Optimization algorithms for resilient path selection in networks

Marco Casazza*, Alberto Ceselli

Dipartimento di Informatica, Università degli Studi di Milano, Italy

Abstract

We study a Resilient Path Selection Problem (RPSP) arising in the design of communication networks with reliability guarantees. A graph is given, in which every arc has a cost and a probability of failure, and in which two nodes are marked as source and destination. The aim of our RPSP is to find a subgraph of minimum cost, containing a set of paths from the source to the destination nodes, such that the probability that all paths fail simultaneously is lower than a given threshold. We explore its theoretical properties and show that, despite a few interesting special cases can be solved in polynomial time, it is in general NP-hard. In fact, we prove that even deciding if a given subgraph has a probability of failure not exceeding a given threshold is already NP-Complete. We therefore introduce an integer relaxation that simplifies the computation of such probability, and we design an exact algorithm for the full RPSP exploiting this relaxation and other ad-hoc procedures. We present computational results, highlighting that our exact algorithms can handle graphs with up to 30 nodes within minutes of computing time, consistently producing proven optimal solutions. Moreover, we show that our algorithms can be used also as heuristics, outperforming path protection schemes from the literature also on much larger networks.

Keywords: network reliability, failure probability, path selection, column generation, branch and price

1. Introduction

In a world where everything is connected, network services are an asset. Everyday more and more companies rely on the availability of their online platforms, and the trend is to push services to the cloud in order to reduce costs. Within such a context, network reliability is a key factor for the success of a business. Both physical and logical links are subject to failures: entire portions

*Corresponding author

Email addresses: marco.casazza@unimi.it (Marco Casazza), alberto.ceselli@unimi.it (Alberto Ceselli)

of a network or a virtual network function can fail due to local disruptions, and the overall probability of success of a communication, called *availability*, is often subject to Service Level Agreements (SLA) between network operators and clients.

Operators are well aware of possible adverse phenomena; they continuously need to balance availability with network costs, such as the usage of the links of the network, often expressed as the sum of costs of single links. From a cost perspective, simple paths are generally the cheapest solution, as they only select those links that are necessary to establish the connection with no redundancy; however, they are too fragile: the failure of a single link (or node) induces the overall connection failure. Path protection schemes are known since decades: their main philosophy is to always keep one or more backup paths [1], to re-establish connectivity in case of breakdowns in the primary one. However keeping backup paths is not free of costs: they are normally as expensive as primary ones to setup and maintain. Furthermore, each additional path protects only against the failure of *one* additional link: more reliable schemes can only be obtained by very expensive solutions.

We also mention that similar scenarios arise in logistics, when planners need to find a routing for goods or people in a network where links availability is uncertain, and checking has a cost. It is the case of online flight search engines, for instance: the queries for flights are normally free for the user, but subject to charge for the company.

We address what we name the *Resilient Path Selection Problem (RPSP)*. It is given a network where each arc has both a cost and a probability of failure. Two nodes of the network are marked as endpoints of connections. The RPSP is the problem of finding a minimum cost subgraph, whose probability of containing paths between endpoints which all fail simultaneously (that is, none of them is available) is lower than a given threshold.

Literature review. Network reliability includes a wide range of different works that share a common feature: the components of the network (i.e., links and nodes) may fail. However, the arising optimization problems are very different on the basis of a few distinguishing features, such as which of the components is assumed to be unreliable, the network topology, additional operational constraints involving distances [2].

Indeed, while setup, maintenance or even query costs can be modeled with standard techniques from combinatorial optimization, availability involves *probabilities*, and requires therefore a different setting.

Those problems where nodes are not reliable are usually variants of the well-known Uncapacitated Facility Location Problem: nodes are modeled as facilities and the aim is to minimize both the cost for reinforcing them, and the penalty in case of failures which either require reassignments or leave customers without service [3]. Another option is backing up facilities, assigning customers to both a primary and a secondary facility: if the primary fails its customers are seamlessly served by the secondary. This case is addressed for example by [4] and by [5], where the idea of *extensive* facilities is taken into account.

Multiple backup facilities are also an option: in [6] the authors consider the problem where facilities are partitioned into two sets: unreliable ones that may fail, and reliable ones that do not fail, though more expensive. Each customer must be assigned to a sequence of facilities, either reliable or unreliable, in such a way that if the first facility of the sequence fails, the customer is served by the next one in the sequence as a backup, until the first reliable facility is found or a penalty is paid for the unavailable service.

However there is no trivial way to map reliability problems involving paths, like the RPSP, to problems on nodes. Therefore these models can hardly be applied to our case.

For what concerns the failure of the links, the works in the literature usually revolve around the problem of connecting two terminal nodes. A research stream simplifies the setting by considering solutions which are always composed by a single path. As representative recent examples, in [7] the authors study a variant of the resource constrained shortest path problem where, given an upper bound to the cost of the path and a probability of resource consumption, a path must be found that maximizes the probability of not exceeding the available resources. They propose branch-and-bound approaches that solve instances with up to 50 nodes within one hour and a half of computing time. Similarly, [8] proposes an approach to find the most reliable multi-leg (single path) flight itinerary to reach a destination within a time limit. The authors suppose that each edge of the network is a leg departing and arriving at given times, which are modeled by certain probability distributions.

Another simplification made in the literature is to consider only collections of *independent* paths, that is being either arc or node disjoint. For instance, in [9] the authors study the problem of finding a set of $k \geq 2$ arc-independent paths such that the probability of connecting the source to the destination is at least a given threshold and the cost of the selected arcs is minimized.

In [10] the authors propose a tabu search approach for a survivable network design problem where each pair of source and destination node of the graph defines the number of disjoint paths that must be found to establish a reliable connection.

The more general problem of establishing reliable communications is considered in works such as [11], where the authors investigate on the complexity of computing the probability of failure of a given network, and [12], where the authors propose heuristic methods and a mathematical model to actually find reliable networks, based on an enumeration of all the possible states of the network. However, since the number of states is exponential in the number of network nodes, the authors avoid the generation of all the states at the cost of losing any optimality guarantee on the solutions obtained.

When the problem consists in connecting all terminal nodes in a graph, since computing the reliability of any generic network is known to be NP-hard, evolutionary metaheuristics have been exploited [13].

Also in [14] the authors attack the problem of finding a subgraph connecting all nodes of a graph in such a way that a given availability probability threshold is satisfied. The authors make the assumption that the probabilities of failure of

the links are all the same, which eventually lead to a simpler way of computing the reliability of the network: after this simplification the probability of failure depends only on the number of the selected links. In fact, the authors decompose the problem in a finite set of subproblems, each forcing solutions with a fixed number of selected links.

A different type of simplification is applied in [15], where the authors study the problem of maximizing the reliability of a network under budget constraints and propose an integer formulation using sample average approximation.

We finally report that there are indeed general means of handling stochastic components in optimization problems [16]. One of them is the methodology of chance-constrained programming. While they vastly prove effective in continuous optimization, successful applications to combinatorial problems require either to trade genericity for efficacy, or to embed specific problem structure and additional theoretical properties. In fact, the approach from the literature that is the closest to our setting is that of [17], where the authors also search for a subgraph of minimum cost connecting two endpoints. A reliability constraint, ensuring a path to exist between the endpoints, is modeled to be respected in probability. In order to obtain resolution algorithms, the authors assume that failures are not completely random, but instead a restricted set of failure scenarios is known, together with their realization probabilities. When this is the case, their formulation can be effectively optimized by exact Benders decomposition techniques. When this is not the case, heuristic solutions need to be found by a-priori sampling a set of scenarios, using Monte Carlo methods, and setting uniform scenario realization probabilities.

Main contributions and outline. In this paper we combine modeling generality, theoretical guarantees and computing affordability. We introduce an integer linear programming formulation for the RPSP, allowing us to explicitly model arbitrary failures on links, without any need for simplifications on the probability computations or approximations by a-priori sampling.

We investigate its theoretical properties, and propose exact branch-and-price solution algorithms. Our contribution is mainly methodological, as prior contributions from the literature make it clear that effective exact handling of probabilities in combinatorial optimization problems like the RPSP requires the careful design of novel algorithmic techniques. However, our algorithms allow us also to experimentally assess the practical gain of using RPSP models instead of more traditional path protection mechanisms.

In Section 2 we formalize the RPSP, we provide a mathematical programming formulation, and we highlight a few key theoretical properties. They include general hardness proves as well as simplifying conditions and polynomially solvable special cases. Unfortunately, it turns out that even checking if the availability of a *given* subgraph is above a given threshold in NP-Complete. Therefore, in Section 3 we introduce a suitable model for feasibility by failure probability computations, exploiting Bayesian Networks. Then, in Section 4 we propose a noticeable integer relaxation, whose simplifying idea is to consider path failures as independent events, and we design an exact branch-and-price

algorithm that combines such a relaxation and the feasibility models to solve the RPSP to proven optimality. In Section 5 we present the results of an extensive experimental analysis. In Section 6 we draw some conclusions.

2. Modeling and theoretical properties

A graph $G = (N, A)$ is given, where N is the set of nodes and A is the set of arcs. A source node $s \in N$ and a destination node $t \in N$ are also given, and for each arc $a \in A$ we are given a cost c_a and its probability of failure $0 \leq p_a \leq 1$. In this paper we perform the mild assumption that failures of single arcs are independent events, even when they belong to the same path.

A feasible path $r = (\sigma_1, \sigma_2, \ldots, \sigma_k)$ is a sequence of nodes σ_m having $\sigma_1 = s$, $\sigma_k = t$, and such that it exists an arc between each pair of nodes σ_m and σ_{m+1}, for $m = 1, \ldots, k-1$.

Given a SLA maximum failure target $0 \leq \mathcal{F} \leq 1$, a RPSP solution is a set of feasible (not necessarily disjoint) paths $S = \{r_1, r_2, \ldots\}$ such that the probability of failure of all the *paths* connecting s to t is at most $\mathcal{F}$ or, in other words, that the probability of having an available connection from s to t is at least $\mathcal{A} = 1 - \mathcal{F}$. The cost of a solution is the sum of the costs of the *arcs* traveled by at least one path in S. A feasible RPSP solution is optimal when its cost is minimum.

It is peculiar that failure probabilities are measured on selected paths, while costs on selected arcs. Intuitively, the RPSP is the problem of selecting a subgraph, which is a superposition of possibly overlapping paths, such that $\mathcal{F}$ is not exceeded and the sum of the costs of the selected arcs is minimum. When an arc a is traversed by at least one path, its cost c_a is paid. The cost of an arc is paid only once even if it is shared by many paths.

