A'(L)$ are the set of arcs used in A and A', respectively. Minimizing expression (12) is equivalent to the objective function stated in (4) and the path found by the labeling algorithm can trivially be split into a primary and a backup path.

5. RESULTS

In this section, the efficiency of the FIPP protection method is tested on eight different networks. Basic network data for the 8 networks is given in Table 2. We have chosen to use the simple demand matrix $D^{kl} = 1$ for each pair of nodes (k, l).

We performed a series of experiments on a computer with 2.4 GHz AMD Opteron 250 dual-core processors and with 12 GByte RAM. The computer ran openSUSE 11.0 Linux and we used CPLEX 9.13. In Tables 3 and 4 we compare the computation times when the QCDPP subproblem is solved using the SPPRC labeling algorithm and a standard MIP solver, respectively.

It can be seen from Tables 3 and 4 that the SPPRC labeling algorithm is significantly faster on all tested networks. Furthermore, three of the networks, Norway Ta1 and Cost266, cannot be solved using the MIP solver due to excessive memory consumption.

Given the column generation algorithm, we are now able to compute the optimal protection capacity required for relaxed FIPP protection (Table 5). We find the results in Table 5 interesting because it shows how efficient the relaxed FIPP method is. The FIPP method uses at most 8% extra network capacity

TABLE 3. SPPRC labeling algorithm results.

		Columns					Time		
Network	Rows	Initial	Final	PerIt	PerDem	Iter	Total	CG	CGPct
Cost239 [4]	705	81	1451	42.81	0.78	32	11	1	4.56
Europe [1]	498	99	470	46.38	0.59	8	1	1	36.36
Newyork [1]	2472	169	5292	47.44	0.40	108	2438	1875	76.94
Ta1 [1]	2826	327	4013	43.88	0.16	84	17612	17385	98.71
FranceSND [1]	2280	345	2944	57.76	0.19	45	235	191	81.29
Norway [1]	2901	402	3704	58.96	0.17	56	1177	967	82.22
USA [11]	2358	423	3076	60.30	0.16	44	156	77	49.65
Cost266 [1]	3858	723	6516	62.29	0.09	93	2050	1051	51.29

Rows: Number of rows in LP. Initial: Initial number of master problem columns. Final: Final number of master problem columns. PerIt: Number of columns added per iteration. PerDem: Number of columns added per iteration per demand. Iter: Number of column generation iterations. Total: Total running time in seconds. CG: Total column generation running time in seconds. CGPct: Column generation (labeling algorithm) solution time as percentage of total time.

TABLE 4. MIP results.

		Columns					Time		
Network	Rows	Initial	Final	PerIt	PerDem	Iter	Total	CG	CGPct
Cost239 [4]	705	81	677	1.00	0.02	597	154	145	93.77
Europe [1]	498	99	307	1.00	0.01	209	31	31	98.36
Newyork [1]	2472	169	2328	1.00	0.01	2160	6491	5943	91.56
Ta1 [1]	2826	—	—	—	—	—	—	—	—
FranceSND [1]	2280	345	1408	1.00	0.00	1064	9434	9356	99.17
Norway [1]	2901	—	—	—	—	—	—	—	—
USA [11]	2358	423	1532	1.00	0.00	1110	2406	2304	95.77
Cost266 [1]	3858	—	—	—	—	—	—	—	—

Rows: Number of rows in LP. Initial: Initial number of master problem columns. Final: Final number of master problem columns. PerIt: Number of columns added per iteration. PerDem: Number of columns added per iteration per demand. Iter: Number of column generation iterations. Total: Total running time in seconds. CG: Total column generation running time in seconds. CGPct: Column generation (MIP) solution time as percentage of total time.

compared with the theoretical lower bound achieved using Complete Rerouting [31] and on average only 4% extra network capacity. We acknowledge that this is only part of the story; when the demands are required to be integer (for each demand the entire communication flow is routed on the same primary path and the same backup path) the ROBB is going to increase.

6. FUTURE RESEARCH

The mathematical model on which we base our results is by choice constructed to be as simple as possible. A number of additional model features can be incorporated into the model and some of these may certainly change the above conclusions. In this section we briefly describe two important model refinements.

