SAPIENZA
UNIVERSITÀ DI ROMA

DIPARTIMENTO DI SCIENZE STATISTICHE

DOTTORATO IN RICERCA OPERATIVA - XXVI CICLO

Robust Path Planning: Models and Algorithms

Author:
Marco SENATORE

Supervisor:
Prof. Gianpaolo ORIOLO

Contents

Introduction **1**

1 Related work **5**

1.1 The Replacement Path Problem and the Most Vital Arc Problem . . 6

1.2 The Canadian Traveller Problem 7

1.3 The Stochastic On-time Arrival Problem 10

2 The Online Replacement Path Problem **11**

2.1 The On-line Replacement Path Problem 12

2.1.1 Notations, definitions and some properties 12

2.1.2 Feasible potential and optimality conditions 17

2.1.3 A label setting algorithm for ORP 22

2.1.4 The undirected case 25

2.2 ORP with arbitrary costs 27

2.2.1 ORP with arbitrary costs in absence of negative cycles 27

2.2.2 ORP with arbitrary costs in presence of negative cycles . . . 28

2.3 Bi-objective ORP 32

2.4 k-ORP 37

2.4.1 The Shortest Path Heuristics 39

3 ORP Generalizations **42**

3.1 k-Hop ORP 43

3.1.1 Radius ORP 50
3.2 Strong k-Hop ORP 51

4 ORP Game **54**
4.1 ORP Game in mixed strategy: Stochastic ORP 57

Conclusions and future work **62**

Bibliography **66**

Introduction

Modeling the effects of limited reliability of networks in modern routing schemes is important in many applications. It is often unrealistic to assume that the nominal network known at the stage of decision making will be available in its entirety at the stage of solution implementation. Several research directions have emerged as a result. The main paradigm in most works is to obtain a certain 'fault-tolerant' or 'redundant' solution, which takes into account a certain set of likely network realizations at the implementation phase.

Shortest paths are often used in order to minimize routing time. In faulty network, however, simply taking the shortest path might lead to very large delays due to link failures. Two related problems that were extensively studied in the literature are the *Most Vital Arc problem (MVA)* and the *Replacement Path problem (RP)*. MVA asks given a graph $G = (V, E)$ and two nodes $s, t \in V$ to find the edge $e \in E$ whose removal results in the maximal increase in the s-t distance in G. The input to RP additionally includes a shortest path P, and the goal is to find for every $e \in P$ a shortest s-t path P_e avoiding e.

In the context of robust network design both MVA and RP should be interpreted as problems in which the Routing Mechanism (RM) is informed about the failed edge *in advance*, namely when standing at s. This assumption is unrealistic in many situations, in which failures occur *online*, and in particular, after the routing has started. Examples of such situations range from accidents and traffic jams in road network to truly adversarial setups, in which the adversary is motivated to conceal the failure for as long as possible.

In this thesis, in order to propose a more realistic model, we assume that RM discovers the identity of the failed edge, if any, while traversing the network. Depending on the way RM acquires knowledge about the network itself, we define different problems.

In the *Online Replacement Path problem (ORP)* [2] we assume that the materialized scenario is revealed to the RM 'in the last minute', namely only when the package reaches one of the endpoints of the failed edge and attempts to cross it. From this point on the package is routed through a *detour*, namely a path from the current node to the destination that avoids the failed edge. The *robust length* of a path is the maximum total travel time over all possible failure scenarios, and the goal is to provide the RM with a path with minimum robust length.

ORP models online failure scenarios that occur in many situations, some of which we described before. In other applications it is only necessary to route a certain object within a certain time, called a *deadline*. As long as the object reaches its destination before the deadline, no penalty is incurred. On the other hand, if the deadline is not met, a large penalty is due. An example of such an application is organ transportation for transplants (see e.g. Moreno, Valls and Ribes [18]), in which it is critical to deliver a certain organ before the scheduled time for the surgery. In this application it does not matter how early the organ arrives at the destination, as long as it arrives in time. In such applications it is often too risky to take an unreliable shortest path, which admits only long detours in some scenarios, whereas a slightly longer path with reasonably short detours meets the deadline in *every* scenario. Thus, this domain of applications can also benefit from ORP.

Our first result is a polynomial algorithm for ORP. Concretely, we show that optimal robust u-t path can be found in time $O(m + n \log n)$ in undirected graphs and $O(n\mathrm{SP}(n, m))$ in directed graphs for all sources $u \in V$ and a single destination t, where $\mathrm{SP}(n, m)$ is the complexity of a single shortest path computation on a graph with n nodes, m edges and nonnegative costs. We prove various properties of ORP, in particular, we show the existence of a tree of optimal paths, and that

the robust length is monotonic with respect to taking subpaths of optimal paths. These properties lead to a natural label-setting algorithm. We also develop a poly-time algorithm for the generalization of ORP when some fixed number of edges can fail; even if more involved, the algorithm goes along the same lines of that for the single-failure case. For the formal definition of ORP and a detailed description of the related results we refer to Chapter 2.

In those applications in which failures are not extremely likely, the ORP solutions might be too conservative. For this reason we study *Bi-objective ORP*, the optimization problem of finding a shortest path in the graph with robust length at most a given bound. In Chapter 2 we show that this problem admits an algorithm with running time $O(m + n \log n)$ in undirected graphs (and $O(mn + n^2 \log n)$ in directed graphs). We also show that the Pareto front of the latter bi-objective problem has linear size in the size of the graph for both directed and undirected graphs. This could be useful in practical applications, as the decision maker can efficiently plot the trade-off between the *nominal* and the *robust* length of Pareto-efficient solutions.

We study various extensions of ORP; in particular we introduce two models that provide a middle ground between MVA and ORP.

In Chapter 3 we introduce the *k-Hop ORP* problem, in which the RM is informed about the failed edge e as soon as it reaches a node that is k hops away from e on the nominal path. While 0-Hop ORP is simply ORP, one easily sees that $(n-1)$-Hop ORP is equivalent to MVA. For $k \in \{1, \cdots, n-2\}$ we obtain an interesting continuum of problems between ORP and MVA. We show that some of the nice properties that hold for ORP no longer hold for k-Hop ORP. In particular, while a tree of optimal paths always exists, the robust length of a subpath in this tree can be larger than the robust length of the original path. Nevertheless, we obtain a label-setting algorithm for this problem, whose running time is polynomial for both the directed and the undirected case and is independent from k. Let us observe that our results regard only the case where k is bounded. We briefly describe also another variant called *Radius ORP*, in which the RM is informed about the failed edge e as

soon as it reaches a node that is at most at distance R from e on the nominal path, where R is a given radius indicating how far the RM can see along the nominal path. It turns out that Radius ORP can be solved like k-Hop ORP, since the key properties of the latter problem still hold for the former one. Going back to the case in which we measure the distance from the failed edge in terms of hops, we cannot find any longer a poly time algorithm to solve the variant where the RM is informed about the failed edge e as soon as it reaches a node that is k hops away from e in *the graph*, and not just on *the nominal path*. While this variant is equivalent to ORP for $k = 0$, we show that, already with $k = 1$, it is NP-hard to approximate within a factor of $3 - \epsilon$ for undirected graphs (and provide a simple algorithm meeting this factor), and it is strongly NP-hard to decide if there exists a nominal path with finite robust length for directed graphs.

In Chapter 4 we study the *ORP Game*, a two players' game related to MVA and ORP. In this game a first player, the *path builder*, is interested in arriving from s to t as quickly as possible. The second player, the *interdictor*, tries to make the latter distance as long as possible by removing a single edge from the graph. The *strategies* for the two players are the s-t paths, and the edges $e \in E$, respectively. One can see ORP and MVA as variants of the ORP Game, in which strategies are not communicated simultaneously. We show that the instances of the game which admit a pure Nash Equilibrium (NE) are exactly those where the values of the optimal solutions to ORP and MVA are equal, and build upon this fact to give an $O(m + n \log n)$-time algorithm that finds it in undirected graphs (and in time $O(n\mathrm{SP}(n, m))$-time in directed graphs), or reports that no pure NE exists. Furthermore, we address the problem of finding a NE of the mixed strategy extension of ORP Game.

Finally, we conclude this thesis summarizing some open points and addressing some future research directions.

Chapter 1

Related work

In this chapter, we review a few problems that are related to ORP.

In the Introduction we already discussed the connection between ORP and the Replacement Path problem (RP) and the Most Vital Arc problem (MVA), underlining how, in the context of robust network design, both MVA and RP should be interpreted as problems in which the RM is informed about failures in the network in advance. In Section 1.1 we give a detailed description of the state of the art of these two well-known problems.

Another problem which bears resemblance to ORP is the *Canadian Traveller problem* (CTP), whose properties and different variants are extensively described in Section 1.2, while in Section 1.3 we present some stochastic models that can be seen as the stochastic analogue of ORP.

We close this section with a quick review of some related work on robust counterparts of the shortest path problem. The shortest path problem with cost uncertainty was studied by Yu and Yang [34], who consider several models for the scenario set. These results were later extended by Aissi, Bazgan and Vanderpooten [3]. These works also considered a two-stage min-max regret criterion. Dhamdhere, Goyal, Ravi and Singh [10] developed the demand-robust model and gave an approximation algorithm for the shortest path problem. A two-stage feasibility counterpart of the shortest path problem was addressed by Adjiashvili and Zenklusen [1].

1.1 The Replacement Path Problem and the Most Vital Arc Problem

Let us start by formally define the Replacement Path problem and the Most Vital Arc problem. In the former, we are given a graph (directed or undirected) $G = (V, E)$, two nodes $s, t \in V$ and a shortest s-t path P and we are asked to find for every edge $e \in P$ a shortest s-t path P_e in $G - e$. In the latter, we are given a graph (directed or undirected) $G = (V, E)$, two nodes $s, t \in V$ and we are asked to find the edge $e \in E$ whose removal results in the maximal increase in the s-t distance in G. It is clear, then, that the two problems are very close to each other.

The Replacement Path problem was first introduced by Nisan and Ronen [25]. The motivation for their definition stemmed from the following question in auction theory: what is the true price of a link in a network, when we try to connect two distinct vertices s and t, and every edge is owned by a self-interested agent. A key concept in in mechanism design is the Vickrey-Clarke-Groves pricing mechanism. In a pricing mechanism, each agent declares his preference, and the system computes a payment function for each agent. The importance of VCG stems from the fact that, under this protocol, each agent optimizes his utility by disclosing his true preference. Nisan and Ronen [25] show that if the payoff for each agent is given by formula (1.1), then the pricing mechanism is indeed a Vickrey-Clarke-Groves pricing mechanism.

$$c(e) := \begin{cases} d(s,t;G-e) - d(s,t;G|_{e=0}), & \text{if } e \text{ belongs to the } s\text{-}t \text{ shortest path} \\ 0, & \text{otherwise} \end{cases} \tag{1.1}$$

That is, if the edge e does not belong to the shortest s-t path in G, then its agent receives zero payment. Otherwise, the payment to e is the difference between the cost of the shortest s-t path not using e, and the cost of the shortest s-t path assuming e free.

The complexity of the RP problem for undirected graphs is well understood. The first paper to study this problem is due to Malik, Mittal and Gupta [17], who give a simple $O(m + n \log n)$ algorithm. A mistake in this paper was later corrected

by Bar-Noy, Khuller and Schieber [4]. As a bi-product, the latter result implies an $O(m + n \log n)$-time algorithm for the Most Vital Arc (MVA) problem. This running time is asymptotically the same as a single source shortest path computation. Nardelli, Proietti and Widmayer [22] later extended the result to account for node failures. In [19] the same authors gave an algorithm that finds a detour-critical edge on a shortest path. The complexity for MVA was later improved by Nardelli, Proietti and Widmayer [21] to $O(m\alpha(m, n))$, where $\alpha(\cdot, \cdot)$ is the Inverse Ackermann function.

In directed graphs the situation is significantly different. A trivial upper bound for RP corresponds to $O(n)$ single shortest path computations. This gives $O(n(m + n \log n))$ for general directed graphs with nonnegative weights. This was slightly improved to $O(mn + n^2 \log \log n)$ by Gotthilf and Lewenstein [14]. The challenge of improving the $O(mn)$ bound for RP on directed graphs was mainly tackled by restricting the class of graphs or by allowing approximate solutions. Along the lines of the former approach, algorithms were developed for unweighted graphs (Roditty and Zwick [28]) and planar graphs (Emek, Peleg and Roditty [11], Klein, Mozes, and Weimann [16] and Wulff-Nilsen [32]). The latter approach was successfully applied to obtain $\frac{3}{2}$-approximate solutions by Roditty [27] and $(1+\epsilon)$-approximate solutions by Bernstein [6]. Weimann and Yuster [31] applied fast matrix multiplication techniques to obtain a randomized algorithm with sub-cubic running time for certain ranges of the edge weights.

1.2 The Canadian Traveller Problem

The Canadian Traveller Problem (CTP) was first introduced in 1991 by Papadimitriou and Yannakakis; in [26] they define several versions of the shortest path problem when the graph is not known in advance, but is revealed dynamically, while it is being traversed. For a given instance, there are a number of possibilities, or realizations, of how the hidden graph may look. Given an instance, a description of how to follow the instance in the best way is called a *routing policy* or *strategy*. The CTP task is to compute the expected cost of the *optimal policies*.

The authors took inspiration from a situation commonly encountered by travellers in Canada: once a driver reaches an intersection, he sees whether the incident roads are snowed out or not and consequently decides which road to take.

Formally speaking, in the CTP we are given a directed graph $G = (V, E)$, a non-negative cost function on the edges, a start node $s \in V$, a target node $t \in V$ and a ratio r. Each edge $(u, v) \in E$ can fail, but the searcher finds out whether the edge can be traversed or not only when u is visited. In the original version of CTP, the goal is to find a strategy for traversing the graph, starting from s, ending in t, such that the total distance traversed is no more than r times the shortest path from s to t in G in the realized scenario.
The CTP can be seen as the specification of a *two-person game*, between a searcher and a malicious adversary , who sets the realizations of the edges so as to maximize the ratio. This problem turns out to be PSPACE-*complete* [26].

In the same work, Papadimitriou and Yannakakis define also a couple of stochastic variants of CTP in which the lengths of the edges are random variables ruled by independent probability distributions and, again, when we arrive at a node we discover the actual length of its incident edges. In this case the goal could be either devising a strategy that minimizes the *expected* ratio to the optimal path, or devising a strategy that minimizes the expected distance traversed from the start to the goal node. These stochastic optimization problems are known to be #P-*hard*.

In [23] Nikolova and Karger focus on the stochastic CTP version in which the goal is to find an optimal policy for reaching , from a source s a destination t that minimizes the expected cost. The authors choose to seek out special cases for which exact solutions exist and, using techniques from the theory of Markov Decision Processes, they give an exact algorithm for graphs of parallel undirected paths with random two-valued edge costs. They also offer a partial generalization to traversing perfect binary trees.

In [5] the authors consider several variants of the CTP:

1. *The Recoverable Canadian Traveller Problem*

In this variant each node x of the graph is associated with a non-negative real time $l(x, x)$ interpreted as the recovery time of edges adjacent to x. Whenever the traveller, at a node x, finds a blocked edge e, he/she can either use another edge or wait the related recovery time and then use the edge e. Note that the number of edges that can fail is bounded by an integer k. The authors give an $O(k^2m + knlogn)$ algorithm for designing a travel strategy from all site to a fixed destination that guarantees the shortest worst-case travel time, under the assumption that $l(x, x) \leq l(e)$ (the travel time of e), for any e adjacent to x.

2. *The Stochastic Recoverable Canadian Traveller Problem*

   In this stochastic variant of the problem described above, each edge e is associated with a probability $p(e)$, being the probability for an edge to be failed. There is no bound on the number of failures. The authors provide an $O(mlogn)$ algorithm for designing a travel strategy from all sites to a fixed destination that guarantees the shortest expected travel time, under the assumption that $l(x, x) \leq l(e)$ (the travel time of e), for any e adjacent to x.

3. *The k-Canadian Traveller Problem*

   Let k be the bound on the number of blockages that may occur during a traveller. The authors prove that for arbitrary k the problem of designing a travel strategy that guarantees the shortest worst-case travel time is PSPACE-*complete*. For $k = 1$, they give an $O(m + nlogn)$ time algorithm for designing the strategy, claiming that it can be extended to give a polynomial-time travel strategy for any constant k.

This last problem can be seen as a policy-based variant of the problem we study in Chapter 2. The authors first claim, without proof, that the problem of finding an optimal routing policy reduces to the problem of finding an optimal path. Then they claim a result that is close to the one we present in Section 2.1. However, as we discuss later, we believe that these results are not adequately supported in [5] by

rigorous arguments.

1.3 The Stochastic On-time Arrival Problem

As we already mentioned in the Introduction, ORP models can have a strong impact to those applications where it is only necessary to route a certain object within a certain time, called deadline, dealing, at the same time, with the unreliability of the network due to links failures. In other cases such unreliability can be due to uncertainty about the links travel times, but still the goal is to meet a certain deadline with *high probability*. These applications have been modeled with the Stochastic On-Time Arrival problem (SOTA), that was first introduced in [33] by Fan et al.