2.1. Mathematical programming formulation

Let R be the set of all feasible paths. We encode each path $r \in R$ as a pattern $\bar{\mathbf{z}}^r \in \mathbb{B}^{|A|}$ having $\bar{z}^r_a = 1$ if arc a belongs to r, and 0 otherwise. We therefore model the RPSP as follows:

$$\min \quad \sum_{a \in A} c_a \cdot x_a \tag{1}$$

$$\text{s.t.} \quad P(\mathbf{y}) \leq \mathcal{F} \tag{2}$$

$$\bar{z}^r_a \cdot y^r \leq x_a \qquad \forall a \in A, r \in R \tag{3}$$

$$x_a \in \mathbb{B} \qquad \forall a \in A \tag{4}$$

$$y^r \in \mathbb{B} \qquad \forall r \in R. \tag{5}$$

Each y^r is a binary variable associated to a path r and it is set to 1 if path r is selected, 0 otherwise. We also denote as $\mathbf{y}$ the vector of all y^r variables. Each x_a is a binary variable that is set to 1 if arc a belongs to at least one path in the solution, and 0 otherwise. The $P(\mathbf{y})$ term is a function computing the probability that all the paths having $y^r = 1$ fail simultaneously. The objective

function (1) aims at minimizing the overall cost of the selected arcs. Constraint (2) imposes that the overall failure probability of the set of selected paths does not exceed the SLA target $\mathcal{F}$. Constraints (3) impose that no path containing arc a is selected, unless variable x_a is set to 1; symmetrically, when costs are positive, x_a is set to 1 only if arc a belongs to at least one selected path.

2.2. Theoretical properties

From now on we assume w.l.o.g. that $c_a > 0$, as it is profitable to fix as select in preprocessing any arc with $c_a \leq 0$ (and either use it in selected paths or not). Let us denote as $\bar{p}^r = 1 - \prod_{a \in A}(1 - p_a) \cdot \bar{z}_a^r$ the probability of failure of each path $r \in R$, computed as 1 minus the probability that the path r succeeds. We can first observe that:

Observation 1. *If* (1) – (5) *is feasible, then there always exists an optimal solution where no path r having $\bar{p}^r = 1$ is selected,*

as these may increase the solution cost without contributing to satisfy constraint (2). Therefore we assume w.l.o.g. that:

Remark 1. *Any arc having $p_a = 1$ can be removed from A.*

We also prove that:

Observation 2. *If an optimal solution exists, in which a path r with $\bar{p}^r = 0$ is selected, then also the solution selecting* only *r is optimal.*

In fact, by selecting a path having probability of failure equal to 0, any failure target $\mathcal{F}$ is satisfied. This does not imply that such a path always represents an optimal RPSP solution, as its cost can be higher than a suitable combination of other paths. It however follows that:

Corollary 1. *If a path r having $\bar{p}^r = 0$ is not representing an optimal solution, then no optimal solution exists in which such a path is selected.*

These observations lead to the following:

Theorem 1. *When $\mathcal{F} = 0$ any instance of RPSP can be solved in polynomial time.*

A simple proof is to solve a traditional Shortest Path Problem on a subgraph of G containing only the arcs $a \in A$ with probability of failure $p_a = 0$.

Therefore, since when $\mathcal{F} = 1$ the problem admits a trivial *null* solution, in the following we assume w.l.o.g. $0 < \mathcal{F} < 1$. We remark that, in this general case, the following negative result holds:

Theorem 2 ([18])**.** *When $0 < \mathcal{F} < 1$, even the problem of deciding if a RPSP solution is feasible is NP-complete.*

Proof. The proof directly follows from [18], observing that in our case two terminals are given ($k = 2$ in the notation of [18]): under this condition, on general graphs as ours, the problem of deciding if the failure target is respected becomes NP-Complete as soon as paths composed by more than two arcs are allowed, even if the graph was undirected. □

This also trivially implies that the whole RPSP, on general graphs, is NP-Hard.

However, Theorem 1 leads to the following:

Observation 3. *A path having $\bar{p}^r = 0$, if any, can be found in polynomial time.*

We therefore conclude by observing the following.

Observation 4. *Given an instance of RPSP, an optimal solution is the best between the path of minimum cost found on the subset of arcs having $p_a = 0$ as in Theorem 1, and the solution on a RPSP where paths having $\bar{p}^r = 0$ are forbidden.*

In the following, we then assume that all paths having $\bar{p}^r = 0$ and $\bar{p}^r = 1$ are excluded from set R. In fact, the only interesting member of the former class (that of minimum cost) can be found in polynomial time in a preprocessing phase, while the latter are never selected in an optimal solution. We remind to Section 4.3.1 for a description on how such a removal is done dynamically.

3. Modeling and solving the failure probability computation problem.

As reported in the previous section, the problem of deciding if a given subgraph encodes a solution satisfying the failure probability constraint is NP-Complete. Being a cornerstone of our methods, we therefore designed a methodology that is based on two ingredients: the availability of a path representation of a RPSP solution and the inference of a Bayesian Network (BN).

BNs are mathematical formulations that model stochastic processes and can be exploited to compute the probabilities of success and failure of complex systems [19]. A BN can be represented as a directed acyclic graph where each node corresponds to a random variable and each arc represents the dependency between two random variables. BNs are solved accordingly by conditional probability computations.

BN are certainly a convenient modeling tool, but unfortunately not a direct template for the design of computationally effective methods. In fact, their use in network optimization is complicated by two factors. First, the number of random variables (and therefore the size of the BN) may grow combinatorially with network size. It is indeed our case. We have three types of random variables: the main one models the failure of the whole connection; this is dependent on a set of random variables, each modeling the failure of a single path; these, in turn, are dependent on a set of random variables, each modeling the failure of single arcs. The latter ones are independent. Since the number of paths grows combinatorially in network size, so does the size of a BN.

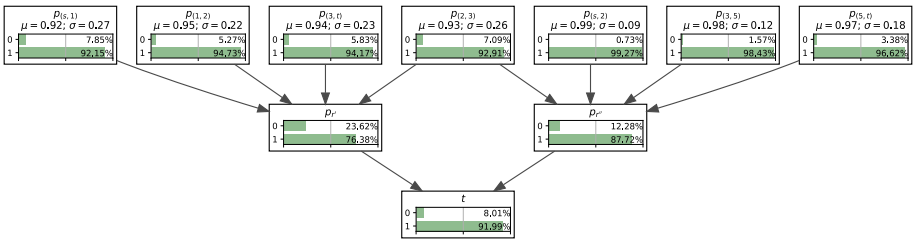

Figure 1: Example of a Bayesian Network for a solution with two paths r' and r'' sharing arc $(2, 3)$.

Second, the evaluation of the BN requires to compute tables, having one row for each possible combination of logical values to the independent events in the system. In our case, they are the failure (or not) of each single arc. Therefore, also the size of these tables grows combinatorially with the network size. We also remark that, in general, even the enumeration of paths in a graph requires computing effort.

However, in our case we can rely on features of our models which strongly limit these drawbacks. In particular, instead of statically creating a BN for the full network, we create BNs on the fly only for selected candidate solutions produced by a relaxation of our model, satisfying (3) – (5) and only a simplification of (2) (as detailed in Section 4). Additionally, these solutions already come decomposed in terms of selected paths: these are the only ones needing a corresponding random variable to be considered, and therefore requiring a node in the BNs. That is, even if the number of potential paths in the network can be huge, very few of them are actually selected (and ready available) in a candidate solution. On top of this simplification, fast processing techniques from the literature can be applied in our case.

Formally, given a set of selected paths $\bar{R}$ satisfying (3) – (5) and a relaxation of (2), we design a BN $\beta = (\Phi, \Omega)$ where Φ and Ω are the set of nodes and arcs of the BN, respectively. Set Φ has one node ϕ_a for each arc a traveled by any path $r \in \bar{R}$, one node ϕ_r for each path $r \in \bar{R}$, and one terminal node ϕ_t. Nodes ϕ_a, ϕ_r, and ϕ_t represent the random variables that describe the failure or the success of arcs, paths, and whole communication.

Only the ϕ_a nodes are unconditional nodes. Instead each ϕ_r node succeeds only if all the ϕ_a nodes corresponding to arcs in path r succeed. While node ϕ_t succeeds if at least one node ϕ_r succeeds.

An example of a BN representing a solution with two distinct paths $r' = (s, 1, 2, 3, t)$ and $r'' = (s, 2, 3, 5, t)$ is depicted in Figure 1: each random variable corresponding to an arc is given a probability of failure (corresponding to label 0) and a probability of success (corresponding to label 1). While the probability of failure of a random variable ϕ_r depends solely on the probabilities of the arcs of r, the probability of failure of the communication, represented by the node ϕ_t, also considers the fact that the two paths r' and r'' share arc $(2, 3)$.

Once the BN is created, we conduct inference on the node ϕ_t in order to

obtain the probability of failure and success of the communication. To obtain fast computations, we employ the preprocessing techniques of [20], together with lazy propagation (see Section 5).

4. Optimization algorithms

In order to solve the RPSP, we proceed as follows: we perform an integer relaxation of the RPSP, that we name Independent path RPSP (IRPSP), suitably relaxing the computation of $P(\mathbf{y})$ but keeping integrality conditions (Subsection 4.1). We solve a continuous relaxation of the IRPSP by means of column generation techniques, obtaining a dual bound that is valid also for RPSP (Subsection 4.3).

If the solution found by the continuous relaxation is fractional, we enter a search tree by performing branching to enforce integrality conditions; when all variables have integer values in the continuous relaxation, an integer IRPSP solution is found: we test its feasibility for the RPSP, checking constraint (2) by means of the BN introduced in Section 3, and accept it only if it passes the test. Otherwise, further branching is applied to impose the selection of additional paths to the solution (Subsection 4.4).

To strengthen our formulation we also introduce valid inequalities (Subsection 4.2) showing how they interact with the column generation pricing scheme.

Incumbent integer RPSP solutions are both retained when found through branching, and produced by primal heuristics (Subsection 4.5). They are used both for early pruning the search tree, and to perform reduction by variable fixing (Subsection 4.6).