First, in the current model we consider the demands as a volume of communication ρ_k to be established between two nodes in the network. In the fractional FIPP problem this volume may be divided between a number of path pairs and this is probably not desirable for the communication customers. Instead, each customer should be offered one path pair with a certain volume of traffic—corresponding to the original FIPP problem. For the model presented in Section 3, this results in more variables, and furthermore, these variables have to be binary variables. Hence, to solve this model to optimality, a branch-and-price optimization algorithm is necessary.

Second, in the current model there is no bound on the capacity θ_a of an arc $a \in A$. In real-life applications, capacities are acquired in modular amounts and economies of scale can be modeled. Modular capacities can be included into the model by changing the right hand side of constraints (3) to a sum of integer variables, as shown in the modified constraints (13) below:

$$\sum_{k \in K} \sum_{\pi \in P_k(a)} \lambda_\pi^k + \sum_{k \in K} \sum_{\pi \in P_k(a,s)} \lambda_\pi^k \leq \sum_m C_m \cdot \theta_{a,m} \quad \forall\, s \in S, a \in A \setminus F_s$$

Here the capacity variables $\theta_{a,m} \in Z^+$ correspond to different types of connections, each possessing a capacity C_m. The objective function is then modified to include different prices for each type of technology. The price per capacity unit reflects the economies of scale.

TABLE 5. FIPP protection method comparison.

Network	NP capacity	CR RROB	FIPP RROB	Difference
Cost239	86	0.13	0.19	0.06
Europe	158	0.57	0.65	0.08
Newyork	412	0.19	0.24	0.05
Ta1	733	0.76	0.78	0.02
FranceSND	9825	0.66	0.67	0.01
Norway	61	0.59	0.61	0.02
USA	1273	0.50	0.55	0.05
Cost266	14587	0.62	0.64	0.02
Avg.		0.50	0.54	0.04

NP capacity: Non-Protected required network capacity. CR RROB: Complete Rerouting [31] required network capacity relative to NP capacity. FIPP RROB: FIPP required network capacity relative to NP capacity. Difference: Absolute difference between RROB for CR and FIPP.

7. CONCLUSION

In this article we presented an LP model for the fractional FIPP optimization problem. The LP model was solved using column generation. We analyzed the subproblem, proved it to be strongly $\mathcal{NP}$-hard, and devised a labeling algorithm for solving the subproblem more efficiently. Finally, we evaluated the capacity efficiency of the FIPP method on a number of network instances. The results indicate that the FIPP method appears to be a very efficient protection method—on an average only requiring 4% more network capacity than complete rerouting, the absolute lower bound for single link failure protection.

REFERENCES

[1] Survivable network design data library, webpage. Available at: http://sndlib.zib.de.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network flows: Theory, algorithms, and applications, Prentice Hall, Englewood Cliffs, NJ, 1993.

[3] J. Anderson, B. Doshi, S. Dravida, and P. Harshavardhana, Fast restoration of ATM networks, IEEE J Select Areas Commun 12 (1994), 128–138.

[4] P. Batchelor, B. Daino, P. Heinzmann, D.R. Hjelme, P. Leuthold, R. Inkret, G.D. Marchis, H.A. Jager, F. Matera, M. Joindot, B. Mikac, A. Kuchar, H.-P. Nolting, E. Coquil, J. Spath, F. Tillerot, B. Caenegem, N. Wauters, and C. Weinert, Study on the implementation of optical transparent transport networks in the European environment-results of the research project COST 239, Photonic Network Commun 2 (2000), 15–32.

[5] R. Bhandari, Survivable networks — Algorithms for diverse routing, Kluwer, Boston, 1999.

[6] N. Boland, J. Dethridge, and I. Dumitrescu, Accelerated label setting algorithms for the elementary resource constrained shortest path problem, Oper Res Lett 34 (2006), 58–68.

[7] E. Calle, J.L. Marzo, and A. Urra, Protection performance components in MPLS networks, Comput Commun 27 (2004), 1220–1228.

[8] W.M. Carlyle, J.O. Royset, and R.K. Wood, Lagrangian relaxation and enumeration for solving constrained shortest-path problems, Networks 52 (2008), 256–270.

[9] G. Desaulniers, J. Desroisers, Y. Dumas, S. Marc, B. Rioux, M.M. Solomon, and F. Soumis, Crew pairing at Air France, Eur J Oper Res 2 (1997), 245–259.