In SOTA we are given a directed network $G = (V, E)$, a destination $d \in V$, and the weight of each edge $(i, j) \in E$ is a random variable with probability density function $p_{ij}(t)$ representing the probability that the travel time of (i, j) is equal to t. Such probability distribution functions are supposed to be independent. Given a budget time T, an optimal routing strategy is defined to be a policy that maximizes the probability of arriving, from any possible source $s \in V$, at the destination node d within time T. The solution is provided as a set of adaptive decision rules encoded in functions $u_i(t)$ that, for any node $i \in V$ and time budget t, denote the probability of reaching d from i in less than time t following the optimal policy.

In [33] Fan et al. formulate SOTA as a stochastic dynamic programming problem, solved using a standard successive approximation algorithm that in acyclic network converges in an unbounded number of steps. In [29], Samaranayake et al. show how the SOTA problem can be solved exactly in a finite number of steps, even in cyclic networks, thanks to a label setting algorithm based on the existence of a minimal link travel-time on each link; the authors also provide several speedup techniques and experimental results for the San Francisco road network.

In [24] Nikolova tackles the SOTA problem restricting the optimal policy to be a simple path and gives an exact $n^{\Theta(logn)}$ algorithm for the case of independent normally distributed edge lengths, which is based on quasi-convex maximization.

Chapter 2

The Online Replacement Path Problem

In the Introduction, we already gave an informal definition of the main topic of our work, that is the *Online Replacement Path problem (ORP)*. Let us briefly remind it: we assume that a Routing Mechanism (RM) wants to route a package as fast as possible between two nodes of a faulty network, meaning that one of its edges can fail. Furthermore we assume that the materialized failure scenario is revealed to the RM 'in the last minute', namely only when the package reaches one of the endpoints of the failed edge and attempts to cross it. From this point on the package is routed through a *detour*, namely a path from the current node to the destination that avoids the failed edge. The *robust length* of a path is the maximum total travel time over all possible failure scenarios, and the goal is to provide the RM with a path with minimum robust length. In this Chapter we formally define the ORP problem, providing a polynomial algorithm to solve it.

First of all, let us observe that in [12], Xiao et al. define the Anti-Risk Path problem (ARP) that is actually equivalent to ORP: they solve ARP only for undirected graphs developing an $O(mn + n^2 logn)$ algorithm and, in the conclusions of their work, they ask the question whether it is possible to find a more efficient approach to solve the problem for undirected graphs; the result we state in subsection 2.1.4

positively answers to such question, providing an improvement factor of $O(n)$. We also stress that the authors raise and leave open further questions about the complexity of the problem for directed graphs and for graphs with arbitrary costs: the first problem is solved in subsection 2.1.3, while the second one is tackled in section 2.2.

We will show a solving approach that recalls in some aspects the techniques used for the Shortest Path problem (for more details we refer the interested reader to [9]), underlining, on the other hand, all the differences between the two problems.

In section 2.3 we discuss a problem linking ORP and the Shortest Path problem; we conclude this Chapter by addressing in section 2.4 the case where a bounded number of edges of the network can fail, showing that it is possible to easily adapt the algorithm proposed for the single failure case.

2.1 The On-line Replacement Path Problem

2.1.1 Notations, definitions and some properties

Let us establish some notation first. We are given an edge-weighted directed graph $G = (V, E, \ell)$, where $\ell : E \to \mathbb{R}^+$ (we assume that the edge weights ℓ are nonnegative), and a destination $t \in V$. Let n and m be the number of nodes and edges of the input graph, respectively. A *path* P is a sequence $v_0, e_1, v_1, \ldots, e_k, v_k$, where $v_i \in V$ for each $i = 0, \ldots, k$ and $e_i = (v_{i-1}, v_i) \in E$ for each $i = 1, \ldots, k$. We indicate with $V(P) = \{v_0, \ldots, v_k\}$ the set of nodes of P and with $E(P) = \{e_1, \ldots, e_k\}$ the set of edges of P. We say that P is *from* v_0 *to* v_k, or that is a v_0-v_k *path*. It is *closed* if $v_0 = v_k$; it is *edge-simple* if $e_1, \ldots, e_k$ are distinct; it is *simple* if $v_0, \ldots, v_k$ are distinct; it is a *cycle* if it is closed, $v_0, \ldots, v_{k-1}$ are distinct and $k \geq 1$. Let $N^+(u)$ be the set of outgoing neighbors and $N^-(u)$ the set of ingoing neighbors of u in G. For a set of edges $A \subset E$ let $\ell(A) = \sum_{e \in A} \ell(e)$. For an edge $e \in E$ and a set of edges $F \subseteq E$, let $G - e$ and $G - F$ be the graph obtained by removing the edge e and the edges in F, respectively. For two paths Q_1, Q_2 with the property that last node

of Q_1 is the first node of Q_2 we let $Q_1 \oplus Q_2$ be their concatenation. Let us stress that in this subsection and in the following ones we will talk about simple paths. For two nodes $u, v \in V$ let $\mathcal{P}_{u,v}$ be the set of simple u-v paths in G. For a simple path P containing nodes u and v let $P[u, v]$ be the subpath of P from u to v. For an edge $e \in E$ and $u \in V$ let Q_u^{-e} be some fixed shortest u-t path in $G - e$ and let $\pi_u^{-e} = \ell(Q_u^{-e})$. We use the convention that $Q_u^{-e} = \emptyset$ and $\pi_u^{-e} = \infty$ if u and t are in different connected components in $G - e$.

It is convenient to define the *detour* of a path:

Definition 2.1.1. *The detour P^{-e} of a (simple) path $P \in \mathcal{P}_{v,t}$ with respect to an edge $e = (u, u') \in E$ is:*

- *the walk $P[v, u] \oplus Q_u^{-(u,u')}$, if $(u, u') \in E(P)$ (where u is the node closer to v on P);*
- *P, otherwise.*

The length of a detour is given by:

- $\ell(P^{-e}) = \ell(P[v, u]) + \pi_u^{-(u,u')}$, *if* $(u, u') \in E(P)$;
- $\ell(P^{-e}) = \ell(P)$, *otherwise.*

In the ORP context, the cost of connecting a node v to the destination t via a v-t path is defined as follows:

Definition 2.1.2. *Given a node $v \in V$, the robust length of the (simple) v-t path P is*

$$\mathrm{Val}(P) = \max_{e \in E} \ell(P^{-e}).$$

The robust length of a v-t path P is simply the maximal possible cost incurred by following P until a certain node, and then taking the best possible detour from that node to t which avoids the next edge on the path. To avoid confusion, we stress that in ORP we assume the existence of at most one failed edge in the graph. Consider

next a scenario in which an edge $(u, u') \in E(P)$ fails and let $u \in V$ be the node which is closer to v; clearly, the best detour is a shortest u-t path in the graph $G - (u, u')$.

We can now formally define ORP:

Definition 2.1.3 (The Online Replacement Path problem)**.** ***Given:*** *an edge-weighted directed graph $G = (V, E, \ell)$, where $\ell : E \to \mathbb{R}^+$ and a destination $t \in V$.*
Find: *for every $v \in V$ an optimal v-t path, namely a path P minimizing* $\mathrm{Val}(P)$ *over all paths $P \in \mathcal{P}_{v,t}$.*

Remark 2.1.4. *We observe that in definition 2.1.3 we are claiming that in order to find a v-t path with minimum robust length for each $v \in V$, we can restrict without loss of generality our focus to simple v-t paths. It is easy to check that this is directly implied by nonnegativity of ℓ.*

In the following, we are going to indicate with $\mathit{bottleneck}(P)$ the edge which is responsible for the worst case scenario for a given path P. We have $\mathit{bottleneck}(P) = \emptyset$ iff $\mathrm{Val}(P) = \ell(P)$, while $\mathit{bottleneck}(P) = (u, u')$ iff $\mathrm{Val}(P) = \ell(P[v, u]) + \pi_u^{-(u,u')}$. Furthermore, we will often call *nominal* the path P and the value $\ell(P)$ it is what we call *nominal length of P*.

We now provide a simple graph-theoretical characterization of paths with finite robust length. Let $U^2 \subset V$ be the set of nodes in G that are 2-edge connected to t, i.e., all nodes u such that there are two edge-disjoint u-t paths in G. Note that, following Theorem 2.1.5, if two nodes s and t are in different components of $G[U^2]$, there will be no s-t path with finite robust length.

Theorem 2.1.5. *A path $P \in \mathcal{P}_{s,t}$ has finite robust value if and only if $V(P) \subset U^2$.*

Proof. Assume first that $V(P) \not\subset U^2$. Let $u \in V(P)$ such that there are no two edge-disjoint u-t paths in G. In this case there exists an edge e such that $G - e$ contains no u-t paths. In fact $e \in P$ and u is reachable on P in $G - e$, since P contains a u-t path. It follows that $\mathrm{Val}(P) = \ell(P^{-e}) = \infty$, as u is reached in the scenario that e fails.

Assume next $V(P) \subset U^2$. We evaluate $\ell(P^{-e})$ for some failure scenario $e \in P$. Let $u \in V(P)$ be the node incident to P at which the failure is discovered. Since $u \in U^2$, there exists at least one u-t path Q in $G - e$. Thus, $\ell(P^{-e}) \leq \ell(P[s,u]) + \ell(Q) < \infty$. We conclude that $\mathrm{Val}(P) < \infty$. □

In the rest of this subsection we are going to show some important properties of the robust length. First of all, we can prove the following useful lemma:

Lemma 2.1.6. *Let $P_u \in \mathcal{P}_{u,t}$ and let $v \in N^-(u)$ be a node, not incident to P_u. Then* $\mathrm{Val}((v,u) \oplus P_u) = \max\{\ell(v,u) + \mathrm{Val}(P_u), \pi_v^{-(v,u)}\}$.

Proof. Let $P_v = (v,u) \oplus P_u$. Applying the definition we compute

$$\begin{aligned}
\ell(v,u) + \mathrm{Val}(P_u) &= \ell(v,u) + \max\{\ell(P_u), \max_{(z,w)\in E(P_u)}\{\ell(P_u[u,z]) + \pi_z^{-(z,w)}\}\} \\
&= \max\{\ell(v,u) + \ell(P_u), \max_{(z,w)\in E(P_u)}\{\ell(v,u) + \ell(P_u[u,z]) + \pi_z^{-(z,w)}\}\} \\
&= \max\{\ell(P_v), \max_{(z,w)\in E(P_v)\setminus\{(v,u)\}}\{\ell(P_v[v,z]) + \pi_z^{-(z,w)}\}\}.
\end{aligned}$$

Substituting in the desired expression we obtain

$$\begin{aligned}
&\max\{\ell(v,u) + \mathrm{Val}(P_u), \pi_v^{-(v,u)}\} = \\
&\max\{\ell(P_v), \max_{(z,w)\in E(P_v)\setminus\{(v,u)\}}\{\ell(P_v[u,z]) + \pi_z^{-(z,w)}\}, \pi_v^{-(v,u)}\} = \mathrm{Val}(P_v),
\end{aligned}$$

which proves the lemma. □

Furthermore, nonnegativity of ℓ implies $\mathrm{Val}(P) \geq \mathrm{Val}(P[v,t])$, whenever $v \in V(P)$, meaning that the robust length of a path has a sort of monotonicity property with respect to proper subpaths.

In order to introduce the next property, we observe that in ORP we do not have a suboptimality principle as in the Shortest Path problem; indeed, given an optimal v-t path P, it might be false that any subpath of P from a node $u \in V(P)$ to t is also optimal for u. In figure 2.1 we note, for example, that the path $P = \{v,u,t\}$ is optimal for v and it has robust length 1000, while the subpath $P[u,t]$ (that, in this case, is simply the edge (u,t)), whose robust length is 900, is not optimal for u.

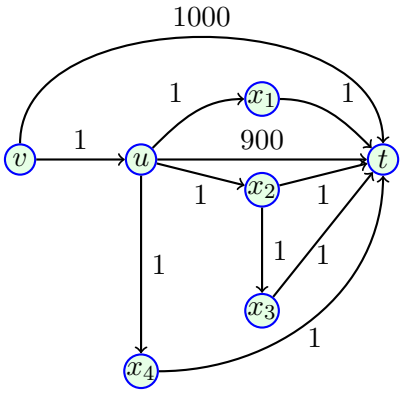

Figure 2.1: An example showing that subpaths of optimal path could be non-optimal.

Indeed, the optimal path for u is $P' = \{u, x_2, t\}$ with robust length 3.

On the other hand, also the path $P[v,u] \oplus P'$ is optimal for v, having robust length 1000. This fact is well explained by the following lemma, which states a sort of weak optimality principle for the ORP Problem.

Lemma 2.1.7 (Weak optimality principle)**.** *Let $P_v \in \mathcal{P}_{v,t}$ be an optimal path from v, $u \in V(P_v)$ and $P_u \in \mathcal{P}_{u,t}$ be an optimal path from u. Then the path $P'_v = P_v[v,u] \oplus P_u$ satisfies $\mathrm{Val}(P'_v) = \mathrm{Val}(P_v)$, namely it is also optimal from v.*

Proof. Since P_v is optimal for v, it is sufficient to prove that

$$\mathrm{Val}(P'_v) \leq \mathrm{Val}(P_v). \tag{2.1}$$

We define $P'_u = P_v[u,t]$; since P_u is optimal for u, we have

$$\mathrm{Val}(P_u) \leq \mathrm{Val}(P'_u). \tag{2.2}$$

By definition of robust length and by recursively applying lemma 2.1.6, we have $\mathrm{Val}(P_v) = \max\{A, \alpha + \mathrm{Val}(P'_u)\}$ and $\mathrm{Val}(P'_v) = \max\{A, \alpha + \mathrm{Val}(P_u)\}$, where $A = \max_{(z,w)\in E(P_v[v,u])}\{\ell(P_v[v,z]) + \pi_z^{-(z,w)}\}$ and $\alpha = \ell(P_v[v,u])$.

On the one hand, if $\mathrm{Val}(P'_v) = A$, then 2.1 trivially holds. On the other hand, if $\mathrm{Val}(P'_v) = \alpha + \mathrm{Val}(P_u)$, because of 2.2 we have:

$$\mathrm{Val}(P'_v) = \alpha + \mathrm{Val}(P_u) \leq \alpha + \mathrm{Val}(P'_u) = \mathrm{Val}(P_v),$$

and 2.1 holds. □

The properties of the robust length listed so far give us an idea about the reason why the solution to ORP can be provided as tree of optimal paths rooted in t.

2.1.2 Feasible potential and optimality conditions

As mentioned at the end of the previous subsection, a solution to the Online Replacement Path problem, defined in 2.1.3, is a tree of optimal paths rooted in t. In particular, in this subsection we provide a necessary and sufficient condition that such a tree has to satisfy in order to be optimal.

First of all, let us introduce the following definition:

Definition 2.1.8. *We call* potential *a vector* $y :\to \mathbb{R}^{|V|}$ *such that* $y(t) = 0$. *We call a potential* feasible *if and only if* $\forall (u, v) \in E$ *it holds:*

$$y(u) \leq \max\left\{\ell(u, v) + y(v), \pi_u^{-(u,v)})\right\} \tag{2.3}$$

The next lemma shows that a feasible potential is a vector such that, for any $v \in V$, $y(v)$ is a lower bound of the robust length of any simple v-t path.

Lemma 2.1.9. *Let* y *be a feasible potential and let* P *be a* v-t *path. Then* $y(v) \leq \mathrm{Val}(P)$.

Proof. Let $P = \{v \equiv v_1, v_2, \ldots, v_k \equiv t\}$ be a v-t path. Since y is feasible, we have:

$$y(v_i) \leq \max\left\{\ell(v_i, v_{i+1}) + y(v_{i+1}), \pi_{v_i}^{-(v_i,v_{i+1})})\right\} \quad \forall i = 1, \ldots, k-1 \tag{2.4}$$

We delve into two cases.

First case: $\mathit{bottleneck}(P) = \emptyset$**.** In this case, the worst scenario for P is the nominal one. Note that, in this case, the following holds:

$$\ell(P[v_i, v_k]) \geq \pi_{v_i}^{-(v_i,v_{i+1})} \quad \forall i = 1, \ldots, k-1 \tag{2.5}$$

Claim 1. *$y(v_i) \leq \ell(P[v_i, v_k])$, for each $i = 1, \ldots, k-1$.*

Proof We go by induction on i. If $i = k-1$, from (2.5) and (2.4), we have:

$$\ell(v_{k-1}, v_k) \geq \pi_{v_{k-1}}^{-(v_{k-1}, v_k)}$$

$$y(v_{k-1}) \leq \max\left\{\ell(v_{k-1}, v_k) + y(v_k), \pi_{v_{k-1}}^{-(v_{k-1}, v_k)}\right\}$$

As $y(v_k) = 0$, we conclude that $y(v_{k-1}) \leq \ell(v_{k-1}, v_k) = P[v_{k-1}, v_k]$.