4.1. Integer relaxation

As discussed, computing $P(\mathbf{y})$ is hard in the general case, because the paths in R could share arcs, and thus, although arcs failures are independent, paths failures are not. Therefore, we build our computational methods around the following relaxation:

Theorem 3. *When the failures of the paths are assumed to be independent, Inequality* (2) *becomes:*

$$\tilde{P}(\mathbf{y}) = \prod_{r \in R} \bar{p}^r \cdot y^r \leq \mathcal{F}. \tag{6}$$

Hence, replacing (2) *with* (6) *produces an integer relaxation of the RPSP.*

Proof. In fact, $\tilde{P}(\mathbf{y}) \leq P(\mathbf{y})$ for each $\mathbf{y}$, and therefore each solution which is feasible with respect to (2), is also feasible for (6). Besides matching the common understanding of joint probability of dependent and independent random events, it is possible to formally prove the statement by induction on the number of paths and the number of shared arcs. □

Indeed Equation (6) is nonlinear, but by means of logarithmic mapping and since y^r are binary variables, we obtain:

$$\sum_{r \in R} \log(\bar{p}^r) \cdot y^r \leq \log(\mathcal{F}). \tag{7}$$

Formally, we indicate as Independent paths RPSP (IRPSP) the relaxed model obtained by (1) – (5), replacing (2) with (7).

A strong point about our relaxation is that it keeps the encoding of *single path* failure probabilities unchanged. It is therefore much stronger than standard approaches like the use of Boole-Bonferroni inequalities (or union bounds) and similar approximations [16]. Finding tight bounds on generic mathematical programs is indeed a challenging research stream on its own [21].

4.2. Valid inequalities

To strengthen our formulation we include sets of valid inequalities that either increase the number of paths selected in a solution or the number of arcs selected.

First, let $\delta^+(i)$ (resp. $\delta^-(i)$) be the set of the outgoing arcs from i (resp. ingoing arcs to i), we can state that at least one arc outgoing the source node and one arc ingoing the destination node are selected in a feasible integer solution, and therefore

$$\sum_{a \in \delta^+(s)} x_a \geq 1 \tag{8}$$

and

$$\sum_{a \in \delta^-(t)} x_a \geq 1 \tag{9}$$

are valid inequalities

Then, let r^* be a feasible path minimizing the probability of failure. We use the value of the probability of failure p^{r^*} of such a path to compute the minimum number of paths $k_{\min}$ that must be selected in a solution in order to satisfy constraint (7).

Formally, let

$$k_{\min} = \lceil \log(\mathcal{F}) / \log(p^{r^*}) \rceil,$$

we have that

$$\sum_{r \in R} y^r \geq k_{min} \tag{10}$$

is a valid inequality.

Furthermore, we design a set of inequalities to reduce the gap between $\tilde{P}(y)$ and $P(y)$ by computing a more accurate probability of failure.

Let $B \subset A$ be an arbitrary subset of arcs and let $R_B = \{r \in R \mid \bar{z}_a^r = 1, \forall a \in B\}$ and $R_{\bar{B}} = R \setminus R_B$ be two partitions of set R, that are the subset of paths traveling all the arcs in B and the subset of paths missing at least one arc in B, respectively. It follows that if any arc in B fails, all the paths in

R_B fail too. Therefore, the probability of failure of the paths in R_B is at least $1 - \prod_{a \in B}(1 - \bar{p}^a)$. Therefore, when a subset B is found such that

$$1 - \prod_{a \in B}(1 - p_a) > \prod_{r \in R_B} \bar{p}^r$$

we have

$$\prod_{r \in R} \bar{p}^r = \prod_{r \in R_B} \bar{p}^r \prod_{r \in R_{\bar{B}}} \bar{p}^r < (1 - \prod_{a \in B}(1 - p_a)) \prod_{r \in R_{\bar{B}}} \bar{p}^r$$

thus the inequality

$$(1 - \prod_{a \in B}(1 - p_a)x_a) \prod_{r \in R_{\bar{B}}} \bar{p}_r \cdot y^r \leq \mathcal{F}$$

strengthen our formulation, possibly reducing the gap between $\tilde{P}(y)$ and $P(y)$.

In our setting, we found to be profitable to include all inequalities for $|B| = 1$, obtaining the following constraints

$$(1 - (1 - p_a)x_a) \prod_{r \in R | \bar{z}_a^r = 0} \bar{p}^r \cdot y^r \leq \mathcal{F}, \, \forall a \in A$$

that we linearize into

$$\log(p_a) \cdot x_a + \sum_{r \in R} (1 - \bar{z}_a^r) \log(\bar{p}^r) y^r \leq \log(\mathcal{F}), \, \forall a \in A. \tag{11}$$

4.3. Solving the continuous relaxation of IRPSP

Concerning model (1) – (5), we observe that the set R grows exponentially in the size of the graph. We recur to column (and row) generation techniques to solve the continuous relaxation of the IRPSP (C-IRPSP). We refer to the model containing the full set of columns and rows as *Master Problem (MP)*; in order to optimize it we iteratively solve a *Restricted MP (RMP)* involving only a small set of columns $\bar{R} \subseteq R$, and only the constraints of the set (3) related to the elements of $\bar{R}$. Given a RMP optimal dual solution, we search for negative reduced cost columns, which are in the MP but not in the RMP, by solving a *pricing problem*. If no such column is found, the optimal RMP solution is optimal for the C-IRPSP as well. The corresponding value is a valid lower bound for the IRPSP and as a consequence for the RPSP. Otherwise we iterate.

In particular we relax the integrality on y^r variables with non-negativity conditions only, and the integrality on x_a variables by both non-negativity conditions and variable upper bounds of value 1. An upper bound of value 1 to the value of y^r variables is therefore imposed implicitly by means of constraints (3). In fact, while these variable upper bounds are often disregarded in column generation algorithms, in our case the following holds.

Theorem 4. *When the graph is connected, by dropping variable upper bounds, an optimal solution to the continuous relaxation of* (1), (7), (3) – (5) *always exists, in which only a single path having best ratio between cost and failure probability is selected (possibly in multiple copies).*

Proof. Due to Constraints (3) and the sense of the objective function, we have that in an optimal solution $x_a = \sum_{r \in R} \bar{z}_a^r \cdot y^r$. Therefore, the objective function (1) can be rewritten as

$$\min \sum_{r \in R} \left(\sum_{a \in A} c_a \cdot \bar{z}_a^r \right) \cdot y^r. \tag{12}$$

Since constraints (4) and (5) are relaxed, only Constraint (7) remains: the problem turns out to be a continuous unbounded knapsack, whose optimal solution is known to have the structure of our claim. □

In a round of preliminary explorations, we experimented variants of model (1) – (5), but all those neglecting such variable upper bounds yielded very poor relaxations. On the other hand, constraints (3) are *exponential in number* and need to be managed implicitly and dynamically, as we do with the columns. We handle them with a special procedure, which intertwines with the pricing algorithm, discussed in the following.

We initialize the RMP by including both the path of minimum cost and the path with smaller probability of failure in $\bar{R}$.

4.3.1. Pricing problem.

Let $\mu_{ar} \leq 0$, $\eta \leq 0$, $\nu \geq 0$, and $\pi_a \leq 0$ be the dual variables corresponding to constraints (3), (7), (10), and (11) respectively.

The reduced cost of a column r is:

$$\chi_r = -\sum_{a \in A} \mu_{ar} \cdot z_a - \nu - \sum_{a \in A} \pi_a \cdot (1 - z_a) \cdot p - \eta \cdot p \tag{13}$$

The pricing problem can be stated as follows:

$$
\begin{aligned}
\min \quad & \chi_r \\
\text{s.t.} \quad & p = \log(1 - \prod_{a \in A} (1 - p_a) \cdot z_a) && && (14)\\
& \sum_{a \in \delta^+(i)} z_a = \sum_{a \in \delta^-(i)} z_a = v_i && \forall i \in N \setminus \{s,t\} && (15)\\
& \sum_{a \in \delta^+(s)} z_a = \sum_{a \in \delta^-(t)} z_a = 1 && && (16)\\
& \sum_{a \in \delta^+(S)} z_a \geq v_i && \forall S \subseteq N, |S| > 1, k \in S && (17)\\
& p \leq 0 && && (18)\\
& v_i \in \mathbb{B} && \forall i \in N && \\
& z_a \in \mathbb{B} && \forall a \in A &&
\end{aligned}
$$

where variable p is the logarithm of the probability of failure of the path, each variable v_i is set to 1 if node i is selected in the path, and 0 otherwise, each variable z_a is set to 1 if arc a is selected, and 0 otherwise.

We remind that according to Observation 4 we can assume that p is always well defined.

Constraint (14) accounts the failure probability of the path, while constraints (15) – (17) ensure that a path from s to t is selected.

The objective function (13) minimizes the difference between costs and profits. Besides a fix profit $-\nu$, we gain a profit η that is proportional to the logarithm of the probability of failure of the path, and for each arc a that is not selected in the solution we also gain a profit π_a still proportional to the probability of failure of the path. Furthermore, we pay a cost $-\mu_{ar}$ to select arc a. However, if $r \notin \bar{R}$, constraints (3) cannot be binding. In fact, they do not even appear in the RMP. Therefore we implicitly have $\mu_{ar} = 0$. That is, we incur the cost $-\mu_{ar}$ *only if* the path r defined by variables z_a is already in the RMP; furthermore in such a case we know that $\chi_r \geq 0$, since the RMP is solved to optimality.

4.3.2. Pricing algorithm.

To solve our pricing problem we devised a dynamic programming algorithm: let $l = (i, c, p, \rho)$ be a label defining the partial reduced cost c and the probability p of a partial path $\rho = (s, \ldots, i)$. Our algorithm is designed as follows.

Initialization. We start by creating a single label $l = (s, g_{cut}, p, \rho)$, representing a partial path $\rho = (s)$ starting from source node s, where $g_{cut} = -\sum_{a \in A} \pi_a$ is the profit gained when arcs are avoided in the solution, and $p = 1$ is the probability of success of the partial path. We then push label l in a queue of labels.

Extension. At each iteration we select a label $l = (i, g_{cut}, p, \rho)$ from the queue in a LIFO way and create a new label $l' = (j, g_{cut}, p', \rho')$ for each arc $a = (i, j)$ outgoing from i to j, such that:

- we add the node j to the partial path, that is $\rho' = (s, \sigma_1, \sigma_2, \ldots, i, j)$;
- we decrease the profit g_{cut} because of the selected arc a, that is $g'_{cut} = g_{cut} + \pi_a$;
- we update the probability of success of the partial path $p' = p \cdot (1 - p_a)$.