[10] G. Desaulniers, J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis, and D. Villeneuve, "A unified framework for deterministic time constrained vehicle routing and crew scheduling problems," Fleet Management and Logistics, T. Crainic, G. Laporte (Editors), Kluwer, Norwell, MA, 1998, Chapter 3, pp. 57–93.

[11] J. Doucette, D. He, W.D. Grover, and O. Yang, Algorithmic approaches for efficient enumeration of candidate p-cycles and capacitated p-cycle network design, Proc 4th Int Workshop Design Reliable Commun Networks, Banff, Canada, 2003, pp. 212–220.

[12] M. Dror, Note on the complexity of the shortest path models for column generation in VRPTW, Oper Res 42 (1994), 977–978.

[13] I. Dumitrescu and N. Boland, Improved preprocessing, labeling and scaling algorithms for the Weight-Constrained Shortest Path Problem, Networks 42 (2003), 135–153.

[14] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems, Networks 44 (2004), 216–229.

[15] M.R. Garey and D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, Freeman, New York, 1979.

[16] W.D. Grover, Mesh-based survivable networks, Prentice Hall, Saddle River, NJ, 2004.

[17] D. Haskin and R. Krishnan, A method for setting and alternative label switch path to handle fast reroute, Work in progress, Internet Engineering Task Force, 2002. Available at: www.ietf.org.

[18] J.Q. Hu, Diverse routing in optical mesh networks, IEEE Trans Commun 51 (2003), 489–494.

[19] S. Irnich, Resource extension functions: Properties, inversion, and generalization to segments, OR Spectr 30 (2008), 113–148.

[20] S. Irnich and G. Desaulniers, "Shortest path problems with resource constraints, Column generation," G. Desaulniers, J. Desrosiers, M. M. Solomon (Editors), Springer, New York, 2005, Chapter 2, pp. 33–65.

[21] S. Irnich and D. Villeneuve, The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$, INFORMS J Comput 18 (2006), 391–406.

[22] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger, Subset-row inequalities applied to the vehicle-routing problem with time windows, Oper Res 56 (2008), 497–511.

[23] P. Laborczi and T. Cinkler, Efficient algorithms for physically-disjoint routing in survivable GMPLS/ASTN networks, Proc 11th Telecommun Network Strategy Plann Symp, Vienna, Austria, 2004, pp. 185–192.

[24] P. Laborczi, J. Tapolcai, P.-H. Ho, T. Cinkler, A. Recski, and H.T. Mouftah, Algorithms for asymmetrically weighted pair of disjoint paths in survivable networks, Proc 3th Int Workshop Des Reliable Communication Networks, Budapest, Hungary, 2001, pp. 220–227.

[25] J.F. Maurras and S. Vanier, Network synthesis under survivability constraints, 4OR 2 (2004), 53–67.

[26] S. Orlowski, Local and global restoration of node and link failures in telecommunication networks, Studiengang Wirtschaftsmathematik, Technische Universität Berlin, Berlin, Germany, 2003.

[27] M. Pioro and D. Medhi, Routing, flow, and capacity design in communication and computer networks, Morgan Kaufmann, San Francisco, 2004.

[28] G. Righini and M. Salani, Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints, Discrete Optim 3 (2006), 255–273.

[29] G. Righini and M. Salani, New dynamic programming algorithms for the resource constrained elementary shortest path problem, Networks 51 (2008), 155–170.

[30] T. Stidsen, M. Kiese, B. Petersen, S. Spoorendonk, and M. Zachariasen, Network capacity planning with shared protection, in press.

[31] T. Stidsen and P. Kjærulff, Complete rerouting protection, J Opt Network 5 (2006), 481–492.

[32] T. Stidsen and T. Thomadsen, Joint routing and protection using p-cycles, Technical Report-2005-10, Technical University of Denmark, Denmark, 2005. Available at: http://www2.imm.dtu.dk/pubdb/p.php?3939.

[33] J. Suurballe, Disjoint paths in a network, Networks 4 (1974), 125–145.

[34] G. Swallow and L. Andersson, MPLS Working Group, Webpage. Available at: http://www.ietf.org/html.charters/mpls-charter.html.