By induction assume now that $y(v_{j+1}) \leq \ell(P[v_{j+1}, v_k])$, for some j between 1 and $k-2$. The following holds:

$$y(v_j) \leq \max\left\{\ell(v_j, v_{j+1}) + y(v_{j+1}), \pi_{v_j}^{-(v_j, v_{j+1})})\right\} \leq$$

$$\leq \max\left\{\ell(v_j, v_{j+1}) + \ell(P[v_{j+1}, v_k]), \pi_{v_j}^{-(v_j, v_{j+1})})\right\} = \ell(P[v_j, v_k]),$$

where the first inequality holds for the feasibility of y, the second inequality holds by induction, and the last follows from (2.5) with $i = j$. *(End of the proof)*

When $i = 1$, it follows from Claim 1 that $y(v_1) \leq \ell(P[v_1, v_k])$, i.e. $y(v) \leq \text{Val}(P)$.

Second case: *bottleneck*$(P) = (v_j, v_{j+1})$.

In this case, we assume that the worst scenario for P is when the edge (v_j, v_{j+1}) fails, for some j between 1 and $k-1$. We have:

$$\text{Val}(P) = \ell(P[v_1, v_j]) + \pi_{v_j}^{-(v_j, v_{j+1})} \geq \ell(P) \tag{2.6}$$

$$\ell(P[v_i, v_j]) + \pi_{v_j}^{-(v_j, v_{j+1})} \geq \pi_{v_i}^{-(v_i, v_{i+1})} \quad \forall i = 1, \ldots, j-1 \tag{2.7}$$

$$\ell(P[v_j, v_i]) + \pi_{v_i}^{-(v_i, v_{i+1})} \leq \pi_{v_j}^{-(v_j, v_{j+1})} \quad \forall i = j+1, \ldots k-1 \tag{2.8}$$

Claim 2. *$y(v_j) \leq \pi_{v_j}^{-(v_j, v_{j+1})}$.*

Proof Suppose to the contrary that:

$$y(v_j) > \pi_{v_j}^{-(v_j, v_{j+1})}. \tag{2.9}$$

First, observe that, in this case, since y is feasible, it follows that $y(v_j) \leq \ell(v_j, v_{j+1}) + y(v_{j+1})$.

We now show by induction that the followings hold indeed for each $i = j, \ldots, k-1$:

$$y(v_i) > \pi_{v_i}^{-(v_i,v_{i+1})} \tag{2.10}$$

$$y(v_j) \leq \ell(P[v_j, v_{i+1}]) + y(v_{i+1}) \tag{2.11}$$

We have already shown the base case $i = j$. By induction, assume now that (2.10) and (2.11) hold for each $i = j, j+1, \ldots, h$, for some h between j and $k-2$. By (2.8), we obtain:

$$\ell(P[v_j, v_{h+1}]) + \pi_{v_{h+1}}^{-(v_{h+1},v_{h+2})} \leq \pi_{v_j}^{-(v_j,v_{j+1})} < y(v_j) \leq \ell(P[v_j, v_{h+1}]) + y(v_{h+1}),$$

and therefore $y(v_{h+1}) > \pi_{v_{h+1}}^{-(v_{h+1},v_{h+2})}$ (i.e., (2.10) holds w.r.t. $h+1$). Since y is feasible, it also follows that $y(v_{h+1}) \leq \ell(v_{h+1}, v_{h+2}) + y(v_{h+2})$. Actually, it follows from (2.10) that $y(v_i) \leq \ell(v_i, v_{i+1}) + y(v_{i+1})$, for each $i = j, j+1, \ldots, h+1$: summing up these inequalities we have that $y(v_j) \leq \ell(P[v_j, v_{h+2}]) + y(v_{h+2})$ (i.e., (2.11) holds w.r.t. $h+1$).

So, in particular, we have that $y(v_j) \leq \ell(P[v_j, v_k]) + y(v_k) = \ell(P[v_j, v_k])$, and since $y(v_j) > \pi_{v_j}^{-(v_j,v_{j+1})}$, it follows that $\ell(P[v_j, v_k]) > \pi_{v_j}^{-(v_j v_{j+1})}$. But this is in contradiction with (2.6). *(End of the proof)*

Claim 3. $y(v_i) \leq \ell(P[v_i, v_j]) + \pi_{v_j}^{-(v_j,v_{j+1})}$, *for each* $i = 1, \ldots, j-1$.

Proof The proof is by induction. The base case $i = j-1$ is given by what follows: $y(v_{j-1}) \leq \max\left\{\ell(v_{j-1}, v_j) + y(v_j), \pi_{v_{j-1}}^{-(v_{j-1},v_j)}\right\} \leq \max\left\{\ell(v_{j-1}, v_j) + \pi_{v_j}^{-(v_j,v_{j+1})}, \pi_{v_{j-1}}^{-(v_{j-1},v_j)}\right\} =$ $= \ell(v_{j-1}, v_j) + \pi_{v_j}^{-(v_j,v_{j+1})}$, where the first inequality holds by the feasibility of y, the second follows from Claim 2 and the last from (2.7).

By induction, assume that (2.10) and (2.11) hold for each $i = j-1, j-2, \ldots, h$, for some $h \geq 2$. Then: $y(v_{h-1}) \leq \max\left\{\ell(v_{h-1}, v_h) + y(v_h), \pi_{v_{h-1}}^{-(v_{h-1},v_h)})\right\} \leq\leq$ $\max\left\{\ell(v_{h-1}, v_h) + \ell(P[v_h, v_j]) + \pi_{v_j}^{-(v_j,v_{j+1})}, \pi_{v_{h-1}}^{-(v_{h-1},v_h)})\right\} = = \ell(P[v_{h-1}, v_j]) + \pi_{v_j}^{-(v_j,v_{j+1})}$, where the first inequality holds for the feasibility of y, the second is by induction and the last equality follows from (2.7). *(End of the proof)*

Our statement follows then from Claim 3 with $i = 1$: $y(v_1) \leq \ell(P[v_1, v_j]) + \pi_{v_j}^{-(v_j,v_{j+1})} = \mathrm{Val}(P)$. □

Lemma 2.1.9 is what we need in order to prove the desired optimality conditions for ORP.

Lemma 2.1.10. *Let T be a spanning tree of G rooted in t; for any $v \in V \setminus \{t\}$, let $y(v)$ be the robust length of the unique v-t path on T and put $y(t) = 0$. Then T is a tree of optimal paths if and only if the so defined vector y is a feasible potential.*

Proof. Sufficiency easily follows from Lemma 2.1.9. Namely we have a feasible potential y and, for any $v \in V \setminus \{t\}$, we have a v-t path of robust length equal to $y(v)$: Lemma 2.1.9 guarantees that such paths are optimal.

For the necessity, let us assume by contradiction that y is not feasible. Then there exists at least one edge $(u, v) \in E$ such that $y(u) > \max\left\{\ell(u, v) + y(v), \pi_v^{-(u,v)})\right\}$. Let P_u and P_v be the respectively u-t and v-t paths such that $\mathrm{Val}(P_u) = y(u)$ and $\mathrm{Val}(P_v) = y(v)$. Be lemma 2.1.6 we have that $\max\left\{\ell(u, v) + y(v), \pi_v^{-(u,v)})\right\} = \mathrm{Val}((u, v) \oplus P_v)$ The above inequality implies that $\mathrm{Val}(P_u) > \mathrm{Val}((u, v) \oplus P_v)$ which is a contradiction, being P_u an optimal u-t path. □

We conclude this subsection observing another difference between ORP and the Shortest Path problem.

For both problems we deal with the concept of feasible potential, and for both problems, it is easily the case that a feasible potential is not bounded for some nodes. However while for the Shortest Path problem the nodes for which the potential is unbounded, are the ones not connected with the destination and that, consequently, can be deleted from the graph, for ORP the nodes for which the potential is unbounded are those nodes that are not 2-connected with the destination, but they cannot be canceled out since they might be used by detours. For example, looking at figure 2.1.2 we note that the nodes x_1, x_3, x_4 are not 2-connected with t but they belong to the detours of the path $P = \{v, u, x_2, t\}$. Indeed, they are spanned

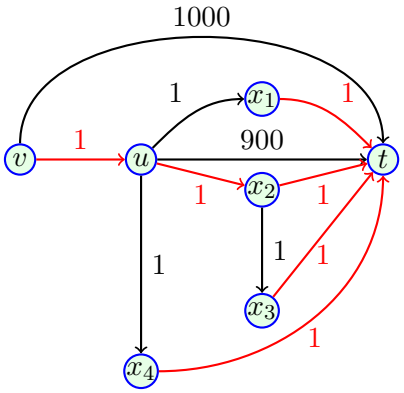

Figure 2.2:

by the red tree made by optimal paths and implementing the feasible potential $y(x_1) = y(x_3) = y(x_4) = \infty$, $y(v) = 1000$, $y(u) = 3$, $y(x_2) = 2$.

2.1.3 A label setting algorithm for ORP

What we show in this subsection is that one can compute in polynomial time a $v-t$ path with minimum robust length for each $v \in V$. Our algorithm for ORP uses the property established by the following lemma.

Lemma 2.1.11. *Let $U \subset V$, with $t \in U$, be the set of nodes x for which the optimal path P_x^* is known, and let $(v,u) \in E$ be an edge such that:*

$$(v,u) \in \underset{(z,w)\in E: w\in U, z\notin U}{\arg\min} \max\{\ell(z,w) + \mathrm{Val}(P_w^*), \pi_z^{-(z,w)}\}. \tag{2.12}$$

Then $P_v^* := (v,u) \oplus P_u^*$ is an optimal nominal v-t path.

Proof. Assume towards contradiction that there exists a path $P_v \in \mathcal{P}_{v,t}$ such that $\mathrm{Val}(P_v) < \mathrm{Val}(P_v^*)$. Let $w \in U$ and $z \in V \setminus U$ be two nodes such that $(z,w) \in E(P_v)$. Consider the partition of P_v given by $P_v = P_v[v,z] \oplus (z,w) \oplus P_v[w,t]$ (Note that if $w = t$ then $P_v[w,t] = \emptyset$ and if $z = v$ then $P_v[v,z] = \emptyset$).

By the choice of (v,u) we have

$$\max\{\ell(v,u) + \mathrm{Val}(P_u^*)), \pi_v^{-(v,u)}\} \leq \max\{\ell(z,w) + \mathrm{Val}(P_w^*), \pi_z^{-(z,w)}\},$$

which, by $\mathrm{Val}(P_v[w,t]) \geq \mathrm{Val}(P_w^*)$ implies

$$\max\{\ell(v,u) + \mathrm{Val}(P_u^*), \pi_v^{-(v,u)}\} \leq \max\{\ell(z,w) + \mathrm{Val}(P_v[w,t]), \pi_z^{-(z,w)}\}.$$

The latter inequality and Lemma 2.1.6 give $\mathrm{Val}(P_v^*) \leq \mathrm{Val}(P_v[z,t])$. On the other hand, from the properties of the Val function we have $\mathrm{Val}(P_v) \geq \mathrm{Val}(P_v[z,t])$. We conclude that $\mathrm{Val}(P_v^*) \leq \mathrm{Val}(P_v[z,t]) \leq \mathrm{Val}(P_v)$; a contradiction.

□

Our algorithm uses a label-setting approach, analogous to Dijkstra's algorithm for shortest paths. In other words, in every iteration the algorithm updates certain *temporary* labels for the vertices of the graph (representing an upper bound of the optimal solution), and fixes a *permanent* label to a single vertex v. This permanent label represents the connection cost of v by an optimal path to t. The algorithm

iteratively builds up a set U, consisting of all vertices for which the optimal path was already computed. Lemma 2.1.11 provides the required equation for our label-setting algorithm, suggesting that in any iteration one should look at the edges in $\delta^-(U)$, find the edge (v, u) satisfying 2.12 and label v as permanent. As for Dijkstra's algorithm, using suitable data structure, it is possible to obtain a more efficient implementation, whose statement is given as Algorithm 1.

Algorithm 1

1: Compute $\pi_u^{-(u,v)}$ for each $(u, v) \in E$.
2: $U = \emptyset; \quad W = V; \quad y'(t) = 0; \quad y'(u) = \infty \ \forall u \in V - t.$
3: $successor(u) = \text{NIL} \ \forall u \in V.$
4: **while** $U \neq V$ **do**
5: Find $u = \arg\min_{z \in W} y'(z)$.
6: $U = U + u; \quad W = W - u; \quad y(u) = y'(u).$
7: **for all** $(v, u) \in E$ with $v \in W$ **do**
8: **if** $y'(v) > \max\{\ell(v, u) + y(u), \pi_v^{-(v,u)}\}$ **then**
9: $y'(v) = \max\{\ell(v, u) + y(u), \pi_v^{-(v,u)}\}$.
10: $successor(v) = u.$
11: **end if**
12: **end for**
13: **end while**

As explained above, the correctness of the algorithm is a direct consequence of Lemma 2.1.11, but we prove it formally in the following lemma.

Lemma 2.1.12. *Algorithm 1 provides an optimal solution to ORP*

Proof. Let us specify that in this proof, given a node $u \in V$ selected at the step 5 of the algorithm, we call *scansion of* u the set of operations from step 7 to step 12. First of all, it is easy to check by induction on the size of the set U of scanned nodes, that at any stage of the algorithm we have the following properties holding:

1. if $y(v) \neq \infty$, then it is the robust length of simple v-t path;

2. if $successor(v) \neq NIL$, then *successor* defines a simple v-t path of robust length at most $y(v)$.

This implies that in order to conclude the proof, by lemma 2.1.10, we need to show that the obtained vector y is a feasible potential when the algorithm terminates.

Claim 4. *For each $w \in V$, let $y'(w)$ be the value of $y(w)$ when w is chosen to be scanned. If u is scanned before v, then $y'(u) \leq y'(v)$.*

Proof Suppose $y'(v) < y'(u)$ and let v be the earliest node scanned for which this is true. When u was chosen to be scanned, we had $y'(u) = y(u) \leq y(v)$, so $y(v)$ was lowered to a value $y'(v)$ less than $y'(u)$ after u was chosen to be scanned but before v was chosen. So $y(v)$ was lowered when some node w was scanned, and it was set to $y'(v) = \max\{\ell(v,w) + y'(w), \pi_v^{-(v,w)}\}$. By choice of v, we have $y'(w) \geq y'(u)$. We now have two cases:

1. if $\ell(v,w) + y'(w) \geq \pi_v^{-(v,w)}$, then $y'(v) = \ell(v,w) + y'(w)$, and since $\ell(v,w) \geq 0$ we have $y'(v) \geq y'(w) \geq y'(u)$, a contradiction.

2. if $\ell(v,w) + y'(w) < \pi_v^{-(v,w)}$, then $y'(v) = \pi_v^{-(v,w)} > \ell(v,w) + y'(w)$, and since $\ell(v,w) \geq 0$ we have $y'(v) \geq y'(w) \geq y'(u)$, a contradiction.*(End of the proof)*

We claim, now, that after all nodes are scanned, we have $y(w) \leq \max\{\ell(w,v) + y(v), \pi_w^{-(w,v)}\}$ for all $(w,v) \in E$. Suppose not. Since this was true when v was scanned, it must be that $y(v)$ was lowered after v, say while q was being scanned, and was set to $y(v) = \max\{\ell(v,q) + y'(q), \pi_v^{-(v,q)}\}$; furthermore, for Claim 4, we have $y'(v) \leq y'(q)$. We now have two cases:

1. if $\ell(v,q) + y'(q) \geq \pi_v^{-(v,q)}$, then $y(v) = \ell(v,q) + y'(q)$, and since $\ell(v,q) \geq 0$ we have $y(v) \geq y'(q) \geq y'(v)$, a contradiction.

2. if $\ell(v,q) + y'(q) < \pi_v^{-(v,q)}$, then $y(v) = \pi_v^{-(v,q)} > \ell(v,q) + y'(q)$, and since $\ell(v,q) \geq 0$ we have $y(v) > y'(q) \geq y'(v)$, a contradiction.

□

Consider the running time of Algorithm 1. For steps 2-10 we use the implementation of Fredman and Tarjan [13] for priority queues (heaps) called *Fibonacci Heaps*. We adopt here the same implementation that is used to obtain $O(m + n \log n)$ running time for Dijkstra's algorithm. We omit the details as they are identical to those in [13].

Consider next the implementation of step 1, which amounts to computing the values $\pi_u^{-(u,v)}$ for each $(u, v) \in E$. We start by computing the shortest path tree T in $O(m + n \log n)$ time from every node to t and this would trivially tells us the value of $\pi_u^{-(u,v)}$ for each edge (u, v) of the graph not in T.

We can hence concentrate our efforts on computing π-values for edges in the tree. Unfortunately for directed graphs, the best we can do is to compute the values $\pi_u^{-(u,v)}$ for $(u, v) \in T$ by performing $n - 1$ independent shortest path computations. This means that step 1 is the bottleneck of Algorithm 1 and that it can be trivially implemented in time $O(n\mathrm{SP}(n, m))$, where $\mathrm{SP}(n, m)$ is the complexity of a single shortest path computation on a graph with n nodes, m edges and nonnegative weights.

We are ready to state one of the main results of this section.

Theorem 2.1.13. *Given an instance of ORP the potential y and the corresponding paths can be computed in time $O(n\mathrm{SP}(n, m))$ in directed graphs.*

2.1.4 The undirected case

In this subsection we discuss ORP when the input graph $G = (V, E)$ is undirected. We indicate with $\{u, v\} \in E$ an undirected edge of G connecting two nodes $u, v \in V$.