Dominance. For each new label $l = (i, g_{cut}, p, \rho)$ we perform a dominance check.

If it exists a label $l' = (i, g'_{cut}, p', \rho')$ having $p' > 0$ in the queue of labels with:

(a) higher profit of non selected arcs, that is $g_{cut} \cdot \log(1-p) \leq g'_{cut} \cdot \log(1-p')$,

(b) smaller probability of failure $1 - p \geq 1 - p'$,

(c) smaller partial reduced cost, and at least one of these conditions is strict, then label l cannot lead to an optimal solution, and therefore can be deleted. Otherwise, we add l to the labels queue.

The condition $p' > 0$ is enforced to guarantee that no path is dominated by another one eventually leading to $\bar{p}^r = 0$ at termination.

Checking condition (c), however, needs special care. In fact, due to the contribution of $-\mu_{ar}$ dual variables, the actual reduced cost depends also on the possible belonging of the final path to $\bar{R}$ or not. Therefore, for each $r \in \bar{R}$, we conceptually define as $M_r = \sum_{a \in A} \bar{z}_a \cdot \mu_{ar}$ the contribution of constraints (3) to the reduced cost of path r. Then we define M_ρ as the minimum among M_r values such that the partial path ρ is a sub-path of r. Therefore, $-M_\rho$ is the maximum reduced cost worsening that we might incur if, by extending ρ, we end up generating a path already in $\bar{R}$.

We perform the following relaxed check of the dominance condition on cost:

$$g_{cut} \cdot \log(1-p) - \eta \cdot \log(1-p) \geq g'_{cut} \cdot \log(1-p') - \eta \cdot \log(1-p') - M_{\rho'}$$

that is, we dominate l only if another label l' is found that, although possibly leading to a path already in $\bar{R}$, would produce a non-worse reduced cost.

Algorithmically speaking, one of the key elements of our label correcting algorithm is actually how to manage the set of generated paths: in order to compute M_r and M_ρ costs efficiently we designed a prefix tree which is detailed in the Appendix A.

Stopping criterion. When the labels queue is empty, we stop and select the first label having minimum reduced cost among the ones having $p > 0$. If it is non-negative, we stop column generation. Otherwise we enrich the RMP with that single column, and the corresponding constraints of the set (3), and iterate.

As a technical remark, according to our dominance rule the best path with $p > 0$ is never dominated. Therefore in this way we enforce the removal of paths having $\bar{p}^r = 0$ from $\bar{R}$, matching the assumption of Observation 4, without loss of optimality guarantees.

4.4. Branching strategy

We remark that, the continuous relaxation lower bound found through column generation is a valid dual bound for both IRPSP and RPSP. When an optimal solution to C-IRPSP is found, which is fractional and whose value is lower than a RPSP primal bound, we enter a search tree. We designed three branching strategies: the first is meant to speed up the reduction of the integrality gap, the second drives towards integrality on variables x_a, while the latter is used only to ensure that integer solutions satisfy failure target constraint (2).

Branching on number of paths. In our first branching strategy we force the sum of the path variables to be integer. Let $\tilde{y}^r$ be the value of variable y^r in a solution of C-IRPSP which is fractional. If the number of selected paths, that is $\sum_{r\in\bar{R}} \tilde{y}^r$, is fractional we create two branching nodes, the first with an additional constraint

$$\sum_{r\in\bar{R}} y^r \leq \left\lfloor \sum_{r\in\bar{R}} \tilde{y}^r \right\rfloor \tag{19}$$

and the second one with constraint

$$\sum_{r\in\bar{R}} y^r \geq \left\lceil \sum_{r\in\bar{R}} \tilde{y}^r \right\rceil. \tag{20}$$

Both constraints origin new dual variables that are, respectively a fixed cost or a fixed profit for any additional priced path. In neither case the structure of our pricing problem is changed.

Branching on arcs. When the sum of path variables is integer, we proceed by fixing arc variables to integer values: we search for the most fractional arc $\hat{a}$, that is $\hat{a} \in \operatorname{argmin}_{a\in A}\{|x_a - 0.5|\}$, and we create two new branching nodes: one having $x_{\hat{a}} = 1$ and one where $x_{\hat{a}} = 0$.

In both nodes we add a constraint to the MP, respectively:

$$\sum_{r\in\bar{R}} \bar{z}_a^r \cdot y^r \geq 1 \tag{21}$$

and

$$\sum_{r\in\bar{R}} \bar{z}_a^r \cdot y^r \leq 0. \tag{22}$$

When we set $x_{\hat{a}} = 0$, each column having $z_a^r = 1$ is removed from the RMP, arc $\hat{a}$ is removed from the graph, and the pricing problem remains unchanged. Instead, when we set $x_{\hat{a}} = 1$ and add the corresponding constraint in the MP, such a constraint induces a new dual variable in the pricing problem that is a profit for selecting arc $\hat{a}$, changing the structure of pricing.

Therefore we modify our dynamic programming algorithm as follows: in the dominance phase of our algorithm, we add a new condition that is:

(d) a label l is dominated by a label l' only if the partial path ρ has visited all nodes in ρ'.

In fact, this condition excludes the case where l is dominated but still be profitable by collecting the additional profit of an arc $\hat{a}$. A formal proof is given in the Appendix B.

Branching for failure probability. When an integer solution is found, both in terms of path and arc variables, the values of x_a variables define a subgraph $\tilde{G} = (N, \tilde{A})$ that connects s to t and satisfies constraint (6), and y^r variables define a corresponding path representation. It is indeed a feasible IRPSP solution, which is also an optimal IRPSP solution for the particular branching node subproblem. However, when solving the RPSP we must perform an additional check to ensure that the actual probability of failure of the selected paths is lower than $\mathcal{F}$, satisfying also constraint (2).

Such a check is performed by BN computations, as detailed in Section 3.

If the failure probability BN check succeeds, the solution found is feasible also for the RPSP (and optimal for the particular branching node subproblem as well): we update the value of the RPSP primal bound (and keep the solution as incumbent), if profitable, we close the corresponding branching node and we backtrack.

On the contrary, the failure of the BN check is unfortunately not enough to mark the branching node as infeasible and backtrack. A clear example may be given by any RPSP instance whose C-IRPSP solution at the root node is automatically integer, and therefore feasible and optimal for the IRPSP, but infeasible for the RPSP. In such a case we are not allowed to stop branching, since we have no guarantee that, by forcing either additional paths or arcs variables to be selected, no feasible RPSP solution can be found. However, the IRPSP and the C-IRPSP as a consequence, is blind with respect to both possibilities.

Therefore, we may fall in one of these two cases: either the subgraph selected in such an integer C-IRPSP solution is not part of any optimal RPSP solution, or it belongs to an optimal solution but additional paths are required. We generate two children branching nodes accordingly.

In the first child we impose that at least one of the arcs of the selected subgraph has its x_a variable set to 0, that is

$$\sum_{a \in \tilde{A}} x_a \leq |\tilde{A}| - 1. \tag{23}$$

In the second child we set all variables $x_a = 1$ for $a \in \tilde{A}$, and we increase the number of selected paths, that is

$$\sum_{r \in \bar{R}} y^r \geq \left\lceil \sum_{r \in \bar{R}} \tilde{y}^r \right\rceil + 1. \tag{24}$$

We remark that increasing the number of selected paths does not necessarily increase the cost of a solution: in fact, the selected paths may be a subset of all the paths that can be generated from the selected subgraph.

4.5. Primal heuristics

To speed up the overall branch-and-bound procedure we design two primal heuristics, one greedy heuristic that is run once before the start of the overall branch-and-price framework, and one rounding heuristic run at the end of the evaluation of each branching node.

Independent path heuristic. Our first heuristic is run once and iteratively searches for a new path connecting the source to the destination in such a way that no arc of the new path has been traveled before. In other words, our heuristic searches for a set of independent paths.

Our heuristic works as follows: (a) we find a minimum cost path from s to t and we add such a path to a set $\tilde{R}$ of minimum cost paths; (b) if no path can be found, the heuristic stops without any feasible solution; (c) if the paths contained in $\tilde{R}$ satisfy constraint (2), we stop with a feasible solution; (d) otherwise all the arcs belonging to the paths in $\tilde{R}$ are removed from the graph, and procedure restarts from step (a).

We remark that any solution found by such heuristic is feasible for both IRPSP and RPSP: in fact, because no arc is shared between the selected paths, those paths are independent, and the probability of failure is not greater than $\mathcal{F}$ also for RPSP.

The philosophy of this heuristic is to be aggressive on cost while producing a feasible solution for the IRPSP. As a weakness it might be fragile w.r.t. the failure constraint, especially on poorly connected graphs.

Knapsack heuristic. The second primal heuristic is run at the end of the column generation process that solves the C-IRPSP of each node of the branching tree.

Given the values of y^r variables of a fractional solution of the C-IRPSP , we select the subset $\tilde{R} \subseteq \bar{R}$ of paths having positive y^r variables, that is $\tilde{R} = \{r \in \bar{R} \mid y^r > 0\}$. Then we solve the following optimization problem aiming to select a subset of $\tilde{R}$ such that the failure constraint (7) is satisfied at minimum cost:

$$\min \sum_{r \in \tilde{R}} \bar{c}^r \cdot w_r \tag{25}$$

$$\text{s.t.} \sum_{r \in \tilde{R}} \log(\bar{p}^r) \cdot w_r \leq \log(\mathcal{F}) \tag{26}$$

$$w_r \in \mathbb{B} \tag{27}$$

where each w_r variable is set to 1 if path r is selected, 0 otherwise. However, we can reformulate the problem in terms of selecting those paths that should not be part of a solution: in fact, the objective function (25) can be reformulated as

$$\sum_{r \in \tilde{R}} \bar{c}^r - \max \sum_{r \in \tilde{R}} \bar{c}^r \cdot (1 - w_r) \tag{28}$$

while constraint (26) can be reformulated as follows:

$$\sum_{r\in\tilde{R}} \log(\bar{p}^r)\cdot w_r \leq \log(\mathcal{F}) =$$

$$\sum_{r\in\tilde{R}} \log(\bar{p}^r)\cdot w_r - \sum_{r\in\tilde{R}} \log(\bar{p}^r) \leq \log(\mathcal{F}) - \sum_{r\in\tilde{R}} \log(\bar{p}^r) =$$

$$\sum_{r\in\tilde{R}} -\log(\bar{p}^r)\cdot(1-w_r) \leq \log(\mathcal{F}) - \sum_{r\in\tilde{R}} \log(\bar{p}^r) \quad (29)$$

Therefore, to select a subset of paths that satisfy constraint (7), we solve a binary knapsack problem where we have an item for each path in $\tilde{R}$, each item has a non-negative profit $\bar{c}^r$ and a non-negative weight $\lceil -\log(\bar{p}^r)\rceil$, and the capacity of the knapsack is set to $\lfloor \log(\mathcal{F}) - \sum_{r\in\tilde{R}} \log(\bar{p}^r)\rfloor$. Since we always start from a feasible fractional solution of the C-IRPSP , we also have that the capacity of such a knapsack is always non-negative.

4.6. Reduction techniques

Within the column generation procedure, we found to be profitable to exploit the value of a valid RPSP primal bound to reduce the size of our problem. In fact, if we are given a primal bound to the value of an optimal RPSP solution, we can discard all paths in R having a cost greater than such primal bound, since if they were selected they would provide a more expensive solution.

Therefore, in our methodology we make use of such consideration as follows.

Arc variable fixing. In a preliminary phase of our algorithm we compute each path of minimum cost from s to t passing through an arc $a \in A$.

Every time a feasible solution is found and the RPSP primal bound is improved, we then fix all arc variables x_a corresponding to arcs having a minimum cost path greater than the primal bound value, as passing though that arc would lead to a more expensive subgraph.

Expensive path avoidance. We further exploit the value of a RPSP primal bound to avoid the generation of paths with higher cost in the pricing procedure. In fact, in a feasible integer solution, those paths that cost more than a feasible solution are not selected. To achieve such behavior we include an additional constraint

$$\sum_{a\in A} c_a \cdot z_a \leq \mathcal{UB} \quad (30)$$

in our pricing problem, where $\mathcal{UB}$ is the value of a primal bound.

Our dynamic programming algorithm is then modified as follows: a label l also includes the cost of the partial path considering arcs costs only. Such a cost is then taken into account when testing the dominance between labels, and the pricing procedure avoid the extension of labels that have a cost higher than $\mathcal{UB}$.

Also, whenever we find an improving primal bound, we fix y^r variables already in the RMP that correspond to paths with a cost higher than such a primal bound.

5. Computational results

We implemented our algorithms in C++ using SCIP framework [22], while LP subproblems were solved using the simplex algorithm implemented in CPLEX 12.6 [23]. In SCIP we kept the default settings although we set single thread execution. The resolution of the BNs is carried out using the framework of [20]. It automatically performs BN minimalization by introducing intermediate nodes by need, before applying lazy propagation algorithms.

We produced a randomly generated benchmark consisting of graphs where each node is a point whose coordinates are randomly drawn in the range $(-100, 100)$.

Each pair of nodes is connected with a probability γ. If an arc a exists, its cost c_a is the euclidean distance between its endpoints, and its probability of failure p_a is chosen uniformly at random as one of the values in the set $\{5 \cdot 10^{-3}, 10^{-3}, 10^{-4}, 10^{-5}\}$. We generated instances for different sizes of the graph ($|N| \in \{20, 30\}$) and different density ($\gamma \in \{0.25, 0.50, 0.75\}$). For each pair of size and density, we generated 10 instances, thereby producing 60 instance graphs with costs and probabilities on arcs overall.

Also, for each instance graph we perform several runs where we set a different failure probability target $\mathcal{F} \in \{1 \cdot 10^{-1}, 5 \cdot 10^{-2}, 1 \cdot 10^{-2}, 5 \cdot 10^{-3}, 1 \cdot 10^{-3}, 5 \cdot 10^{-4}, 1 \cdot 10^{-4}, 5 \cdot 10^{-5}, 1 \cdot 10^{-5}, 5 \cdot 10^{-6}, 1 \cdot 10^{-6}\}$, for a total of 660 RPSP instances.

All our tests have been conducted on a PC equipped with an AMD Ryzen Threadripper 1950X CPU at 3400 MHz and 32GB of memory and setting a time limit of one hour of computing time for each run.

We do not report algorithm profiling, but we mention that the component requiring the largest share of computing time is by far the resolution of LP subproblems. Indeed, we experimented on several settings for LP resolution algorithms, without substantial improvements. Instead all our custom algorithmic components, when taken independently, run extremely fast. Also the computing time spent in pricing new reduced cost variables is on average smaller by one order of magnitude w.r.t the time spent in solving RMPs.

5.1. Strength of the IRPSP relaxation

At first, we evaluate how tight is the dual bound produced by the IRPSP integer relaxation. We designed the experiment as follows. Branching for failure probability was deactivated: each integer solution found during the exploration of the search tree was retained. Upon termination, the best among them represents an optimal *IRPSP* solution. That might in general be super-optimal for the RPSP; nevertheless, it represents a valid dual bound for the RPSP. In Tables 1 and 2 we report, for networks of 20 and 30 nodes respectively, results as average values over classes of 10 instances having same density γ and same target failure probability $\mathcal{F}$, as indicated in the first two columns. The analysis

Table 1: Computational effort for IRPSP instances with 20 nodes.

γ	$\mathcal{F}$	# s.	dual. gap (%)	br. nodes	time (s)	$\mathcal{F}$ gap (%)	RSPP opt. gap (%)
0.25	$1 \cdot 10^{-1}$	10	-	1.0	0	0.0	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	0.0	0.0
	$1 \cdot 10^{-2}$	10	-	1.6	0	0.0	0.0
	$5 \cdot 10^{-3}$	10	-	4.3	0	1.1e+01	-2.6
	$1 \cdot 10^{-3}$	10	-	19.2	0	2.9e+02	-7.5
	$5 \cdot 10^{-4}$	10	-	18.8	0	6.4e+02	-7.9
	$1 \cdot 10^{-4}$	10	-	18.4	0	3.8e+03	-28.6
	$5 \cdot 10^{-5}$	10	-	73.4	0	3.2e+03	-21.7
	$1 \cdot 10^{-5}$	10	-	243.1	3	5.8e+03	-29.0
	$5 \cdot 10^{-6}$	10	-	276.7	2	3.9e+04	-43.3
	$1 \cdot 10^{-6}$	10	-	758.4	25	1.7e+05	-45.5
0.50	$1 \cdot 10^{-1}$	10	-	1.0	0	0.0	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	0.0	0.0
	$1 \cdot 10^{-2}$	10	-	1.0	0	0.0	0.0
	$5 \cdot 10^{-3}$	10	-	8.1	0	2.0e-01	-2.1
	$1 \cdot 10^{-3}$	10	-	5.0	0	4.0e+02	-3.8
	$5 \cdot 10^{-4}$	10	-	10.0	0	9.0e+02	-3.5
	$1 \cdot 10^{-4}$	10	-	21.4	0	4.2e+03	-10.8
	$5 \cdot 10^{-5}$	10	-	61.5	0	5.6e+03	-9.1
	$1 \cdot 10^{-5}$	10	-	90.3	133	1.1e+04	-21.0
	$5 \cdot 10^{-6}$	10	-	125.9	34	2.1e+04	-22.0
	$1 \cdot 10^{-6}$	10	-	79.6	95	6.4e+04	-22.9
0.75	$1 \cdot 10^{-1}$	10	-	1.0	0	0.0	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	0.0	0.0
	$1 \cdot 10^{-2}$	10	-	1.0	0	0.0	0.0
	$5 \cdot 10^{-3}$	10	-	1.6	0	0.0	0.0
	$1 \cdot 10^{-3}$	10	-	6.8	0	4.0e+02	-0.1
	$5 \cdot 10^{-4}$	10	-	6.0	0	9.0e+02	-0.1
	$1 \cdot 10^{-4}$	10	-	26.3	1	2.7e+03	-9.8
	$5 \cdot 10^{-5}$	10	-	37.5	1	2.8e+03	-12.7
	$1 \cdot 10^{-5}$	10	-	41.5	1	3.0e+03	-21.0
	$5 \cdot 10^{-6}$	10	-	2985.7	116	6.8e+03	-18.6
	$1 \cdot 10^{-6}$	10	-	3230.4	98	7.7e+04	-20.0

in this subsection complements further (although preliminary) results concerning the general behavior of column generation for the IRPSP case, which have been presented in [24].

In columns 3 to 6 we report the number of instances terminating within the time limit of one hour, the average IRPSP optimality gap left open on the instances that did not terminate (expressed as relative value w.r.t. IRPSP dual bound), the average number of explored branching nodes and the average computing time on the terminating instances. In all but a few cases the IRPSP could be solved to proven optimality within the time limit. On the terminating instances, computing times are in the range of few minutes. Unfortunately, on the instances hitting the timeout the optimality gap remains large. This suggests that instances with specific structures may be harder than others, although our investigations did not highlight which is the critical structure. The expected computing effort increases both as γ increases and as $\mathcal{F}$ decreases; that is, dense instances with low failure target probability are harder.

Table 2: Computational effort for IRPSP instances with 30 nodes.