The definitions introduced so far (detour, robust length, etc.) remain the same, and, more important, the described properties still hold; for example, if the graph in figure 2.1.1 was undirected the considerations made in subsection 2.1.1 in order to explain the weak optimality principle for ORP would still be valid.

Therefore we claim that Algorithm 1 works also for undirected graphs. The idea is, like it is usually done for the Shortest Path problem on undirected graph, to take any undirected edge $\{u, v\} \in E$ and split it into two directed edges (u, v) and (v, u)

with the same cost of the original edge and then run Algorithm 1 on this transformed graph.

Then main difference between the directed and undirected case is that for the second one we can provide an efficient implementation of step 1 of Algorithm 1 that relies on an algorithm of Nardelli, Proietti and Widmayer [20] for computing swap edges in graphs with respect to a shortest path tree.

In step 1 we need to compute, for any undirected edge $\{u, v\} \in E$, the values $\pi_u^{-(u,v)}$ and $\pi_v^{-(v,u)}$ representing the cost of the shortest path in $G - \{u, v\}$ from u to t and from v to t respectively (note, indeed, that the edge $\{u, v\}$ can be crossed both from u to v and from v to u).

We start by computing the undirected shortest path tree T in $O(m + n \log n)$ time from every node to t and this would trivially tell us the values of $\pi_u^{-(u,v)}$ and $\pi_v^{-(v,u)}$ for each edge $\{u, v\}$ of the graph not in T. It would also tell us the value $\pi_v^{-(v,u)}$ for each each edge $\{u, v\}$ of the graph in T, being v the closest node to t on the unique u-t path in T, so that we can concentrate our efforts on computing the values $\pi_u^{-(u,v)}$ for each each edge $\{u, v\}$ of the graph in T.

This problem is a well-known variant of *best swap edge* problem studied by Nardelli, Proietti and Widmayer [20], called the $\{r, v\}$-*Problem*. The authors give an algorithm for computing all such best swap edges with running time $O(m\alpha(m, n))$ for undirected graphs, where $\alpha(\cdot, \cdot)$ is the Inverse Ackermann function.

Therefore we obtain a procedure for step 1 with running time $O(m\alpha(n, m) + n \log n) = O(m + n \log n)$ [30]. Since for steps 2-10 we use the same implementation mentioned in the previous subsection, we have that the running time of Algorithm 1 for undirected graphs is $O(m + n \log n)$.

Theorem 2.1.14. *Given an instance of ORP the potential y and the corresponding paths can be computed in time $O(m + n \log n)$ in undirected graphs.*

We note that Bar-Noy and Schieber [5] sketch a similar algorithm with the same time bound for a problem that can be seen as a policy-based variant of ORP (see section 1.2). Their result is stated without proof, and we are not aware of a proof

that does not build on the result of Nardelli, Proietti and Widmayer [20], which appeared afterwards.

2.2 ORP with arbitrary costs

In this subsection we tackle ORP problem when we allow the edge-weights to be negative, that is we assume $\ell : E \to \mathbb{R}$. We also assume to be given a directed graph $G = (V, E)$, since we have already seen in subsection 2.1.4 how to extend the obtained results to undirected graphs. Recall that we fix a destination node $t \in V$ and we assume that at most one edge can fail.

We split this section in two cases: in a first moment we assume that there are no cycles with negative length in G and we hint an algorithmic approach to solve this case; in a second moment we make some considerations about the existence of a solution in the case where we allow the presence of cycles with negative length.

2.2.1 ORP with arbitrary costs in absence of negative cycles

We assume that there can be edges with negative cost, but that there are no cycles with negative length in G. First of all, we claim that also in this case we can assume without loss of generality that an optimal solution, if any, is a simple path. Even more, it is quite easy to see that almost everything we proved in section 2.1 still holds; the only part that must be reconsidered is the algorithmic one, since in the proof of the correctness of Algorithm 1 we use the nonnegativity of ℓ.

On the other hand, as just said, we can still use the idea of feasible potential (see definition 2.1.8) in order to obtain optimality conditions for our problem (see 2.1.10). Then it is quite natural to raise the question whether it is possible to develop for ORP with arbitrary costs, but no negative cycles, an algorithm like the Ford-Bellman one for Shortest Path problem, that is an algorithm that, until a feasible potential is not obtained, iteratively finds an edge not satisfying equation (2.3) and corrects it.

We conjecture that the following algorithm solves ORP problem under the assumption that there are arbitrary costs on the edges, but there are no cycles with

negative length:

Algorithm 2

1: Initialize y; $y(t) = 0$; $y(u) = \infty \ \forall u \in V - t$.

2: Initialize $successor$; $successor(u) = \text{NIL} \ \forall u \in V$.

3: **while** y is not a feasible potential **do**

4: Find $(v, u) \in E$ such that $y(v) > \max\{\ell(v, u) + y(u), \pi_v^{-(v,u)}\}$.

5: Put $y(v) = \max\{\ell(v, u) + y(u), \pi_v^{-(v,u)}\}$ and $successor(v) = u$.

6: **end while**

We refer to future work for the proof of what follows: Algorithm 2 is correct and that it terminates after a finite number of iterations; providing a bound to such number is an interesting open question.

2.2.2 ORP with arbitrary costs in presence of negative cycles

We now turn our attention to the case where G can have negative cycles. It is well known that, given an edge-weighted graph G, we can find a solution for the Shortest Path problem (or certify that such solution does not exist) in polynomial time if and only if G has no cycle with negative length. As it is shown in the figure 2.3 this is not true any longer for ORP, since, even though there are two negative cycles, we can see that in this example the red edges provide an optimal solution to ORP with bounded robust values (in particular, we note that in this example the solution is provided as a tree rooted in t as it happens for ORP with nonnegative costs, see the end of subsection 2.1.2). The example of figure 2.3 shows also that it might be possible to find a solution to ORP even though the graph is such that a negative cycle exists in any possible failure scenario.

It would be useful to find a result as the one stated before for the Shortest Path problem, that is a sufficient and necessary condition for the existence of a solution to ORP in presence of arbitrary costs and cycles with negative length. Let us focus on the example shown in figure 2.4; first of all we note that for the nodes a and d there exists a solution with bounded robust value, that is the edges (a, t) and

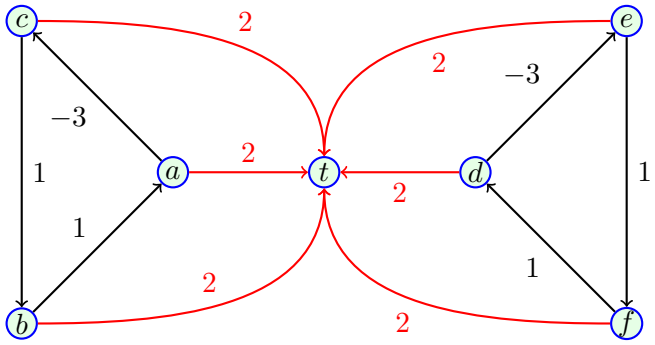

Figure 2.3: A graph with negative cycles for which a solution to ORP exists (red edges). Note that this graph is also such that in any failure scenario there exists a negative cycle.

(d, t) respectively. On the other hand, this is not true if we take as origin node b or node c and the reason is the following: in both cases, we can reach the cycle $\mathcal{C} = \{(b, c), (c, b)\}$ of length -2 both in the nominal scenario (no failures) and in all those scenarios in which an edge not belonging to $\mathcal{C}$ fails; moreover $\mathcal{C}$ has the property that if one of its edges fails, a cycle with negative length is still reachable in the remaining graph: in particular, the cycle $\{(b, a), (a, c), (c, b)\}$ with length -1 is reachable if (b, c) fails and the cycle $\{(b, c), (c, d), (d, b)\}$ with length -1 is reachable if (c, b) fails.

This example hints that it is easy to proof what follows: *if there exists a finite solution to ORP then G has no cycle $\mathcal{C}$ with negative length and such that for each (u, v) of $\mathcal{C}$ no cycle with negative length is reachable from u in $G - (u, v)$ (in other words $\pi_u^{-(u,v)}$ is not $-\infty$).*

We conjecture that the stated condition is also sufficient for the existence of a finite solution to ORP. We leave the proof of this fact as an open point.

We conclude this section stressing that ORP with arbitrary costs and no restriction about cycles with negative length presents an important difference with respect both the Shortest Path problem and ORP with nonnegative costs. In this case, indeed, we cannot restrict without loss of generality the domain of optimal solutions to simple paths, because the optimal solution for a given source node $v \in V$ can

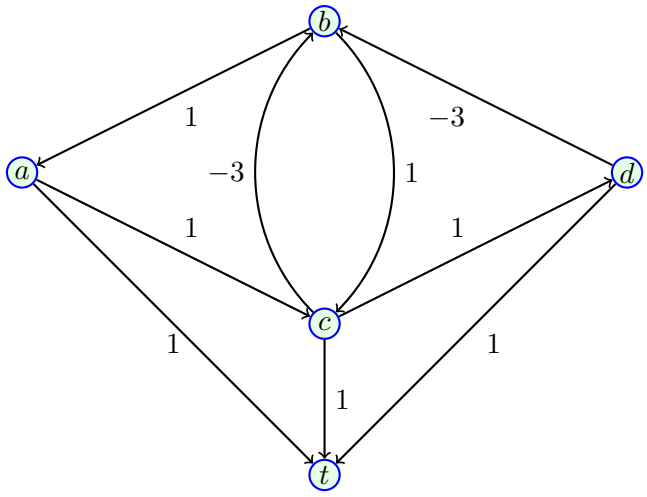

Figure 2.4: A graph for which there is no a solution to ORP for each origin node.

have cycles, as it is shown in the example presented in figure 2.5. In this example, indeed, it is easy to check that the simple v-t path with minimum robust length is $P_1 = \{v, a, b, d, t\}$, being $\text{Val}(P_1) = 5$, while the path $P_2 = \{v, a, b, c, a, b, d, t\}$ has $\text{Val}(P_2) = 4$ and it is not simple.

Moreover figure 2.5 also hints that in ORP with arbitrary costs, in order to find a solution with bounded robust value, we can restrict our focus on no-simple paths that cycle a finte number of times without loss of generality. Consider the no-simple path P_2, for which $\text{Val}(P_2) = 4$. It holds that $bottleneck(P_2) = \emptyset$, but we note that also the scenario in which the edge (v, a) fails has cost 4; this implies that the robust length of P_2 cannot be less than 4. We also note that if the nominal scenario corresponded to the simple path P_1 it would cost 5, but traversing once the cycle $\{(a, b), (b, c), (c, a)\}$ it goes down to 4. Cycling an infinite number of times, the nominal cost would become arbitrary small, but it would be useless since, as we have observed before, we would still have $\text{Val}(P_2) = 4$.

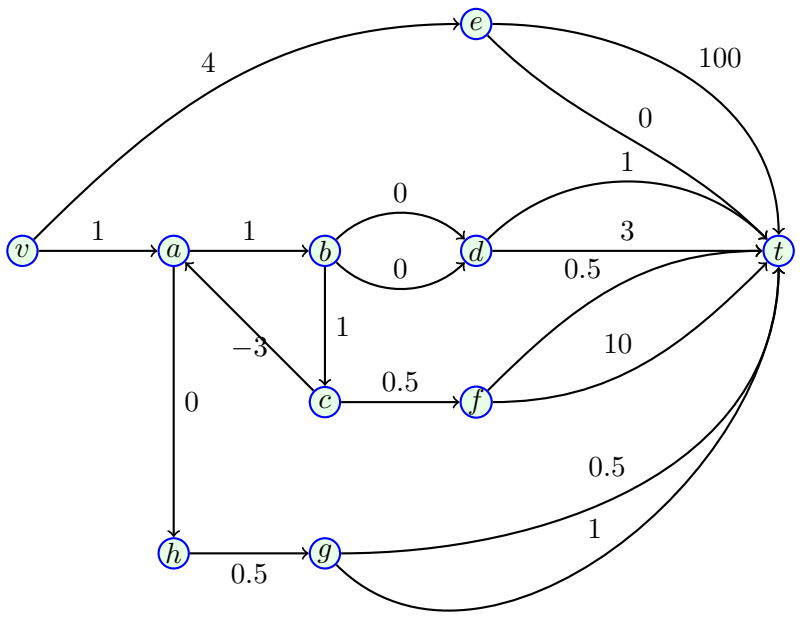

Figure 2.5: An example showing that, in presence of negative costs and negative cycles, the solution can be a no-simple path.

2.3 Bi-objective ORP

In this section we turn to a natural question linking ORP and the Shortest Path problem. Consider an instance of s-t ORP for which the optimal nominal path is not unique. While all optimal paths P have the same robust length, let us say B^*, they might differ in terms of their nominal length $\ell(P)$. In those application in which failures are no that likely, we might thus be interested in obtaining a path attaining the optimal robust length with minimum nominal length. In general, one can consider the following bi-objective problem for any bound $B \geq B^*$.

$$z(s, B) = \min_{P \in \mathcal{P}_{s,t},\ \mathrm{Val}(P) \leq B} \ell(P).$$

The latter problem asks to find a *Pareto-optimal* s-t path in G with objective functions robust length and nominal length. We call this problem *Bi-objective ORP*. Bi-objective ORP bears resemblance to the Bi-objective Shortest Path problem [15]. In the latter problem one seeks to obtain a Pareto-optimal s-t path in the graph with objective functions corresponding to ordinary length with respect to two different length functions. In this section we show that the two problems differ significantly in terms of their complexity. Concretely, we will show that a solution to bi-objective ORP and the entire Pareto front can be found in polynomial time. This contrasts to the Bi-objective Shortest Path problem, which is NP-hard, and its Pareto front can be of exponential size in the size of the graph. Our first result is:

Theorem 2.3.1. *Bi-objective ORP can be solved in time $O(m+n\log n)$ in undirected graphs and in time $O(mn + n^2 \log n)$ in directed graphs.*

Proof. We prove that Algorithm 3 solves the problem in the stated time. We adopt the implementation we used to obtain the same running time for Algorithm 1. The details are identical, thus omitted. It remains to prove its correctness.

We define first a value function for partial paths. Concretely, for $u \in V$ and s-u path P define

$$\mathrm{L}(P) = \max\{\ell(P), \max_{e \in P} \ell(P^{-e})\}.$$

Observe that $\mathrm{L}(P) = \mathrm{Val}(P)$ holds for s-t paths.

We prove by induction on the cardinality of S that every node $u \in S$ during the execution of the algorithm satisfies $d(u) = z(u)$, where

$$z(u) = \min_{P \in \mathcal{P}_{s,u} : \mathrm{L}(P) \leq B} \ell(P).$$

This implies in particular that for $u = t$ we obtain the desired result.

The claim is obviously true in the first iteration. Assume this to be true until the k-th iteration. Consider the node $u \in \bar{S}$ chosen in the $k+1$-st iteration in the algorithm. Assume towards contradiction that $d(u) > z(u)$, and let $P^* \in \mathcal{P}_{s,u}$ be a path attaining $\mathrm{L}(P^*) = z(u)$. Consider the first edge $(x, y) \in P^*$ in the traversal from s to u with the property that $x \in S$ and $y \in \bar{S}$. Note that by the inductive assumption we have $d(y) \leq z(x) + \ell(x, y)$. Indeed, since $x \in S$ we know that in the iteration that x was added to S one had $d(x) = z(x)$. Furthermore, in step 9 of this iteration, $d(y)$ was updated using the edge (x, y).

It remains to note that $d(y) \leq \mathrm{L}(P^*) < d(u)$, since $\mathrm{L}(Q) \geq \mathrm{L}(Q[s, w])$ holds whenever $Q \in \mathcal{P}_{s,u}$ and $w \in V(Q)$, while $P^*[x, t]$ can always be assumed to satisfy $\mathrm{L}(P^*[s, x]) = z(x)$. This contradicts the choice of u in step 4 of the algorithm.

□

Let us turn to the problem of computing the Pareto front. Recall that a *Pareto front* F of a bi-objective optimization problem with objective functions f and g is a set of Pareto-optimal solutions to the problem with the property that, for every other solution X, there exists a solution $Y \in \mathrm{F}$ such that $f(X) \geq f(Y)$ and $g(X) \geq g(Y)$. A Pareto front F for an instance of Bi-objective ORP is a set of paths, such that, for every Pareto-optimal path P, there exists a path in F with not longer robust length and not longer nominal length. In general, having an entire Pareto front at hand is of course advantageous in practical applications, as it gives the decision maker a complete list of efficient strategies (see the plot in figure 2.6).