γ	$\mathcal{F}$	# s.	dual. gap (%)	br. nodes	time (s)	$\mathcal{F}$ gap (%)	RSPP opt. gap (%)
0.25	$1 \cdot 10^{-1}$	10	-	1.0	0	-	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	-	0.0
	$1 \cdot 10^{-2}$	10	-	1.0	0	-	0.0
	$5 \cdot 10^{-3}$	10	-	1.0	0	-	0.0
	$1 \cdot 10^{-3}$	10	-	2.6	0	-	0.0
	$5 \cdot 10^{-4}$	10	-	2.6	0	-	0.0
	$1 \cdot 10^{-4}$	10	-	27.1	4	1.0e+03	-4.0
	$5 \cdot 10^{-5}$	10	-	50.7	0	2.2e+03	-4.9
	$1 \cdot 10^{-5}$	10	-	95.9	1	5.0e+03	-24.0
	$5 \cdot 10^{-6}$	10	-	173.6	2	1.2e+04	-13.5
	$1 \cdot 10^{-6}$	10	-	286.4	6	2.9e+04	-13.9
0.50	$1 \cdot 10^{-1}$	10	-	1.0	0	-	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	-	0.0
	$1 \cdot 10^{-2}$	10	-	1.0	0	-	0.0
	$5 \cdot 10^{-3}$	10	-	1.4	0	-	0.0
	$1 \cdot 10^{-3}$	10	-	2.2	0	-	0.0
	$5 \cdot 10^{-4}$	10	-	2.0	0	-	0.0
	$1 \cdot 10^{-4}$	10	-	12.6	0	-	0.0
	$5 \cdot 10^{-5}$	10	-	122.0	1	3.9e+03	-0.1
	$1 \cdot 10^{-5}$	9	51.3	285.6	11	1.4e+02	-19.0
	$5 \cdot 10^{-6}$	10	-	601.4	21	5.4e+02	-24.0
	$1 \cdot 10^{-6}$	8	50.4	874.4	91	1.4e+04	-17.8
0.75	$1 \cdot 10^{-1}$	10	-	1.0	0	-	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	-	0.0
	$1 \cdot 10^{-2}$	10	-	1.0	0	-	0.0
	$5 \cdot 10^{-3}$	10	-	1.0	0	-	0.0
	$1 \cdot 10^{-3}$	10	-	3.4	0	-	0.0
	$5 \cdot 10^{-4}$	10	-	3.4	0	-	0.0
	$1 \cdot 10^{-4}$	10	-	820.6	31	-	0.0
	$5 \cdot 10^{-5}$	10	-	820.2	33	1.0e+03	-0.8
	$1 \cdot 10^{-5}$	9	38.0	1332.9	368	3.8e+02	-18.0
	$5 \cdot 10^{-6}$	8	21.0	2771.0	16	1.5e+03	-22.2
	$1 \cdot 10^{-6}$	8	35.9	1127.8	41	4.5e+03	-20.4

In column 7 we report the average difference between the actual failure probability of the optimal IRPSP solution (or best known in case of timeout), when measured by BN computations, and $\mathcal{F}$ (expressed as relative value w.r.t. $\mathcal{F}$). In column 8 we report the difference between the same IRPSP solution and the best known RPSP solution (which is usually the global RPSP optimum, as found by our algorithms, see subsection 5.2), expressed as relative values w.r.t. the best known RPSP solution value. Both values are remarkably tight when $\mathcal{F}$ is high, but worsen quickly as $\mathcal{F}$ decreases.

Overall, we conclude that our algorithms are indeed effective for the IRPSP. Although being a pertinent ingredient, however, our experiments verify that IRPSP solutions require to be embedded with further components to be successful in RPSP algorithms: IRPSP bounds are in general not particularly tight at the root node, but can be improved quickly by branching and reduction.

Table 3: Computational effort for RPSP instances with 20 nodes.

γ	$\mathcal{F}$	# s.	dual. gap (%)	br. nodes	time (s)	BN nodes
0.25	$1 \cdot 10^{-1}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$1 \cdot 10^{-2}$	10	-	1.6	0	0.0
	$5 \cdot 10^{-3}$	10	-	8.5	0	0.8
	$1 \cdot 10^{-3}$	10	-	291.5	1	0.8
	$5 \cdot 10^{-4}$	10	-	337.9	4	0.0
	$1 \cdot 10^{-4}$	8	43.8	6634.6	2	10.4
	$5 \cdot 10^{-5}$	9	11.6	7930.3	137	0.0
	$1 \cdot 10^{-5}$	8	47.9	9357.0	18	2.3
	$5 \cdot 10^{-6}$	5	67.9	13823.5	285	2.5
	$1 \cdot 10^{-6}$	3	59.6	17148.1	657	1.8
0.50	$1 \cdot 10^{-1}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$1 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-3}$	10	-	210.3	1	5.9
	$1 \cdot 10^{-3}$	10	-	159.0	1	0.2
	$5 \cdot 10^{-4}$	10	-	129.8	1	0.0
	$1 \cdot 10^{-4}$	10	-	205.0	1	1.5
	$5 \cdot 10^{-5}$	10	-	539.2	4	0.0
	$1 \cdot 10^{-5}$	8	52.7	16054.4	510	2.5
	$5 \cdot 10^{-6}$	8	52.5	16364.5	521	0.1
	$1 \cdot 10^{-6}$	8	53.9	17542.9	533	1.1
0.75	$1 \cdot 10^{-1}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$1 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-3}$	10	-	1.6	0	0.0
	$1 \cdot 10^{-3}$	10	-	6.8	0	0.0
	$5 \cdot 10^{-4}$	10	-	6.2	0	0.0
	$1 \cdot 10^{-4}$	9	50.1	1104.4	180	1.0
	$5 \cdot 10^{-5}$	8	46.5	2728.4	52	0.0
	$1 \cdot 10^{-5}$	7	50.5	3280.9	204	1.3
	$5 \cdot 10^{-6}$	7	51.2	3114.2	179	0.0
	$1 \cdot 10^{-6}$	6	53.6	2069.5	64	0.1

5.2. Solving RPSP with optimality guarantees

As second experiment, we analyzed the behavior of our branch-and-price algorithm in solving the full RPSP to proven optimality. As before, tables 3 and 4 report, for networks of 20 and 30 nodes respectively, results as average values over classes of 10 instances having same density γ and same target failure probability $\mathcal{F}$, as indicated in the first two columns.

In columns 3 to 6 we report in turn the number of instances terminating within the time limit, the average optimality gap left open on the instances that did not terminate (expressed as relative value w.r.t. the dual bound), the average number of explored branching nodes and the average computing time on the terminating instances. Our algorithms can consistently solve instances to proven optimality. Similarly to the previous experiment, each test either produced proven optimal solutions within minutes, or hit the timeout leaving non negligible gaps. The parameter which has highest impact on the computational behavior is the failure probability target $\mathcal{F}$. Neither graph density nor size seems to be so correlated to the computational effort required to solve

Table 4: Computational effort for RPSP instances with 30 nodes.

γ	$\mathcal{F}$	# s.	dual. gap (%)	br. nodes	time (s)	BN nodes
0.25	$1 \cdot 10^{-1}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$1 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-3}$	10	-	1.0	0	0.0
	$1 \cdot 10^{-3}$	10	-	2.6	0	0.0
	$5 \cdot 10^{-4}$	10	-	2.6	0	0.0
	$1 \cdot 10^{-4}$	10	-	776.1	16	1.0
	$5 \cdot 10^{-5}$	10	-	882.6	12	0.0
	$1 \cdot 10^{-5}$	8	77.6	6527.1	309	4.0
	$5 \cdot 10^{-6}$	9	54.5	6217.3	262	0.0
	$1 \cdot 10^{-6}$	8	55.7	3494.4	59	0.0
0.50	$1 \cdot 10^{-1}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$1 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-3}$	10	-	1.4	0	0.0
	$1 \cdot 10^{-3}$	10	-	2.2	0	0.0
	$5 \cdot 10^{-4}$	10	-	2.0	0	0.0
	$1 \cdot 10^{-4}$	10	-	12.6	0	0.0
	$5 \cdot 10^{-5}$	10	-	130.4	2	0.0
	$1 \cdot 10^{-5}$	6	51.0	1414.3	53	4.2
	$5 \cdot 10^{-6}$	5	55.3	1378.4	64	0.0
	$1 \cdot 10^{-6}$	5	57.3	1249.0	51	0.0
0.75	$1 \cdot 10^{-1}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$1 \cdot 10^{-2}$	10	-	1.0	0	0.0
	$5 \cdot 10^{-3}$	10	-	1.0	0	0.0
	$1 \cdot 10^{-3}$	10	-	3.4	0	0.0
	$5 \cdot 10^{-4}$	10	-	3.4	0	0.0
	$1 \cdot 10^{-4}$	10	-	901.4	40	0.0
	$5 \cdot 10^{-5}$	10	-	949.0	41	0.0
	$1 \cdot 10^{-5}$	6	50.2	2178.7	159	3.2
	$5 \cdot 10^{-6}$	5	51.6	2108.3	201	0.0
	$1 \cdot 10^{-6}$	4	45.4	3891.1	22	0.0

RPSP. In column 7 we report the average number of times in which branching for failure probability was called, that is the average number of BN evaluations per instance. Such values are usually very low; that is, normally the first two branching rules are enough to both improve the dual bounds and allow primal bounding heuristics to find optimal RPSP solutions.

5.3. Comparing to standard protection schemes

As third experiment we tried to evaluate the impact in network optimization when using RPSP models instead of standard path protection schemes from the literature.

We consider as benchmark methods k-shortest path models, whose philosophy is to find a set of k arc disjoint paths between s and t, whose sum of costs is minimum [1]. For fixed k, optimal k-shortest path solutions can be found in polynomial time by simple min-cost flow computations on a graph identical to G, having capacity 1 on each arc, and having s (resp. t) as source (resp. sink) of k units of flow.

Table 5: Comparison of path protection schemes on instances with 20 nodes.

γ	$\mathcal{F}$	2-shortest path				3-shortest path			
		# inf	$\mathcal{F}$ gap (%)	# target missed	RSPP opt. gap (%)	# inf	$\mathcal{F}$ gap (%)	# target missed	RSPP opt. gap (%)
0.25	$1 \cdot 10^{-1}$	2	-	0	192.5	5	-	0	370.8
	$5 \cdot 10^{-2}$	2	-	0	192.5	5	-	0	370.8
	$1 \cdot 10^{-2}$	2	-	0	190.4	5	-	0	365.5
	$5 \cdot 10^{-3}$	2	-	0	167.9	5	-	0	307.7
	$1 \cdot 10^{-3}$	2	-	0	75.2	5	-	0	211.5
	$5 \cdot 10^{-4}$	2	-	0	75.2	5	-	0	211.5
	$1 \cdot 10^{-4}$	2	-	0	29.0	5	-	0	128.8
	$5 \cdot 10^{-5}$	2	-	0	29.0	5	-	0	128.8
	$1 \cdot 10^{-5}$	2	9.9e+01	5	2.5	5	-	0	74.2
	$5 \cdot 10^{-6}$	2	2.5e+02	6	-21.2	5	-	0	44.1
	$1 \cdot 10^{-6}$	2	1.7e+03	6	-27.3	5	-	0	59.7
0.50	$1 \cdot 10^{-1}$	0	-	0	124.9	0	-	0	275.3
	$5 \cdot 10^{-2}$	0	-	0	124.9	0	-	0	275.3
	$1 \cdot 10^{-2}$	0	-	0	124.9	0	-	0	275.3
	$5 \cdot 10^{-3}$	0	-	0	104.4	0	-	0	237.7
	$1 \cdot 10^{-3}$	0	-	0	87.0	0	-	0	209.9
	$5 \cdot 10^{-4}$	0	-	0	79.0	0	-	0	196.6
	$1 \cdot 10^{-4}$	0	-	0	57.6	0	-	0	162.1
	$5 \cdot 10^{-5}$	0	-	0	34.3	0	-	0	123.0
	$1 \cdot 10^{-5}$	0	1.6e+02	1	-1.3	0	-	0	62.8
	$5 \cdot 10^{-6}$	0	1.1e+02	4	-4.7	0	-	0	57.2
	$1 \cdot 10^{-6}$	0	6.5e+02	6	-8.0	0	-	0	51.7
0.75	$1 \cdot 10^{-1}$	0	-	0	145.3	0	-	0	322.4
	$5 \cdot 10^{-2}$	0	-	0	145.3	0	-	0	322.4
	$1 \cdot 10^{-2}$	0	-	0	145.3	0	-	0	322.4
	$5 \cdot 10^{-3}$	0	-	0	135.5	0	-	0	305.0
	$1 \cdot 10^{-3}$	0	-	0	115.7	0	-	0	273.5
	$5 \cdot 10^{-4}$	0	-	0	65.2	0	-	0	169.1
	$1 \cdot 10^{-4}$	0	-	0	27.3	0	-	0	107.7
	$5 \cdot 10^{-5}$	0	-	0	18.7	0	-	0	94.9
	$1 \cdot 10^{-5}$	0	1.6e+02	1	-0.1	0	-	0	65.0
	$5 \cdot 10^{-6}$	0	1.1e+02	5	-3.4	0	-	0	58.6
	$1 \cdot 10^{-6}$	0	7.0e+02	7	-9.9	0	-	0	47.7