Algorithm 3

$S = \emptyset$; $\quad \bar{S} = V$

$d(s) = 0$; $\quad d(v) = \infty \ \forall v \in V - s$

while $t \notin S$ **do**

  Find $u = \arg\min_{z \in \bar{S}} d(z)$

  $S = S + u$

  $\bar{S} = \bar{S} - u$

  for $v \in N(u) \setminus S$ **do**

    if $d(u) + \ell(u, v) \leq d(v)$ **and** $d(u) + \pi_u^{-(u,v)} \leq B$ **then**

      $d(v) = d(u) + \ell(u, v)$

    end if

  end for

end while

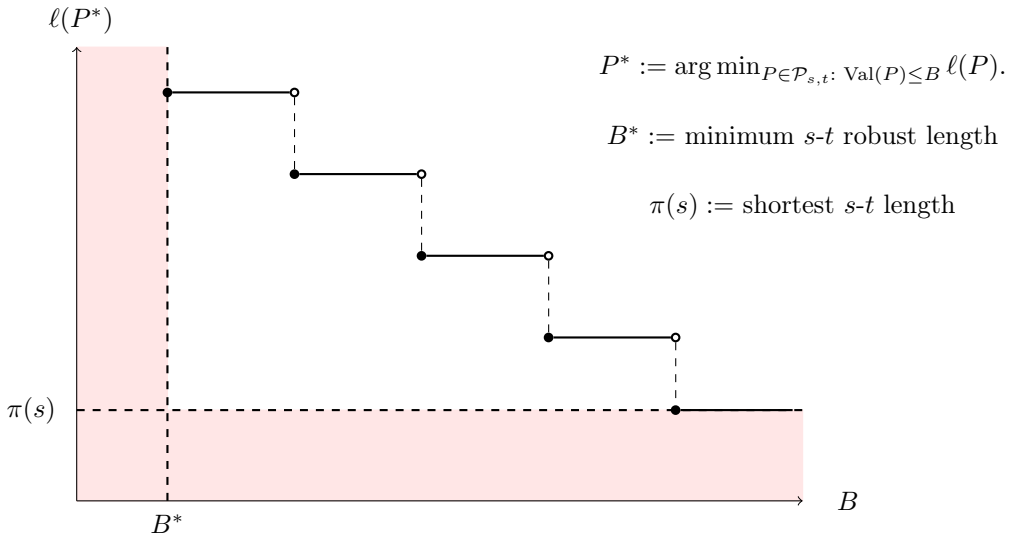

Figure 2.6: Pareto front of Bi-objective ORP.

The following theorem asserts that every Bi-objective ORP instance has a Pareto front with a linear number of paths. The front can be found in polynomial time using Algorithm 4 (note that, for undirected graphs, the algorithm is slightly different).

Algorithm 4

1: $H = G$; F $= \emptyset$.
2: **while** s and t are connected in H **do**
3: Find a shortest s-t path P in H and add it to F.
4: Find a critical edge $e \in E(H)$ (with $\mathrm{Val}(P) = \ell(P^{-e})$) and remove it from H.
5: **end while**
6: Remove from F all dominated paths.
7: Return F.

Theorem 2.3.2. *Every instance of Bi-objective ORP admits a Pareto front* F *with at most* $2m$ *paths (m paths in directed graphs). The Pareto front can be found in time* $O(m^2 + mn \log n)$.

Proof. We prove the theorem for directed graphs first, and then describe a simple adaptation for undirected graphs. We claim that Algorithm 4 builds a Pareto front for the given instance of Bi-objective ORP. This claim clearly implies both the bound on |F| and the bound on the running time of the algorithm. We stress that in step 4 of the algorithm the robust value of the path is computed according to the original graph G.

Let $k \leq m$ denote the number of iterations performed by the algorithm. For $i \leq k$ let $E_i = \{e_1, \cdots, e_i\}$ be a set of edges removed after i iterations of the algorithm (with e_i - the edge removed in the i-th iteration), and P_i be the path obtained in the i-th iteration (such that $e_i \in P_i$ critical). Furthermore, we assume without loss of generality that $e_i \in P_i$. Indeed, since P_i is a shortest path in $E \setminus E_{i-1}$, if some edge $e \notin P_i$ is critical for P_i, then $\mathrm{Val}(P_i) = \ell(P_i) = \ell(P_i{}^{-f})$ for every $f \in E \setminus E_{i-1}$.

We claim that every s-t path P is dominated by some path P_i, with $i \leq k$, namely $\mathrm{Val}(P_i) \leq \mathrm{Val}(P)$ and $\ell(P_i) \leq \ell(P)$. Let P be an arbitrary s-t path. Note first that

since E_k is an s-t cut in G we have $P \cap E_k \neq \emptyset$. Define $j = \min_{r \leq k}\{e_r \in P\}$. To this end consider P_j, the path obtained by the algorithm in the j-th iteration. We know that $P \subset E \setminus \{e_1, \cdots, e_{j-1}\}$, thus by choice of P_j in the algorithm as the shortest path in $G - E_{j-1}$ we have $\ell(P_j) \leq \ell(P)$. Let v be the endpoint of e_j closer to s on P and P_j (it is the same endpoint since G is directed). From the fact that P_j is a shortest path we also know that $\ell(P_j[s,v]) \leq \ell(P[s,v])$, thus from criticality of e_j for P_j we obtain

$$\mathrm{Val}(P) \geq \ell(P^{-e_j}) = \ell(P[s,v]) + \pi_v^{-e_j} \geq \ell(P_j[s,v]) + \pi_v^{-e_j} = \mathrm{Val}(P_j).$$

We conclude that P is dominated by P_j. This concludes the proof for directed graphs.

In undirected graphs is might hold that both P and P_j use the same edge, but in opposite directions. To this end one can make a simple adaptation of the algorithm. We replace every undirected edge with two directed edges in opposite directions, and perform Algorithm 4 on this new graph. One can easily see that the result carries through by using this graph. We stress however, that in step 4 of the algorithm we still use the original (undirected) graph. The result now follows from the fact the new graph has $2m$ edges. □

Let us observe that the complexity bounds stated in Theorem 2.3.2 are due to a pretty rough implementation of Algorithm 4; indeed, we basically run $O(m)$ times the same algorithm on slightly different networks, so we believe to be likely that a smarter and more efficient implementation can be found. We leave this as an open question.

Finally, observe that Algorithm 4 is also an algorithm for ORP, as for any Pareto front F we have $y(s) = \min_{P \in \mathrm{F}} \mathrm{Val}(P)$. This algorithm can be particularly interesting for solving ORP in sparse directed graphs, where the size of the Pareto front might compare favorably with the number of nodes in the graph.

2.4 k-ORP

Let us formally define k-ORP, the online replacement path problem with the assumption that a bounded number k of edges can fail. We refer to the parameter k as the *failure parameter*. In k-ORP a *scenario* corresponds to a removal of any k of the edges in the graph. In this setup it is no longer convenient to describe the problem in terms of paths and detours. Instead we introduce the notion of a *routing strategy*.

A routing strategy $\phi : 2^E \times V \to V$ is a function which, given a subset $F' \subset E$ of known failed links and a vertex $v \in V$, returns a vertex $u \in V$. We call $E' = E \backslash F'$ the set of *active edges*. We are assuming the existence of a certain governing mechanism, which takes as input a routing strategy and executes it on a given instance. Provided with a routing strategy ϕ, this mechanism iteratively moves from the current vertex u to the vertex $v = \phi(F', u)$, where F' is the set of failed links probed so far. The process starts at a given origin s with $F' = \emptyset$ and ends when t is reached.

Since ϕ is deterministic, this process defines a unique, possibly infinite, walk $\theta_\phi(u, E, F)$ in G for each origin $u \in V$ and every scenario $F \subset E$ with $|F| \leq k$. We remark that if G contains at least $k+1$ edge-disjoint s-t paths, there exists a routing strategy, which does not cycle.

Definition 2.4.1. *Given $E' \subset E$, a vertex $u \in V$ and a routing strategy ϕ, the robust value of ϕ with respect to E' and u is defined as*

$$\mathrm{RVal}(u, E', \phi) = \max_{F \subseteq E', |F| \leq k} \ell(\theta_\phi(u, E', F)).$$

Furthermore, given a node $u \in V$, the failure parameter k and $E' \subset E$, we define:

$$y^k(u, E') = \min_\phi \mathrm{RVal}(u, E', \phi).$$

Finally, k-ORP is to find an *optimal routing strategy* ϕ^*, which minimizes $\mathrm{RVal}(s, E, \phi)$, namely to solve

$$\phi^* = \arg\min_\phi \mathrm{RVal}(s, E, R).$$

The following relation, which is a generalization of (2.12), gives rise to a simple recursive algorithm for k-ORP. For each $v \in V$ it holds:

$$y^k(v,E) = \min_{v:(v,u)\in E}\{\max\{\ell(v,u) + y^k(u,E), y^{k-1}(v, E \setminus (v,u))\}\}. \tag{2.13}$$

The correctness of this relation can be proved by induction on k, using the arguments in Section 2.1.

Algorithm 5 solves (2.13) for every $v \in V - t$ and computes the corresponding optimal routing strategy. Note that Algorithm 5 is identical to Algorithm 1 when $k = 1$. To formally prove the correctness of Algorithm 5 one needs to state equivalent monotonicity properties, as well as analogues of Lemma 2.1.6 and Lemma 2.1.11. They are however omitted, as they are identical to those of Section 2.1.

Algorithm 5

1: Compute $y^{k-1}(u, E \setminus (u,v))$ for each $uv \in E$.
2: $U = \emptyset$.
3: $W = V$.
4: $y^k(t,E) = 0$.
5: $y^k(u,E) = \infty$ for each $u \in W$.
6: **while** $U \neq V$ **do**
7: Find $u = \arg\min_{z\in W}\{y^k(z,E)\}$.
8: $U = U + u$.
9: $W = W - u$.
10: **for all** $(v,u) \in E$ such that $v \in W$ **do**
11: **if** $y^k(v,E) > \max\{\ell(u,v) + y^k(u,E), y^{k-1}(v, E \setminus \{(v,u)\})\}$ **then**
12: $y^k(v,E) = \max\{\ell(u,v) + y^k(u,E), y^{k-1}(v, E \setminus \{(v,u)\})\}$.
13: **end if**
14: **end for**
15: **end while**

We conclude by bounding the complexity of a naive implementation of this algorithm. Let $T(m,n,k)$ denote the running time of the algorithm on a graph

(either directed or undirected) with n vertices and m edges and failure parameter k. The results in Section 2.1 give $T(n,m,1) = O(m + n\log n)$ for undirected graph and $T(n,m,1) = O(n\mathrm{SP}(n,m))$ for directed graphs. For $k > 1$ we have $T(m,n,k) = O(m + n\log n) + mT(m-1,n,k-1) = O(m^{k-1}T(n,m,1))$. This finally gives the following theorem.

Theorem 2.4.2. *k-ORP can be solved on undirected graphs in time $O(m^k + m^{k-1}n\log n)$ and on directed graphs in time $O(m^{k-1}n\mathrm{SP}(n,m))$.*

2.4.1 The Shortest Path Heuristics

It is common to use heuristics which rely on shortest paths in routing algorithms. In this section we show that a naive shortest path routing strategy performs very poorly for k-ORP. In fact the approximation guarantee it provides grows exponentially with the adversarial budget k. To this end we define more formally the shortest path heuristics, which we denote by ϕ_{SP}. The routing strategy ϕ_{SP} works as follows. At each vertex $u \in V$ and given a set of known failed edges F', ϕ_{SP} tries to route the package along a shortest path in the remaining graph $G - F'$. In the following lemma we show that ϕ_{SP} is a factor $2^{k+1} - 1$ approximation for the optimal routing strategy, in the presence of at most k failed edges.

Lemma 2.4.3. *Let $\mathcal{I} = (G,s,t)$ be an instance of k-ORP. Then*

$$\mathrm{RVal}(s,E,\phi_{SP}) \leq (2^{k+1}-1)y^k(s,E).$$

Proof. Let $OPT = y^k(s,E)$, $\phi^* = \arg\min_\phi \mathrm{RVal}(s,E,\phi)$ and $Q^0 = \theta_{\phi^*}(s,E,\emptyset)$ be the corresponding nominal path. Consider a set F of failed edges with $|F| \leq k$. The routing strategy ϕ_{SP} defines a certain walk $\omega = (s = u_1, u_2, \cdots, u_m = t)$ in $G - F$. We divide our analysis according to the number of failed edges encountered by ϕ_{SP}. We prove by induction on i, that if the routing strategy encountered a total of $i \leq k$ failed edges then

$$\ell(\omega) \leq (2^{i+1}-1)OPT. \tag{2.14}$$

The base case $i = 0$ corresponds to scenarios in which no failed edge is encountered in the routing. In this case ω is simply a shortest s-t path in G, hence $\ell(\omega) \leq OPT$, as required. Assume next that (2.14) holds for every $j < i$ and consider the case that the routing encounters exactly i failed edges. We can assume without loss of generality that all edges of F were probed and $|F| = i$.

Let $r < m$ be such that u_r is the vertex incident to the i'th failed edge in the routing. In other words, before reaching u_r, the routing probed exactly $i - 1$ failed edges. Let $e = \{u_r, w\}$ be the i'th failed edge probed by the routing strategy. Consider an execution of the routing strategy on the same instance with the different failure scenario $F' = F - e$. The resulting walk ω' will have the first r vertices in common with ω, namely the sub-walk $\sigma = (s = u_1, \cdots, u_r)$ will appear in both walks. By the inductive hypothesis we have that $\ell(\sigma) \leq \ell(\omega') \leq (2^i - 1)OPT$. It remains to bound the length of the tail of ω from u_r until $u_m = t$ to complete the proof. To this end recall that ϕ_{SP} routes the package along the shortest remaining path in the graph. Since the last failure encountered by ϕ_{SP} is e, the remaining path is simply the shortest u_r-t path in $G - F$. To bound the length of this path we construct a u_r-t walk θ as follows. First θ traces the entire route taken by ϕ_{SP} back to s and then uses the walk that ϕ^* would use to reach t from s in the scenario F. Clearly we have $\ell(\theta) \leq (2^i - 1)OPT + OPT$. Furthermore, this walk is intact in $G - F$. This gives the required bound $\ell(\omega) \leq (2^i - 1)OPT + (2^i - 1)OPT + OPT = (2^{i+1} - 1)OPT$ and finishes the proof. □

The bound obtained in Lemma 2.4.3 seems crude at first glance. In particular, in the inductive step we follow the entire walk performed so far backwards to reach s and start over. In the following example we show that the bound of Lemma 2.4.3 is tight.

Esempio 2.4.4. *Let $M \in \mathbb{Z}_+$ be a large integer. Consider the following instance $\mathcal{I} = (G, s, t)$ of k-ORP. The graph contains $k + 1$ parallel edges connecting s and t with length $M + 1$. In addition the graph contains a path $(s, u_1, \cdots, u_k)$ of length $k + 1$. The edge su_1 has length M and every edge $u_i u_{i+1}$ has length $2^i M$. Finally,*

the vertices $u_1, \cdots, u_k$ are connected to t with edges of length zero. The construction is illustrated in Figure 2.7.

Consider the failure scenario, which fails all edges $u_i t$ for $i \in [k]$. The routing strategy ϕ_{SP} will follow the path $(s, u_1, \cdots, u_k)$, then follow it back to s and then take one of the edges with length $M+1$ to t. The total length of this walk is $2(M + 2M + \cdots + 2^k M) + M + 1 = (2^{k+1} - 1)M + 1$. At the same time the optimal routing strategy routs the package along the edges with length $M+1$ with a worst-case cost of $M+1$. The ratio between the two numbers tends to $2^{k+1} - 1$ as M tends to infinity.

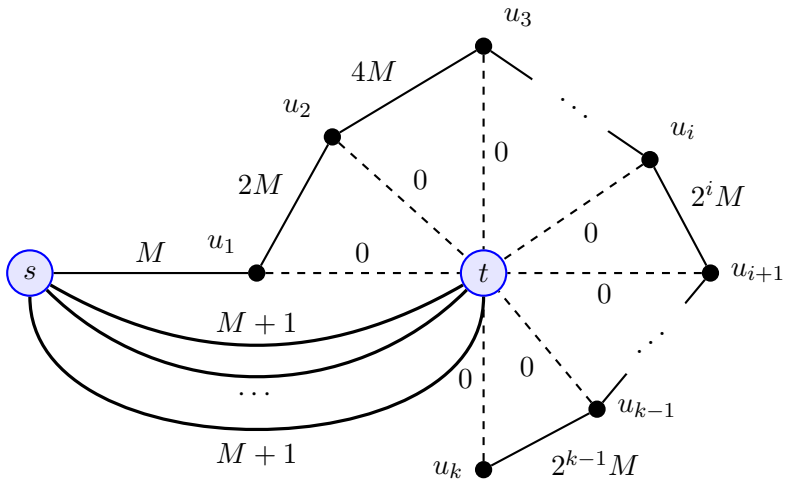

Figure 2.7: A bad example for the shortest path heuristic. The dashed edges correspond to the worst case scenario.

Chapter 3

ORP Generalizations

We have stressed several times so far that in ORP the Routing Mechanism discovers the failure of an edge only when it tries to cross it, providing many examples of applications where such model can be quite suitable. In this Chapter we aim at providing some generalizations of ORP, potentially widening its applicability.

The following example gives an idea about the aspect just mentioned. Imagine that a traveller using Public Transport makes a query to the system via a mobile device in order to reach his/her destination; of course, he/she is provided with a certain journey and starts to follow it. If something of unexpected happens affecting such journey, it is reasonable to think that the system itself can react real-time and send an alert to the traveller's mobile device suggesting 'in advance' a re-planned journey.

Formally defining what we mean by 'in advance', we end up with a list of possible interesting problems that we present and investigate in this Chapter. In section 3.1 we discuss the *k-Hop ORP* problem and the *Radius ORP* problem, while in section 3.2 we tackle the *Strong k-Hop ORP*; as just mentioned these problems differ in the way the RM discovers the failed edge, if any.

3.1 k-Hop ORP

In this section we study the k-Hop ORP problem. As in ORP, we assume that at most one edge of the graph can fail. Furthermore, we assume that we are given an integer k between 0 and $n-1$ and that now the Routing Mechanism (RM) is informed about the failure of edge e as soon as it reaches a node that is k (or fewer) hops away from e on the nominal path P. In particular, if $e \notin P$, the RM won't be aware of the failure of e. It is easy to see that 0-Hop ORP is simply ORP and that $(n-1)$-Hop ORP is equivalent to MVA. For $k \in \{1, \cdots, n-2\}$ we obtain an interesting continuum of problems between ORP and MVA.

Apparently this new setting changes dramatically the problem. In fact, consider a (nominal) s-t path P and an edge e which belongs to P. Denote by $v(P,e)$ the first node of P that 'sees' the failure of e (note that $v(P,e) = s$ if e is at most k-hop away from s on P). More formally:

Definition 3.1.1. *Given a path $P \in \mathcal{P}_{s,t}$, an edge $e = (u,u') \in E(P)$ and an integer $k \in \{1,\ldots,n-1\}$, we define $v(P,e) \in V(P)$ as the first node of P, starting from s, such that $|E(P[v(P,e),u])| \leq k$.*

Being aware of the failure of e already at $v(P,e)$ allows the RM to take a detour before getting to e, as for ORP. This justifies the following redefinition of detours:

Definition 3.1.2. *The detour P^{-e} of a path $P \in \mathcal{P}_{s,t}$ associated with an edge $e \in E$ is:*

- *the walk $P[s, v(P,e)] \oplus Q^{-e}_{v(P,e)}$, if $e \in E(P)$;*
- *P, otherwise.*

The length of a detour is given by:

- $\ell(P^{-e}) = \ell(P[s, v(P,e)]) + \pi^{-e}_{v(P,e)}$, *if $e \in E(P)$;*
- $\ell(P^{-e}) = \ell(P)$, *otherwise.*

We accordingly define the *k-Hop robust length*:

Definition 3.1.3. *Given a node $s \in V$, the k-Hop robust length of the s-t path P is*

$$\mathrm{Val}^k(P) = \max_{e \in E} \ell(P^{-e}).$$

Moreover, we introduce for every $u \in V - t$ the following value:

$$\pi^k_{rob}(u) = \min_{P \in \mathcal{P}_{u,t}} \mathrm{Val}^k(P),$$

We can now formally define k-Hop ORP:

Definition 3.1.4 (The k-Hop Online Replacement Path problem)**.** ***Given:*** *an edge-weighted directed graph $G = (V, E, \ell)$, where $\ell : E \to \mathbb{R}^+$ and a destination $t \in V$.* ***Find:*** *for every $v \in V$ an optimal v-t path, namely a path P minimizing $\mathrm{Val}^k(P)$ over all paths $P \in \mathcal{P}_{v,t}$.*

Our main result in this section is a label-setting algorithm for this problem. Obtaining this algorithm is however a more challenging task than that of obtaining such an algorithm for ORP. In particular, we will see that while there always exists a tree of optimal nominal paths, the k-hop potential needs not be a monotonic function along the paths of this tree. This property contrasts with the structure of optimal solutions to ORP.

Indeed, in the example in figure 3.1 we can see that Val^k has not to be monotonic with respect to proper subpaths. Consider the path $\tilde{P} = \{s, b, c, t\}$ and its subpath $P = \{b, c, t\}$ and evaluate their 1-Hop robust length (that is we assume k=1). It is easy to check that the edge (b, c) is the critical one for P and consequently $\mathrm{Val}^1(P) = 24$, while edge (c, t) is critical for $\tilde{P}$ and consequently $\mathrm{Val}^1(\tilde{P}) = 23$, so that $\mathrm{Val}^1(\tilde{P}) < \mathrm{Val}^1(P)$ even though P is a proper subpath of $\tilde{P}$. Basically, this happens because the scenario in which (b, c) fails has cost 5 in the computation of $\mathrm{Val}^1(\tilde{P})$, while it has cost 24 in the computation of $\mathrm{Val}^1(P)$; in the first case, indeed, such edge can be already seen from s giving us the possibility to take a better detour with respect the second case, where we realize its failure only when we stand at node b.

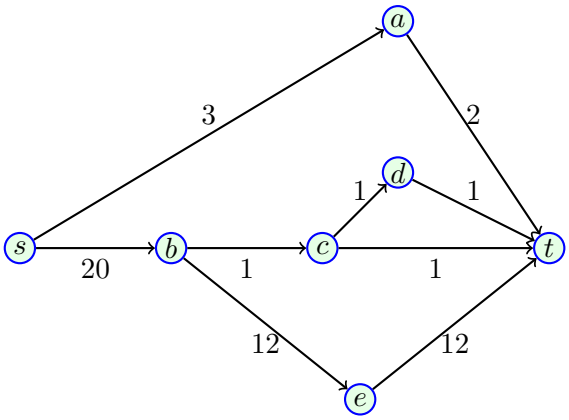

Figure 3.1: An example showing that Val^k is not monotonic with respect to proper subpaths. Assume $k = 1$.

The key property of k-Hop ORP is stated in the following lemma, in which we show that the weak optimality principle proven for ORP (see lemma 2.1.7) can be generalized for k-Hop ORP.

Lemma 3.1.5 (Generalized weak optimality principle)**.** *Let $P_u \in \mathcal{P}_{u,t}$ be an optimal path from u, let $v \in V(P_u)$ and let $P_v \in \mathcal{P}_{v,t}$ be an optimal path from v. Then the path $P'_u = P_u[u, v] \oplus P_v$ satisfies $\mathrm{Val}^k(P'_u) = \mathrm{Val}^k(P_u)$, namely it is also optimal from u.*

Proof. Consider a partition $E = X \cup Y \cup Z$ defined for P_v as follows. We set $X = E \setminus P_v$, $Y = \{e \in P_v : v = v(P_v, e)\}$ and $Z = \{e \in P_v : v \neq v(P_v, e)\}$. For $T \in \{X, Y, Z\}$ define $\bar{T} = \max_{e \in T} \ell({P_v}^{-e})$.

Let $P'_v = P_u[v, t]$ and note that $P'_v \in \mathcal{P}_{v,t}$. We obtain a similar partition $E = X' \cup Y' \cup Z'$, as well as values $\bar{X}', \bar{Y}', \bar{Z}'$, corresponding to P', where every set is defined as above, but with P_v replaced by P'_v. Observe that $\mathrm{Val}^k(P_v) = \max\{\bar{X}, \bar{Y}, \bar{Z}\}$ and $\mathrm{Val}^k(P'_v) = \max\{\bar{X}', \bar{Y}', \bar{Z}'\}$. Furthermore, note that $\bar{X} = \ell(P_v)$ and $\bar{X}' = \ell(P'_v)$.

Claim 1. $\max\{\bar{X}', \bar{Z}'\} \geq \max\{\bar{X}, \bar{Y}, \bar{Z}\}$.

Proof of Claim 1. Since P_v is optimal from v we have $\mathrm{Val}^k(P_v) \leq \mathrm{Val}^k(P'_v)$, thus $\max\{\bar{X}', \bar{Y}', \bar{Z}'\} \geq \max\{\bar{X}, \bar{Y}, \bar{Z}\}$. To prove the claim it suffices to show

$\bar{Y}' \leq \max\{\bar{X}, \bar{Y}, \bar{Z}\}$. By definition of $\bar{Y}'$, this amounts to showing that $\pi_v^{-e} \leq \max\{\bar{X}, \bar{Y}, \bar{Z}\}$ for every $e \in Y = \{e \in P'_v : v = v(P'_v, e)\}$. Consider any such $e \in Y$. Assume first $e \notin P_v$. In this case we have $\pi_v^{-e} \leq \bar{X} = \ell(P_v)$, since P_v is a path in $G - e$. In the remaining case $e \in P_v$ we have $\max\{\bar{Y}, \bar{Z}\} \geq \ell({P_v}^{-e}) \geq \pi_v^{-e}$, as required. □

Consider next a partition $E = A \cup B \cup C \cup D \cup E$ defined for P_u as follows. We set $A = E \setminus P_u$, $B = \{e \in P_u : u = v(P_u, e)\}$, $C = \{e \in P_u[u, v] : u \neq v(P_u, e)\}$, $D = \{e \in P_u[v, t] : v(P_u, e) \notin V(P[v, t])\}$ and $E = \{e \in P_u[v, t] : v(P_u, e) \in V(P[v, t])\}$. For $T \in \{A, B, C, D, E\}$ define $\bar{T} = \max_{e \in T} \ell({P_u}^{-e})$.

We obtain a similar partition $E = A' \cup B' \cup C' \cup D' \cup E'$, as well as values $\bar{A}', \bar{B}', \bar{C}', \bar{D}', \bar{E}'$, corresponding to P'_u, where every set is defined as above, but with P_u replaced by P'_u. Observe that our goal becomes proving

$$\mathrm{Val}^k(P_u) = \max\{\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}\} \geq \max\{\bar{A}', \bar{B}', \bar{C}', \bar{D}', \bar{E}'\} = \mathrm{Val}^k(P'_u).$$

Observe first that $\bar{B} = \bar{B}'$ and $\bar{C} = \bar{C}'$, thus the statement is trivial in the case $\mathrm{Var}^k(P'_u) = \max\{\bar{B}', \bar{C}'\}$. It remains to show that

$$\max\{\bar{A}', \bar{D}', \bar{E}'\} \leq \max\{\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}\}.$$

Set $\alpha = \ell(P_u[u, v])$ and note that $\bar{D}' \leq \alpha + \bar{Y}$. Indeed, every edge $e \in D'$ is seen by a node $w \in V(P'_u)$ that appears before v on P'_u, while $\bar{Y} \geq \pi_v^{-e}$, implying $\ell({P'_u}^{-e}) \leq \alpha + \pi_v^{-e} \leq \alpha + \bar{Y}$.

Since P_v is a subpath of P'_u and P'_v is a subpath of P_u we have $\bar{A} = \alpha + \bar{X}'$ and $\bar{A}' = \alpha + \bar{X}$. For the same reason we obtain $\bar{E} = \alpha + \bar{Z}'$ and $\bar{E}' = \alpha + \bar{Z}$.

Putting things together we are left with proving the inequality

$$\max\{\alpha + \bar{X}, \alpha + \bar{Y}, \alpha + \bar{Z}\} \leq \max\{\alpha + \bar{X}', \bar{B}, \bar{C}, \bar{D}, \alpha + \bar{Z}'\},$$

but this now trivially follows from Claim 1. □

Lemma 3.1.5 and the property that we state hereafter allow us to prove the correctness of a label-setting algorithm. The property follows from the fact that for any $u \in V$ and $e \in E$ one has $\pi^k_{rob}(u) \geq \pi_u^{-e}$.

Property 3.1.6. *Let $u \in V$ and $P \in \mathcal{P}_{u,t}$ be such that $\mathrm{Val}^k(P) = \pi_u^{-e}$ for some edge e seen on P by u. Then P is an optimal path from u.*

Analogously to our algorithm for ORP, we proceed by incrementally computing the optimal path for every node in the graph starting from t. We maintain a set U of nodes for which a robust path was already computed. For $u \in U$ we denote this path by P_u^*. The update rule for U works as follows. First, we check if for some edge $(v,u) \in E$ such that $v \in V \setminus U$ and $u \in U$ it holds that the path $Q = (v,u) \oplus P_u^*$ satisfies the condition in Property 3.1.6. In other words, we check if $\mathrm{Val}^k(Q) = \pi_v^{-e}$ for some edge $e \in Q$ seen by v. If such an edge exists we set $P_v^* := Q$ and $U := U \cup \{v\}$, an update that is valid due to Property 3.1.6. Assume next that no such edge exists. We call the set U in this situation *clean*. The following lemma states an update rule for clean sets U.

Lemma 3.1.7. *Let U be clean and let $(v,u) \in \arg\min_{(q,r)\in E\,:\, r\in U, q\notin U} \mathrm{Val}^k((q,r) \oplus P_r^*)$. Then $\pi_{rob}^k(v) = \mathrm{Val}^k((v,u) \oplus P_u^*)$.*

Proof. Assume towards contradiction that $Q = (v,u) \oplus P_u^*$ is not optimal for v. Let $P \in \mathcal{P}_{v,t}$ be an optimal path from v, namely $\pi_{rob}^k(v) = \mathrm{Val}^k(P)$. Let $(x,y) \in P$ be the last edge on P, when P is traversed from v to t with the property that $x \in V \setminus U$ and $y \in U$. Lemma 3.1.5 allows us to assume that $P[y,t] = P_y^*$, thus $P[x,t] = (x,y) \oplus P_y^*$. By choice of (v,u) and by the assumption that Q is not optimal we have

$$\mathrm{Val}^k(P[x,t]) \geq \mathrm{Val}^k(Q) > \mathrm{Val}^k(P).$$

Consider the edge $f \in P[x,t]$ with the property that

$$\mathrm{Val}^k(P[x,t]) = \ell(P[x,z]) + \pi_z^{-f},$$

where $z = v(P[x,t], f)$ is the first node on $P[x,t]$ that sees f. Clearly, if $z \neq x$ we have

$$\mathrm{Val}^k(P) \geq \ell(P[v,z]) + \pi_z^{-f} \geq \ell(P[x,z]) + \pi_z^{-f} = \mathrm{Val}^k(P[x,t]),$$

contradicting $\mathrm{Val}^k(P) < \mathrm{Val}^k(P[x,t])$. It follows that $z = x$ and $\mathrm{Val}^k(P[x,t]) = \pi_x^{-f}$, implying that U is not a clean set; a contradiction. □

Lemmas 3.1.5 and 3.1.7 immediately imply a polynomial algorithm for k-Hop ORP, whose statement is given as Algorithm 6.

Algorithm 6

1: Compute π_u^{-e} for each $e \in E$ and $u \in V$.
2: Let U be a set of nodes u for which $\pi^k_{rob}(u)$ and the corresponding optimal path P^*_u are known.
3: $U = \{t\}$; $\pi^k_{rob}(t) = 0$.
4: **while** $U \neq V$ **do**
5: **if** $\exists (v,u) \in E$ with $v \in V \setminus U$ and $u \in U$, such that the path $Q = (v,u) \oplus P^*_u$ satisfies Property 3.1.6 **then**
6: $U = U + v$; $P^*_v := Q$; $\mathrm{Val}^k(P^*_v) = \pi^k_{rob}(v)$.
7: **else**
8: Find $(v,u) \in \arg\min_{(q,r)\in E\,:\, r\in U, q\notin U} \mathrm{Val}^k((q,r) \oplus P^*_r)$.
9: $U = U + v$; $P^*_v := (v,u) \oplus P^*_u$; $\mathrm{Val}^k(P^*_v) = \pi^k_{rob}(v)$.
10: **end if**
11: **end while**

Observe that, as we have underlined for ORP, everything presented and proved in this section is valid also if we are given an undirected graph, therefore Algorithm 6 solves also instances with an undirected edge-weighted graph as input.

Consider now the running time of Algorithm 6. First of all we note that steps 4-11 can be roughly implemented in time $O(mn)$ both for directed and undirected graph; we basically run the *while* cycle n times and during each iteration we execute $O(1)$ operations for the edges in the cut induced by U, that gives a complexity of $O(m)$ for a single iteration. It would be interesting to investigate whether the implementation used for steps 2-10 of Algorithm 1 (see subsection 2.1.3) can be used also in this case, in order to obtain an $O(m + n \log n)$ running time also for steps 4-11 of Algorithm 6. On the other hand, even if it turn out to be possible, it is not sure that this would give an effective improvement on the overall complexity bound; the last statement is based on the consideration we are going to make in the rest of the analysis.

As for Algorithm 1, step 1 is the bottleneck of Algorithm 6: it is actually more challenging in the latter case. For ORP, indeed, we need to compute less π *values*; more precisely we compute m of them, since we need to know $\pi_u^{-(u,v)}$ for each $(u,v) \in E$. On the other hand, for k-Hop ORP we need to compute for each $u \in V \setminus t$:

$$\pi_u^{-(v,z)} \quad \forall (v,z) \in E : v \in N^i(u), z \in N^{i+1}(u) \quad i = 0, 1, \ldots, k$$

where $N^i(u)$ is the $i-th$ neighborhood of u, and in particular $N^0(u) \equiv u$. Even worst, since k can go up to $n-1$, we could actually need to compute π_u^{-e} for each $e \in E$ and $u \in V$, that is what we have in step 1 of Algorithm 6. As we discuss in subsection 2.1.3 computing the shortest path tree T in $O(m+n\log n)$ time from every node to t would trivially tells us the value of π_u^{-e} for each edge e of the graph not in T (and for any $u \in V$); we note that this is true both for directed and undirected graphs.