In tables 5 and 6 we assess the relative quality of k-shortest path solutions with respect to RPSP, in terms of both cost and probability of failure, when $k = 2$ or $k = 3$, that represent popular choices in path protection schemes. The tables are organized as in the previous experiments.

No CPU time is reported, as all min-cost flow computations took negligible time. Column 3-7 and 8-11 refer to the cases $k = 2$ and $k = 3$, respectively. We remark that in some instances (especially when graphs are sparse) it might be impossible to find k disjoint paths from s to t, and therefore k-shortest path protection schemes simply cannot be used (while RPSP could); for each technique (columns 3 and 8) we report the number of instances in each class where such a condition happened: these were excluded from the tests. In subsequent columns we report the difference between the actual failure probability of the k-shortest path optimal solution, when evaluated through BN computations, and $\mathcal{F}$ (expressed as relative value w.r.t. $\mathcal{F}$), the number of instances in which the failure probability target was not reached by the k-shortest path protection

Table 6: Comparison of path protection schemes on instances with 30 nodes.

γ	$\mathcal{F}$	2-shortest path				3-shortest path			
		# inf	$\mathcal{F}$ gap (%)	# target missed	RSPP opt. gap (%)	# inf	$\mathcal{F}$ gap (%)	# target missed	RSPP opt. gap (%)
0.25	$1 \cdot 10^{-1}$	1	-	0	169.0	3	-	0	364.4
	$5 \cdot 10^{-2}$	1	-	0	169.0	3	-	0	364.4
	$1 \cdot 10^{-2}$	1	-	0	168.1	3	-	0	362.6
	$5 \cdot 10^{-3}$	1	-	0	168.1	3	-	0	362.6
	$1 \cdot 10^{-3}$	1	-	0	141.5	3	-	0	328.2
	$5 \cdot 10^{-4}$	1	-	0	127.4	3	-	0	289.6
	$1 \cdot 10^{-4}$	1	-	0	78.4	3	-	0	198.5
	$5 \cdot 10^{-5}$	1	-	0	29.0	3	-	0	130.0
	$1 \cdot 10^{-5}$	1	-	0	8.3	3	-	0	87.8
	$5 \cdot 10^{-6}$	0	3.0e+00	1	8.5	2	-	0	88.2
	$1 \cdot 10^{-6}$	0	2.0e+02	4	4.0	2	-	0	78.8
0.50	$1 \cdot 10^{-1}$	0	-	0	125.4	0	-	0	266.5
	$5 \cdot 10^{-2}$	0	-	0	125.4	0	-	0	266.5
	$1 \cdot 10^{-2}$	0	-	0	125.4	0	-	0	266.5
	$5 \cdot 10^{-3}$	0	-	0	122.6	0	-	0	262.1
	$1 \cdot 10^{-3}$	0	-	0	104.0	0	-	0	229.1
	$5 \cdot 10^{-4}$	0	-	0	104.0	0	-	0	229.1
	$1 \cdot 10^{-4}$	0	-	0	93.2	0	-	0	212.1
	$5 \cdot 10^{-5}$	0	-	0	64.6	0	-	0	165.1
	$1 \cdot 10^{-5}$	0	2.6e+02	1	13.1	0	-	0	83.6
	$5 \cdot 10^{-6}$	0	3.1e+02	2	-4.6	0	-	0	52.9
	$1 \cdot 10^{-6}$	0	1.9e+03	2	-4.6	0	-	0	52.9
0.75	$1 \cdot 10^{-1}$	0	-	0	110.3	0	-	0	226.1
	$5 \cdot 10^{-2}$	0	-	0	110.3	0	-	0	226.1
	$1 \cdot 10^{-2}$	0	-	0	110.3	0	-	0	226.1
	$5 \cdot 10^{-3}$	0	-	0	110.3	0	-	0	226.1
	$1 \cdot 10^{-3}$	0	-	0	108.4	0	-	0	223.1
	$5 \cdot 10^{-4}$	0	-	0	100.1	0	-	0	210.1
	$1 \cdot 10^{-4}$	0	-	0	95.4	0	-	0	203.1
	$5 \cdot 10^{-5}$	0	-	0	74.8	0	-	0	170.8
	$1 \cdot 10^{-5}$	0	-	0	23.6	0	-	0	91.9
	$5 \cdot 10^{-6}$	0	1.2e+00	1	-1.4	0	-	0	52.7
	$1 \cdot 10^{-6}$	0	2.0e+02	2	-2.4	0	-	0	51.2

scheme, and the difference between the k-shortest path optimum and the best known RPSP solution value (expressed as relative value w.r.t. the best known RPSP solution value). That is, when this last value is positive, k-shortest path solutions are more expensive than RPSP ones (when they are negative the opposite holds). All values in these columns are averages only over those instances containing k disjoint paths.

Our results convey a clear message: k-shortest path models are unsuitable to meet failure probability targets with effective cost solutions. In all instances both path protection schemes produced solutions which were either more expensive than RPSP (when $\mathcal{F}$ is large), or infeasible w.r.t. failure probability (when $\mathcal{F}$ is small); sometimes, both conditions hold simultaneously. Indeed, a similar effect has been observed in previous attempts from the literature [17]. The aspect of costs appear particularly relevant: often the cost of RPSP solutions is a fraction of that of k-shortest path models, even when $\mathcal{F}$ is very high. Only 2-path protection seems competitive w.r.t. costs, when $\mathcal{F}$ takes values

Table 7: Comparison of RPSP heuristics on instances with 60 nodes.

γ	$\mathcal{F}$	2-shortest path				3-shortest path			
		# inf	$\mathcal{F}$ gap (%)	# target missed	RSPP opt. gap (%)	# inf	$\mathcal{F}$ gap (%)	# target missed	RSPP opt. gap (%)
0.25	$1 \cdot 10^{-1}$	1	-	0	165.3	1	-	0	374.1
	$5 \cdot 10^{-2}$	1	-	0	165.3	1	-	0	374.1
	$1 \cdot 10^{-2}$	1	-	0	146.4	1	-	0	342.5
	$5 \cdot 10^{-3}$	1	-	0	149.9	1	-	0	348.3
	$1 \cdot 10^{-3}$	1	-	0	72.4	1	-	0	200.8
	$5 \cdot 10^{-4}$	1	-	0	72.4	1	-	0	200.8
	$1 \cdot 10^{-4}$	1	5.9e+01	1	48.1	1	-	0	162.0
	$5 \cdot 10^{-5}$	1	2.2e+02	1	28.2	1	-	0	126.5
	$1 \cdot 10^{-5}$	1	6.5e+02	3	2.9	1	-	0	82.5
	$5 \cdot 10^{-6}$	1	8.4e+02	5	-3.2	1	-	0	72.5
	$1 \cdot 10^{-6}$	1	3.9e+03	6	-16.1	1	-	0	46.4
0.50	$1 \cdot 10^{-1}$	0	-	0	146.7	0	-	0	334.2
	$5 \cdot 10^{-2}$	0	-	0	146.7	0	-	0	334.2
	$1 \cdot 10^{-2}$	0	-	0	146.7	0	-	0	334.2
	$5 \cdot 10^{-3}$	0	-	0	84.9	0	-	0	230.8
	$1 \cdot 10^{-3}$	0	-	0	59.8	0	-	0	175.1
	$5 \cdot 10^{-4}$	0	-	0	67.3	0	-	0	189.1
	$1 \cdot 10^{-4}$	0	-	0	49.5	0	-	0	159.7
	$5 \cdot 10^{-5}$	0	-	0	24.1	0	-	0	115.0
	$1 \cdot 10^{-5}$	0	2.4e+00	1	2.1	0	-	0	77.2
	$5 \cdot 10^{-6}$	0	3.2e+01	4	-4.3	0	-	0	66.8
	$1 \cdot 10^{-6}$	0	4.5e+02	5	-8.8	0	-	0	58.6
0.75	$1 \cdot 10^{-1}$	0	-	0	137.4	0	-	0	298.1
	$5 \cdot 10^{-2}$	0	-	0	137.4	0	-	0	298.1
	$1 \cdot 10^{-2}$	0	-	0	137.4	0	-	0	298.1
	$5 \cdot 10^{-3}$	0	-	0	136.5	0	-	0	296.8
	$1 \cdot 10^{-3}$	0	-	0	91.0	0	-	0	213.1
	$5 \cdot 10^{-4}$	0	-	0	99.1	0	-	0	225.7
	$1 \cdot 10^{-4}$	0	-	0	86.8	0	-	0	207.1
	$5 \cdot 10^{-5}$	0	-	0	43.9	0	-	0	132.3
	$1 \cdot 10^{-5}$	0	1.8e+02	2	4.9	0	-	0	69.9
	$5 \cdot 10^{-6}$	0	3.0e+02	3	-7.1	0	-	0	50.5
	$1 \cdot 10^{-6}$	0	1.9e+03	3	-7.8	0	-	0	49.4

in the order of 10^{-6}, but in these cases the violation of the failure probability constraint is very large.

5.4. RPSP as heuristics

The experiments of the previous subsection did not take into account computing time scalability. Therefore, we analyzed the suitability of our RPSP algorithms in a different setting. First, we considered larger instances: we kept the same instance structure, but created graphs with 40 to 60 nodes. Second, we set a limit to RPSP computations, stopping them at the root not or in any case after 60 seconds of CPU time, and the RPSP incumbent was kept as heuristic solution. Our results on the graphs of 60 nodes are reported in Table 7; its structure is identical to that of tables 5 and 6. Other results are omitted, being pretty much in line with previous ones.

The outcome is similar to the previous experiment: only 2-shortest path solutions allow lower costs than heuristic RPSP solutions, when $\mathcal{F}$ is in the

order of 10^{-6}, but especially in these cases the relative violation of the failure probability constraint is large. 3-shortest path solutions are often impossible to find in sparse graphs ($\gamma = 0.25$); when they exist, the failure probability constraint is always met also in the 3-shortest path solutions, but their cost is much larger that the heuristic RPSP ones.

6. Conclusions

The first message we get from our theoretical analysis is the following: there is no free lunch for refining the handling of probabilistic components in network problems like the RPSP. In fact, as soon as constraints explicitly handling failure probabilities are included in optimization models, even feasibility checking problems become NP-Complete. We showed that if no assumption on the topology of the network is made, the RPSP can be solved in polynomial time only for cases where the target failure probability is either 0 or 1. Although these are very specific cases, their understanding helps in the design of RPSP algorithms.

Fortunately, mathematical programming offers tools to cope with such an additional complexity, at least from a computational point of view: we designed algorithms which are able to consistently provide proven optimal solutions on networks of realistic size (i.e. 89% of instances with graphs of 20 nodes, and 90% of those with 30 nodes). The expected computing effort seems to be more correlated to the failure probability target value than to network size. The key ingredients in our case are two: an effective way of checking feasibility (i.e. a careful use of Bayesian Networks), and an effective integer relaxation scheme. In fact, these ingredients nicely fit in a branch-and-price framework.

Finally, we report that indeed models like the RPSP require additional design and computational effort, but their use strongly pay off with respect to standard path protection mechanisms from the literature. Scalability remains a key factor in favor of standard path protection schemes: networks with thousands of nodes appear out of reach for our methods. However, our algorithms prove to behave well also when used as heuristics by strongly limiting the computing time. At the same time, in all our experiments the RPSP solutions were either much more cost effective than competitors, or even the only way of achieving failure probability targets.

Acknowledgments

Partially funded by Università degli Studi di Milano, "Piano Sostegno alla Ricerca 2016-2020" and Regione Lombardia, grant agreement n. E97F17000000009, Project AD-COM. The authors are grateful to three anonymous reviewers, whose insightful comments helped to improve the paper.

References

[1] M. Pioro, D. Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks, The Morgan Kaufmann Series in Networking, Elsevier Science, 2004.

[2] H. Kerivin, A. R. Mahjoub, Design of survivable networks: A survey, Networks 46 (1) (2005) 1–21.

[3] Z.-J. M. Shen, R. L. Zhan, J. Zhang, The reliable facility location problem: Formulations, heuristics, and approximation algorithms, INFORMS Journal on Computing 23 (3) (2011) 470–482.

[4] O. Berman, D. Krass, M. B. C. Menezes, Facility reliability issues in network p-median problems: Strategic centralization and co-location effects, Operations Research 55 (2) (2007) 332–350.

[5] J. Puerto, F. Ricca, A. Scozzari, Reliability problems in multiple path-shaped facility location on networks, Discrete Optimization 12 (2014) 61 – 72.

[6] L. V. Snyder, M. S. Daskin, Reliability models for facility location: The expected failure cost case, Transportation Science 39 (3) (2005) 400–416.

[7] J. Cheng, A. Lisser, Maximum probability shortest path problem, Discrete Applied Mathematics 192 (2015) 40 – 48, 11th Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2012).

[8] M. Redmond, A. Campbell, J. Ehmke, The most reliable flight itinerary problem, Networks 73 (3) (2019) 325–343, cited By 1.

[9] A. Andreas, J. Smith, S. Küçükyavuz, Branch-and-price-and-cut algorithms for solving the reliable h-paths problem, Journal of Global Optimization 42 (2008) 443–466.

[10] S. J. Koh, C. Y. Lee, A tabu search for the survivable fiber optic communication network design, Computers & Industrial Engineering 28 (4) (1995) 689 – 700.

[11] F. Robledo, P. Romero, M. Saravia, On the interplay between topological network design and diameter constrained reliability, in: 12th International Conference on the Design of Reliable Communication Networks (DRCN), 2016.

[12] N. Gonzalez-Montoro, R. Cherini, J. M. Finochietto, A multiple-link failures enumeration approach for availability analysis on partially disjoint paths, in: 13th International Conference on the Design of Reliable Communication Networks (DRCN), 2017.

[13] A. Konak, A. E. Smith, Network Reliability Optimization, Springer US, Boston, MA, 2006, pp. 735–760.

[14] Rong-Hong Jan, Fung-Jen Hwang, Sheng-Tzong Chen, Topological optimization of a communication network subject to a reliability constraint, IEEE Transactions on Reliability 42 (1) (1993) 63–70.

[15] J. Barrera, H. Cancela, E. Moreno, Topological optimization of reliable networks under dependent failures, Operations Research Letters 43 (2) (2015) 132 – 136.

[16] A. Preékopa, Stochastic Programming, Elsevier B.V., 2003, Ch. Probabilistic Programming, pp. 267–351.

[17] Y. Song, J. Luedtke, Branch-and-cut approaches for chance-constrained formulations of reliable network design problems, Mathematical Programming Computation 5 (2013) 397–432. doi:10.1007/s12532-013-0058-3.

[18] E. Canale, H. Cancela, F. Robledo, P. Romero, P. Sartor, Diameter constrained reliability: Complexity, distinguished topologies and asymptotic behavior, Networks 66 (4).

[19] P. Weber, L. Jouffe, Complex system reliability modelling with dynamic object oriented bayesian networks (doobn), Reliability Engineering & System Safety 91 (2) (2006) 149 – 162, selected Papers Presented at QUALITA 2003.

[20] C. Gonzales, L. Torti, P.-H. Wuillemin, aGrUM: a Graphical Universal Model framework, in: International Conference on Industrial Engineering, Other Applications of Applied Intelligent Systems, Proceedings of the 30th International Conference on Industrial Engineering, Other Applications of Applied Intelligent Systems, Arras, France, 2017.

[21] E. Boros, A. Scozzari, F. Tardella, P. Veneziani, Polynomially computable bounds for the probability of the union of events, Mathematics of Operations Research 39 (4) (2014) 1311–1329.

[22] T. Achterberg, Scip: solving constraint integer programs, Mathematical Programming Computation 1 (1) (2009) 1–41.

[23] CPLEX development team, IBM Ilog CPLEX Optimization Studio: CPLEX User's Manual - Version 12 Release 6, Tech. rep., IBM corp. (2011).

[24] M. Casazza, A. Ceselli, A. Taverna, Mathematical formulations for the optimal design of resilient shortest paths, in: New Trends in Emerging Complex Real Life Problems, Proceedings of AIRO 2018, 2018.

[25] A. Ceselli, G. Righini, E. Tresoldi, Combined location and routing problems for drug distribution, Discrete Applied Mathematics (2014) 130–145.

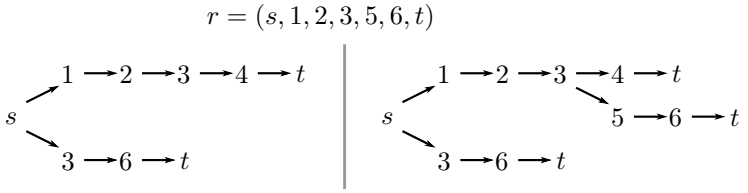

Figure 2: Example of path insertion into the prefix tree: a new path $r = (s, 1, 2, 3, 5, 6, t)$ is inserted; the longest subpath of r found in the tree is $(1, 2, 3)$, and the rest of the path is attached as a new branch.

Appendix A

In this section we detail the implementation of the prefix tree that we use to efficiently retrieve the values of M_ρ values during pricing.

A similar technique is used in [25]. The prefix tree is defined as $T^R = (N^R, A^R)$, where N^R is its set of nodes of the tree and A^R is its set of the arcs.

init The tree starts with $N^R = \{s\}$ and $A^R = \emptyset$.

insert When a column encoding a new path $r = (\sigma_1, \sigma_2, \ldots, \sigma_k)$ is generated, the tree is enlarged: first, we search for the longest branch in the tree T^R, which encodes a subpath $(\sigma_1, \ldots, \sigma_m)$ of r; such a branch may also be the single root node (i.e. $m = 1$). Then a sequence of nodes $(\sigma_{m+1}, \ldots, \sigma_k)$ is created and appended to the branch of the tree. Each insert requires linear time in k.

An example of how the insertion is performed is depicted in Figure 2. It follows that our tree has a root node that is the source node s, while each leaf is a copy of the destination node t.

update A path $r \in \bar{R}$ corresponds to each leaf. At each RMP optimization, such a leaf is labeled with value M_r. Each other node of the tree is labeled recursively with the minimum label among its direct children. This is done bottom up from leaves to the root. Each update requires linear time in N^R.

query Each M_ρ value can be computed during dominance checks by visiting T^R. Each query requires linear time in the length of ρ.

Appendix B

In this section we detail the proof of correctness for branching on arcs. Let us assume that and arc $\hat{a} = (\hat{i}, \hat{j})$ has a positive profit, we may have four distinct cases:

- both partial paths ρ and ρ' has visited $\hat{j}$: in this case both paths either collected or not the profit, but both paths cannot collect it in the future;
- l visited j but l' did not: l cannot collect the profit but l' might collect it if it has not visited i yet;
- both partial paths have not visited j yet: if both did not visit i, both can collect the profit. If both visited i, then they both can collect the profit only if i is the last node of the partial path. If only l visited i, then l cannot collect the profit.

We remark that any other combination where l' has visited a node that l does not it is excluded by our condition.