We can hence concentrate our efforts on computing π-values for edges in the tree, more precisely π_u^{-e} for each $u \in V$ and $e \in E(T)$. For directed graphs we can solve $O(n^2)$ instances of a shortest path problem, for a running time of $O(n^2\mathrm{SP}(n,m))$, where $\mathrm{SP}(n,m)$ is the complexity of a single shortest path computation on a graph with n nodes, m edges and nonnegative weights.
For undirected graphs, instead, we can do what follows: for any $u \in V$, let P_u be the unique u-t shortest path on T; we need to solve a Replacement Path problem given u, t and P_u as input. From section 1.1 we know that this can be done in $O(m\alpha(m,n))$ thanks to the algorithm provided by Nardelli, Proietti and Widmayer [21]. Since we have to do it for any node $u \in V$ we obtain a running time of $O(nm\alpha(m,n))$.

Summarizing we can execute step 1 of Algorithm 6 in $O(n^2\mathrm{SP}(n,m)) + O(m + n\log n) = O(n^2\mathrm{SP}(n,m))$ for directed graphs and in $O(nm\alpha(m,n)) + O(m + n\log n) = O(nm\alpha(m,n))$ for undirected graphs.

Finally, we can state the following:

Theorem 3.1.8. *Given an instance of k-Hop ORP the values π_{rob}^k and the corresponding paths can be computed in time $O(nm\alpha(m,n))$ in undirected graphs, and $O(n^2\mathrm{SP}(n,m))$ in directed graphs.*

We leave as an interesting open question whether the bounds provided in Theorem 3.1.8 can be improved.

3.1.1 Radius ORP

In this subsection we briefly introduce an extension of ORP similar to k-Hop ORP. In k-Hop ORP we assume that the RM is informed about the failed edge when it is k hops away on the nominal path. An alternative definition takes the lengths of edges on this path into account. In this problem, which we call *Radius ORP*, the integer $k \leq n-1$ is replaced by a value $R \leq \ell(E)$ called the *radius*. In this problem the RM is informed about the failed edge e on the nominal path P at the first node that is at distance at most R from its closer endpoint, formally:

Definition 3.1.9. *Given a path $P \in \mathcal{P}_{s,t}$, an edge $e = (u, u') \in E(P)$ and a radius R, we define $v(P, e) \in V(P)$ as the first node of P, starting from s, such that $|E(P[v(P, e), u])| \leq R$.*

The definitions 3.1.2, 3.1.3 and 3.1.4 are adapted accordingly. We claim without proof that our algorithm 6 for k-Hop ORP solves Radius ORP as well. Basically, this follows from fact that Property 3.1.6 and Lemma 3.1.5 remain correct. The latter, indeed, only relies on the following property: the set of edges on P that a node sees is an interval on this path, and furthermore, for every two consecutive nodes $u_1, u_2 \in V(P)$, with u_2 being the closer one to t, the set of edges seen by u_1 in $P[u_2, t]$ is a subset of the set of edges seen by u_2.

Therefore we can state the following result:

Theorem 3.1.10. *Given an instance of Radius ORP the corresponding optimal paths can be computed in time $O(nm\alpha(m, n))$ in undirected graphs, and $O(n^2\mathrm{SP}(n, m))$ in directed graphs.*

3.2 Strong k-Hop ORP

We end this chapter with another variant of k-Hop ORP. In this variant, whose input is identical to that of k-Hop ORP, the information about the failed edge travels through the edges of the entire graph, as opposed to only the edges of the nominal path. Formally, the first node along the chosen nominal path that is informed about the failure of some edge $e \in E$ is the one closest to s that is at most k hops away from e in G. This problem, which we denote by *Strong k-Hop ORP* turns out to be NP-hard to approximate even when $k = 1$. Note that, for $k = 0$, Strong k-Hop ORP reduces to ORP, as for every path the robust value is the same in the two different problems.

Theorem 3.2.1. *for any $\epsilon > 0$ it is NP-hard to approximate Strong 1-Hop ORP within a factor of $3 - \epsilon$ in undirected graphs. In directed graphs it is strongly NP-hard to decide if there exists a nominal path with finite robust length.*

Proof. We describe the proof for undirected graphs first. We show a reduction from a variant of the Hamiltonian Path problem, in which the input additionally specifies a two nodes $u, v \in V$, and the goal is to decide if there exists a Hamiltonian path with endpoints u, v.

Let $G(V, E), u, v$ be an instance of this problem and let $n = |V|$. We construct a graph $H = (V', E')$ comprising an instance to Strong 1-Hop ORP. The graph H is composed of a copy of G alongside two long paths P_1, P_2 and an additional node s. The edges in the copy of G have length zero. The Paths P_1, P_2 have common endpoints s', t ($P_1 \cup P_2$ is a simple cycle). Both P_1 and P_2 have $2n + 2$ nodes including s' and t, and $2n + 1$ edges. Let P_1 and P_2 be the paths passing through the nodes $s', u_1, x_1, u_2, x_2, \cdots, u_n, x_n, t$ and $s', v_1, y_1, v_2, y_2, \cdots, v_n, y_n, t$, respectively. We double the edges $x_i u_{i+1}$ and $y_i v_{i+1}$ for every $i = 1, \cdots, n-1$, effectively making them failure-safe. The lengths of edges on these paths are all set to zero, except for the edges $s'u_1$ and $s'v_1$, whose length is set to one. We also double all edges in the copy of G, as well as the edges su and vs'. Thus, we can assume without loss of

generality that the only faulty edges are u_ix_i and v_iy_i for every $i = 1, \cdots, n$.

Additionally, the nodes s and s' are connected with zero-length edges to u and v, respectively. Let $w_1, \cdots, w_n$ denote the nodes of the copy of G in H. We additionally connect each node w_i to both u_i and v_i with edges of length ∞ for every $i = 1, \cdots, n$. This concludes the construction of the graph H. See Figure 3.2 for an illustration. The critical observation we use is that the only node in the copy of G that sees the edges u_ix_i and v_iy_i is the node w_i.

We claim that deciding whether there exists a solution for the Strong 1-Hop ORP instance with value $\theta < 3$ is equivalent to deciding the existence a Hamiltonian Path in G with end-nodes u and v. Observe that for the obtained Strong 1-Hop ORP instance, if any s-t walk has length smaller than 3, then it has length of exactly 1.

Consider first the case that the required Hamiltonian Path P exists. Set the nominal path in the solution to the Strong 1-Hop ORP instance to

$$su \oplus P \oplus vs' \oplus P_1.$$

We claim that this solution to the Strong 1-Hop ORP instance has value 1. Indeed, the length of the path itself is 1. Consider next any failed edge e on this path. If this edge does not lie on the path P_1, the incurred total length will be at most 1. If, on the other hand, $e = u_ix_i$ is an edge on P_1, this failure will be discovered at the node w_i, thus the optimal detour will follow the rest of the Hamiltonian path to v, then jump to s' and then follow P_2 until t, attaining the length 1. This concludes the first direction of the proof.

Consider next the case that a desired Hamiltonian path does not exist. Consider any nominal path Q for the Strong 1-Hop ORP instance. Let $i \in [n]$ be such that w_i is not on the path Q. Such a node exists due to the assumption that the Hamiltonian path does not exist. We consider two failure scenarios, namely the ones corresponding to the failure u_ix_i and v_iy_i. Observe that since w_i is not contained in Q, the RM will follow Q until an intermediate node x of either P_1 or P_2 in both scenarios. Assume without loss of generality that x is contained in P_1. It follows that if u_ix_i fails the walk performed by the RM will have to backtrack the edge $s'u_1$ and take the path P_2

instead. The resulting walk has length of at least 3. This proves the second direction of the proof.

We proved that for a given instance of Strong 1-Hop ORP, it is NP-hard to distinguish if the optimal solution to the instance has value of at most 1, or it has value of at least $3 - \epsilon$. We conclude that approximating Strong 1-Hop ORP within any factor better than 3 is NP-hard.

The required adaptation for obtaining the result for directed graphs is replacing every undirected edge xy in the reduction above with an edge directed from x to y. In the new graph the only u_i-t and v_i-t paths are the ones through the nodes $u_{i+1}, \cdots, u_n$ and $v_{i+1}, \cdots, v_n$, respectively. The remainder of the proof is a straightforward adaptation of the arguments above.

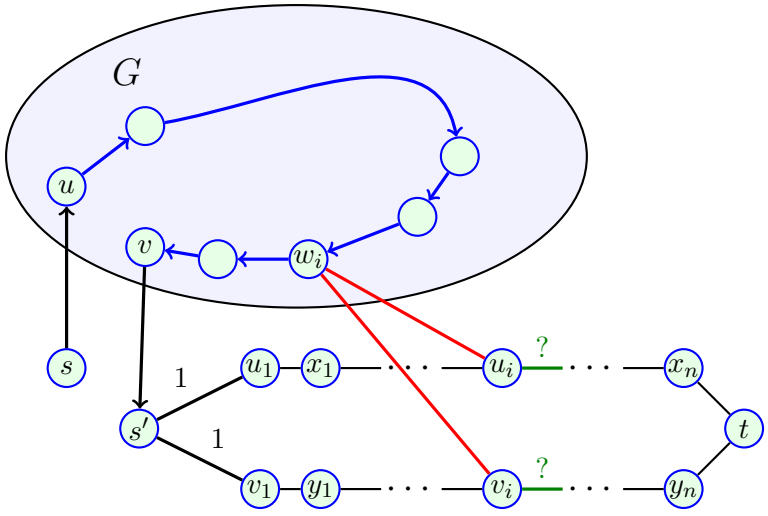

Figure 3.2: The Strong 1-Hop ORP instance in Theorem 3.2.1

□

We note that in 2 s-t connected undirected graphs, every shortest path is a 3-approximation of the optimal solution to Strong 1-Hop ORP, thus the approximability of this problem is settled. The proof of this simple fact is similar to the proof of Lemma 4.0.4, and thus omitted.

Chapter 4

ORP Game

In this Chapter we study ORP problem from a game theoretic perspective. That is based on the randomized network interdiction problem that has been recently introduced by Bertsimas, Orlin and Nasrabadi [7]. Network Interdiction (NI) problems involve two opposing forces, a player and an interdictor, who are engaged in a warlike conflict. The player operates a network in order to optimize some objective function such as moving a supply convoy through the network as quickly as possible or maximizing the amount of materiel transported through the network. The interdictor attempts to limit the player's achievable objective value by interdicting arcs, for example, by attacking arcs to destroy them, to slow travel over them, or to reduce their capacity. NI problems have been intensively studied and applied in many applications area such as military planning, controlling infections in a hospital, controlling floods and drug interdiction. An interesting aspect of these problems is that they can be viewed as a game between the two opponents, the player and the interdictor. In [7] the authors propose a generalization of NI in which both opponents move simultaneously and may use randomized strategy.

We are going to investigate the ORP problem in a similar perspective, exploring a two players' game that is the natural middle ground between the problems MVA and ORP. A first player, the *path builder*, is interested in arriving from s to t as quickly as possible. The second player, the *interdictor*, tries to make the latter distance as

long as possible by removing a single edge from the graph. The *strategies* for the two players are the s-t paths, and the edges $e \in E$, respectively.

In one setup, the interdictor communicates her strategy *first*, i.e. which edge is removed from G. The path builder chooses his strategy *after*: clearly he chooses a shortest path s-t in the graph $G - e$. Therefore, the problem that the interdictor faces in this setting is clearly the MVA problem, as she will remove the edge $e \in E$ maximizing π_s^{-e}, the length of the shortest s-t path in the graph $G - e$. In the following, we let $z^*(MVA)$ be the value of an optimal solution to MVA.

In the other extreme, the path builder communicates his strategy first, i.e. an s-t path P. Then the interdictor moves, and clearly removes the edge e maximizing $\ell(P^{-e})$. Note that we assume that, if $e \in P$, the interdictor will delay the failure of the edge to the point at which the path builder attempts to cross it. Hence, the problem that the path builder faces is exactly ORP, i.e. that of choosing an s-t path with the least robust value. In the following, we let $z^*(ORP)$ be the value of an optimal solution to ORP.

The next lemma, whose simple proof we skip, shows that $z^*(ORP) \geq z^*(MVA)$.

Lemma 4.0.2. *Let P and e be an s-t path and an edge of E, respectively. Then* $\mathrm{Val}(P) \geq z^*(ORP) \geq z^*(MVA) \geq \pi_s^{-e}$.

In our two players' game, that we call the *ORP Game*, both players communicate their strategies *at the same time*. In particular, for a given s-t path P and edge $e \in E$, the payoff for the interdictor is $\ell(P^{-e})$. Lemma 4.0.2 shows that in general $z^*(ORP) \geq z^*(MVA)$. The next theorem characterizes the instances of the ORP Game admitting a pure NE as those for which $z^*(ORP) = z^*(MVA)$.

Theorem 4.0.3. *Let P and e be optimal solutions to the ORP and MVA instances on $G = (V, E)$. Then (P, e) is a pure NE of the ORP Game if and only if* $\mathrm{Val}(P) = \pi_s^{-e}$. *Moreover, in this case,* $\mathrm{Val}(P) = z^*(ORP) = z^*(MVA) = \pi_s^{-e}$.

Proof. Assume first that (P, e) is a pure NE. Since (P, e) is a pure NE, it follows the path builder does not benefit from changing to any other path in $G - e$, while the

interdictor keeps e fixed. In particular, let P' be a shortest s-t path in $G - e$, and note that $\ell(P') = \pi_s^{-e}$. It follows that $\ell(P^{-e}) \leq \pi_s^{-e}$, i.e. also P^{-e} is a shortest path in $G - e$, and $\ell(P^{-e}) = \pi_s^{-e}$. At the same time, the interdictor does not benefit from removing an edge different from e, while the path builder keeps P fixed. Thus, it follows that e is a critical edge for P, namely $\ell(P^{-e}) = \text{Val}(P)$. Therefore we have that $\text{Val}(P) = \ell(P^{-e}) = \pi_s^{-e}$.

Suppose now that P is an s-t path and e an edge of e such that $\text{Val}(P) = \pi_s^{-e}$. We know from Lemma 4.0.2 that $\text{Val}(P) \geq \ell(P^{-e}) \geq \pi_s^{-e}$, therefore $\text{Val}(P) = \ell(P^{-e}) = \pi_s^{-e}$. It follows that e is a critical edge of P, and the interdictor does not benefit from removing an edge different from e. Analogously, the path builder does not benefit from choosing another path, as P^{-e} is a shorter path of $G - e$.

Finally, observe that, whenever $\text{Val}(P) = \pi_s^{-e}$ holds for some s-t path P and edge e of E, then $\text{Val}(P) = z^*(ORP) = z^*(MVA) = \pi_s^{-e}$, from Lemma 4.0.2. □

Theorem 4.0.3 has also the following algorithmic implication. Recall that we can compute $z^*(MVA)$ in time $O(m+n \log n)$ [17], the same running time we obtained for undirected ORP (Theorem 2.1.14). This clearly implies that in time $O(m + n \log n)$ we can compute a pure NE of the ORP Game in undirected graphs, if one exists, or certify that no pure NE exists. Indeed the aforementioned algorithms allow us to check the condition $z^*(ORP) = z^*(MVA)$ and compute corresponding optimal solutions, P^* and e^*, with the latter time complexity. Theorem 4.0.3 asserts that if the latter condition is satisfied, then (P^*, e^*) is a pure NE, otherwise no pure NE exists.

Theorems 2.1.13 and 4.0.3 also imply a $O(n\text{SP}(n, m))$ algorithm for the same problem in directed graphs, since we know that this is the running time of the algorithms used to find $z^*(ORP)$ and $z^*(MVA)$ for directed graphs.

We conclude by analyzing the ratio $\frac{z^*(ORP)}{z^*(MVA)}$. The next lemma shows that, for undirected graphs, it is at most 3.

Lemma 4.0.4. *Let G be undirected with s-t edge-connectivity of at least two. Then $z^*(ORP) \leq 3z^*(MVA)$.*

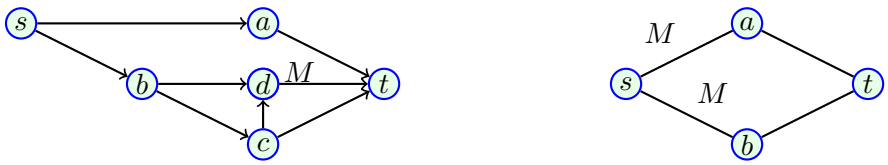

Figure 4.1: On the ratio $\frac{z^*(ORP)}{z^*(MVA)}$. When G is directed, it is unbounded: $z^*(ORP) = M+3$ (attained by the path s, b, c, t) and $z^*(MVA) = 3$ (attained by the edge (s, a)). When G is undirected, it is at most 3 and the bound is tight: $z^*(ORP) = 3M + 1$ (attained by the path s, a, t) and $z^*(MVA) = M + 1$ (attained by the edge sa). Unless otherwise specified, we assume that the length of an edge is 1.

Proof. Let f be a most vital edge for G, and let P be a shortest s-t path of $G - f$ (the existence of P is granted by the hypothesis that G is two s-t edge-connected). By definition, $z^*(MVA) = \ell(P)$. We now show that the robust value of P is at most $3\ell(P)$, which is of course sufficient for our purposes.

Recall that $\mathrm{Val}(P) = \max_{e \in E} \ell(P^{-e})$. Trivially, for each edge $e \notin P$, $\ell(P^{-e}) = \ell(P)$. Consider now an edge $e \in P$: we claim that $\ell(P^{-e}) \leq 3\ell(P)$. This is because $\ell(P^{-e})$ is upper bounded by the length of the following walk Q in $G - e$: take the subpath of P from s until the first node of e back and forth, then (when back at s) take a shortest path to t in $G - e$. Trivially, $\ell(Q) \leq 2\ell(P) + \ell(P)$, where we use the fact that $\ell(P) = z^*(MVA) \geq \pi_s^{-e}$. □

We give in Figure 4.1 (right) an example showing that the bound of Lemma 4.0.4 is tight. In the same figure we show that, to the contrary, for directed graphs, the ratio $\frac{z^*(ORP)}{z^*(MVA)}$ is unbounded, even when there are paths with finite robust value (see Theorem 2.1.5).

4.1 ORP Game in mixed strategy: Stochastic ORP

Theorem 4.0.3 shows that there are no pure NE when $z^*(ORP) \neq z^*(MVA)$. However, from a classical result in Game Theory (see e.g. Chapter 15 in [8]), there will NE when the game is played in *mixed strategies*, as for both players the sets of pure

strategies is finite (simple s-t paths and edges). Whether it is possible to find this mixed NE in polynomial time is an interesting question.

It is well known that, in order to find these equilibria, it is sufficient to determine for each player his/her conservative strategy, and this can be done via linear programming. However, there is a catch: the linear program (the straightforward details can be found in [8]) that the interdictor needs to solve is not polynomially bounded in the size of the graph, since the number of simple $s-t$ paths is not. Interestingly, the related separation problem, turns out to be a stochastic version of ORP, that we call *Stochastic ORP*. Unfortunately, as we show in the following, this separation problem is hard.

First let us define the problem. Recall that in the (deterministic) ORP (see Section 2.1), we implicitly assume that at most one edge fails, and we look for paths with minimum robust length, where the robust length of a path is taken in the *worst* scenario.

In the Stochastic ORP, we again assume that at most one edge fails, but we are given probabilities. Namely, for each $e \in E$, we are given the probability $p(e) \in [0,1]$ that e will be the *unique* edge to fail, and we are also given the probability $p(\emptyset) \in [0,1]$ that no edge fails. The following must hold:

$$p(\emptyset) + \sum_{e \in E} p(e) = 1.$$

In the stochastic ORP, we again look for a path with minimum robust length, but this is now taken *in expectation.*

Definition 4.1.1. *Given a node $v \in V$, the expected robust length of the v-t path P is*

$$\mathbb{E}\mathrm{Val}(P) = (p(\emptyset) + \sum_{e \notin E(P)} p(e))\ell(P) + \sum_{e \in E(P)} \left[p(e)\ell(P^{-e})\right].$$

Our goal is to find, for some $v \in V$, a path P minimizing $\mathbb{E}\mathrm{Val}(P)$ over all paths $P \in \mathcal{P}_{v,t}$. We can now formally define Stochastic ORP:

Definition 4.1.2 (The Stochastic Online Replacement Path problem)**.** ***Given:*** *an edge-weighted undirected (directed) graph $G = (V, E, \ell)$, where $\ell : E \to \mathbb{R}^+$, a source $s \in V$ and a destination $t \in V$. We are also given a value $p(\emptyset) \in [0, 1]$ and, for each $e \in E$, a value $p(e) \in [0, 1]$, such that $p(\emptyset) + \sum_{e \in E} p(e) = 1$.*
Find: *an optimal s-t path, namely a path P minimizing $\mathbb{E}\mathrm{Val}(P)$ over all paths $P \in \mathcal{P}_{s,t}$.*

Remember that solving the above problem in polynomial time would imply that we can find in polynomial time the Nash equilibria for the ORP game in mixed strategy, but we can prove what follows:

Theorem 4.1.3. *Stochastic ORP in directed graphs admits no approximation algorithms with sub-exponential approximation guarantee, unless $P = NP$.*

Proof. We prove that Stochastic ORP is NP hard by a reduction from the Hamiltonian Path problem. Recall that the Hamiltonian Path problem is defined as follows. Given a directed graph $G = (V, E)$ a source $s \in V$ and a target $t \in V$, find a simple s-t path in G visiting all vertices.

It will convenient to assume that the vertices can also fail. In directed graphs this can be modeled with the following easy transformation. Every vertex v with incoming edges $e_1, \ldots, e_k$ and outgoing edges $f_1, \ldots, f_r$ is replaced with two vertices v and v^+ connected by an edge (v, v^+). The edges $e_1, \ldots, e_k$ are connected to v and the outgoing edges $f_1, \ldots, f_r$ are connected to v^+. The failure of the edge (v, v^+) now naturally models the failure of the vertex v. The Hamiltonian path problem now translates into finding a path traversing all of edges of the form (v, v^+). We abuse notation and assume this variant of the problem, and also call the new graph $G = (V, E)$. Let $n = |V|$. We call the edges that correspond to vertices *critical*.

We transform the instance of Hamiltonian Path to an instance of Stochastic ORP as follows. The graph contains a copy of G with zero-length edges. Every critical edge $e = (u, v)$ is subdivided into a path of length two with edges $(u, w(e))$ and $(w(e), v)$. Every vertex of the form $w(e)$ for $e \in E$ is connected with a directed path

of length two to t. The second edge on this path has a positive (and very small) failure probability ϵ, making any feasible path avoid these extra edges (since should this edge fail, the detour cost from the middle vertex of these paths is ∞). These paths have cost zero, thus allowing a zero-cost detour from any vertex of the form $w(e)$.

For a critical edge $e = (u, v)$, the failure probability of every edge of the form $(w(e), v)$ is set to $\frac{1-\epsilon n}{n}$, while the failure probabilities of all other edges (except the ones with failure probability ϵ) is set to zero. All edges mentioned so far have length zero.

Finally, we add two more vertices s', t' to be the source and the target of the Stochastic ORP instance. s' is connected to s with a zero-cost edge with failure probability zero, while t is connected with a unit-cost edge to t'. This edge also has failure probability zero. We call the obtained graph $H = (W, F)$.

We make the following observations. First, with probability $1n\epsilon$ one of the edges $(w(e), v)$ fails. Assume first that a Hamiltonian s-t path P exists in G. In this case one can take $Q = (s', s) \oplus P \oplus (t, t')$ as a nominal path. Clearly, the value of Q is ϵn, as any failure of an edge of the form $(w(e), v)$ will be spotted while in the copy of G, thus the cheap detour from the corresponding critical edge will be taken to reach t' at zero cost. Only with probability ϵn the failure does not occur in the copy of G, but rather in one of the detours from some critical edge to t'. In this case a cost of one is incurred.

Similarly, if a Hamiltonian path does not exist, the value of any nominal path is at least $\epsilon n + \frac{1-\epsilon n}{n}$, as at least one critical edge cannot be seen on the tour. Observe that the ratio between $\epsilon n + \frac{1-\epsilon n}{n}$ and ϵn can be made arbitrarily large by driving ϵ to zero. This ends the proof. □

We make a couple of remarks. First, note that a similar result does not hold for undirected graphs. It is easy to see that taking as nominal path any shortest s-t path provides, as for ORP, a 3-approximation. However, It is still possible to prove NP-hardness of this variant (we omit the details). Then, we observe that Theorem

4.1.3 does not imply that it is not possible to find a Nash equilibria for the ORP game in mixed strategy in polynomial time. Though the separation problems is hard, it might still be possible that finding the NE in mixed strategy is *not* PPAD-complete. We leave this as an open question.

Conclusions and future work

It is well known that there are two main approaches for optimization problems affected by data uncertainty: *stochastic optimization* and *robust optimization*. In the former the data is given as a probability distribution over some set of inputs and the optimization problem is to find a solution that minimizes the expected cost of the solution. In the latter one tries to find a solution that minimizes the cost of the worst case realization of the input data.

In this thesis we have proposed some robust optimization models for the design of a Routing Mechanism (RM) capable of dealing with faulty networks. More precisely our goal has been to develop routing schemes able to react to edge failures that occur *online* and, in particular, after the routing has started.

We have studied the following problem: given a directed (or undirected) graph $G = (V, E)$ with nonnegative costs on the edges, an origin $s \in V$ and a destination $t \in V$, find an s-t path P minimizing the worst-case arrival time, under the assumption that at most one edge e of the network can fail and that RM wants to route package as soon as possible but without knowing the identity of the failed edge, if any. Indeed, RM can discover which edge is failed only while it is traversing the path.

Depending on the way RM discovers the failed edge e, we can define different problems: the *Online Replacement Path problem* (ORP) is the one where RM realizes that e is failed only when it tries to cross it (let us note that for ORP we show how the presented results can be generalized to more than one failure); if the discovery of e happens when RM reaches a node that is at most k hops away from e *on* P or a node that is at distance at most R from the closer endpoint of e again *on* P we have

the *k-Hop ORP problem* and the *Radius ORP problem* respectively. Finally, if RM discovers the failed edge when it reaches a node of P that is at most k hops away from e *on* G, we have *Strong k-Hop ORP*. For the first three problems we provide polynomial time algorithms, while we state an hardness result for the last one.

Ultimately, we can say that our results show that ORP-models comprise a highly flexible and tractable framework for dealing with robustness issues in the design of RM-s.

Moreover, in this thesis, we have raised some questions that are left open. First of all, we ask whether it is possible to improve the running time bounds provided for the algorithms solving k-Hop ORP and Radius ORP in order to obtain the same complexity bounds we have for ORP. We are also interested in understanding if we can find more efficiently a Pareto front for Bi-objective ORP, that is a problem linking ORP to Shortest Path problem, whose solutions help to quantify the trade-off between the robust length and the nominal length of a path.

In Chapter 4 we have studied ORP from a game theory perspective, defining a Network Interdiction game, called *ORP Game*, for which we have been able to characterize and efficiently find those instances presenting a Nash Equilibrium (NE). We have also addressed the problem of finding a NE of the mixed strategy extension of ORP Game, but it is still no clear if such problem is polynomially solvable or is PPAD-complete.

Finally, recall that we also briefly addressed the ORP problem with arbitrary costs; in the case where we can have edges with negative length but no negative cycles, the properties described for ORP are still valid, but a different algorithm is needed: we conjecture that a Ford-Bellman like algorithm can solve the problem. On the other hand, for the case where negative cycles are allowed, we simply conjecture a condition for the existence of a finite solution to the problem. Proving the mentioned conjectures is an interesting open point, as well as finding algorithmic techniques for the second described case.

In this thesis we have observed several times that the presented models might

be quite suitable for transportation problems. Indeed, a future research direction we are interested in, consists in applying ORP-like techniques to Public Transport (PT) Systems that can be very unreliable in some cities. Despite the source of uncertainty affecting PT networks is more often due to the unreliability and to the random nature of the timetable rather than to failures problems, we aim at devising an ORP-like framework for PT. From real-case experiences, we would say that the main source of delays of PT users, consists in the fact that travellers using PT very often miss the planned connections. Our idea is to model the PT network in such a way that some faulty edges represent connections and to test, at this point, our algorithms on the PT System of Rome; in particular we want to compare the quality, in terms of robustness, of our solutions with other commercial solutions commonly used (e.g. Google Transit).

We conclude by suggesting the investigation of a problem, that we call *Flow-ORP* which generalizes the ORP setting to (single commodity) flows. The general idea is pretty simple: we are given a flow, *together with its path decomposition*, and at some point an edge e fails: in the same spirit of ORP we discover the failure of e when we test it: at that point we need to on-line reroute those paths that were using e.

However, our rerouting must be such that the flow on unaffected paths (i.e. paths not containing e), as well as the flow from s to e, for paths containing e, cannot be rerouted: this models situation of continuous flow of physical objects or material. A more formal definition follows.

Let $G = (V, E)$ be a either directed or undirected graph, $c : E \mapsto \mathbb{Z}_+$ a capacity vector on the set of edges and let $s, t \in V$. Let $d \in \mathbb{Z}_+$ be some amount of flow we want to send from s to t, i.e. d is the *demand*. For a s-t path P and an edge $e \in P$ let $s(P, e) \in V$ denote the endpoint of e closer to s on P, and $t(P, e) \in V$ denote the endpoint of e closer to t on P. The simple subpath of P from e to t (excluding e) is denoted by $rest(P, e)$.

Flow-ORP asks to find what we call *an $s - t$ on-line reroutable flow* $f : E \to \mathbb{R}$ and a path decomposition $P_1, \cdots, P_k$ with corresponding flow values $d_1, \cdots, d_k$ (with

$d = \sum_{i\in[k]} d_i$) satisfying the *replacement property*. For an edge $e \in E$ let $I_e \subset [k]$ denote the indices of the paths P_i with $e \in P_i$.

Consider an edge $e = uv$. For $x = u, v$, let $d^x = \sum_{i\in I_e : x = s(P_i,e)} d_i$. d^u and d^v are the residual demands that need to be rerouted respectively from u and v in case e fails. Define the *residual capacity* vector c' corresponding to failure e by setting $c'_e = 0$ and for $a \in E - e$

$$c'_a = c_a - f_a + \sum_{i\in I_e \,:\, a\in rest(P_i,e)} d_i.$$

The *residual flow problem* corresponding to failure $e \in E$ is the (multi source, single sink) flow problem on G with capacity c' and demands d^u from u to t and d^v from u to t.

Finally, the *replacement property* is satisfied if the residual flow problem corresponding to e is feasible for every $e \in E$.

A bunch of natural and very interesting questions arise from this general framework, such as: find the maximum value of an $s - t$ on-line reroutable flow of maximum value, no matter which edge fails; given some demand d, find an $s - t$ on-line reroutable flow of value d with minimum cost, no matter which edge fails etc. These questions are left open.

Bibliography

[1] D. Adjiashvili and R. Zenklusen. An s - t connection problem with adaptability. *Discrete Applied Mathematics*, 159(8):695 – 705, 2011.

[2] David Adjiashvili, Gianpaolo Oriolo, and Marco Senatore. The online replacement path problem. In *ESA*, pages 1–12, 2013.

[3] H. Aissi and D. Bazgan, C. Vanderpooten. Approximation complexity of min-max (regret) versions of shortest path, spanning tree, and knapsack. In *ESA '05*, pages 862–873, 2005.

[4] A. Bar-Noy, S. Khuller, and B. Schieber. The complexity of finding most vital arcs and nodes. Technical report, Univ. of Maryland Institute for Advanced Computer Studies Report No. UMIACS-TR-95-96, College Park, MD, USA, 1995.

[5] Amotz Bar-Noy and Baruch Schieber. The canadian traveller problem. In *SODA*, pages 261–270, 1991.

[6] A. Bernstein. A nearly optimal algorithm for approximating replacement paths and k shortest simple paths in general graphs. SODA '10, pages 742–755, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[7] Dimitris Bertsimas, Ebrahim Nasrabadi, and James B. Orlin. On the power of randomization in network interdiction. *CoRR*, abs/1312.3478, 2013.

[8] V. Chvatal. *Linear Programming.* Series of books in the mathematical sciences. W. H. Freeman, 1983.

[9] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization.* John Wiley & Sons, Inc., New York, NY, USA, 1998.

[10] K. Dhamdhere, V. Goyal, R. Ravi, and M. Singh. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *FOCS '05*, pages 367–378, Washington, DC, USA, 2005. IEEE Computer Society.

[11] Y. Emek, D. Peleg, and L. Roditty. A near-linear-time algorithm for computing replacement paths in planar directed graphs. *ACM Trans. Algorithms*, 6:64:1–64:13, September 2010.

[12] Y.Y. Fan, R.E. Kalaba, and J. E. Moore. Arriving on time. *Journal of Optimization Theory and Applications*, 127(3):497, 2005.

[13] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34:596–615, July 1987.

[14] Z. Gotthilf and M. Lewenstein. Improved algorithms for the k simple shortest paths and the replacement paths problems. *Inf. Process. Lett.*, 109:352–355, March 2009.

[15] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.

[16] P. N. Klein, S. Mozes, and O. Weimann. Shortest paths in directed planar graphs with negative lengths: A linear-space $o(n \log^2 n)$-time algorithm. *ACM Trans. Algorithms*, 6:30:1–30:18, April 2010.

[17] K. Malik, A. K. Mittal, and S. K. Gupta. The k most vital arcs in the shortest path problem. *Operations Research Letters*, 8(4):223–227, 1989.

[18] A. Moreno, A. Valls, and A. Ribes. Finding efficient organ transport routes using multi-agent systems. In *In: Proceedings of the IEEE 3rd International Workshop on Enterprise Networking and Computing in Health Care Industry (Healthcom)*, pages 233–258, 2001.

[19] E. Nardelli, G. Proietti, and P. Widmayer. Finding the detour-critical edge of a shortest path between two nodes. *Information Processing Letters*, 67(1):51–54, 1998.

[20] E. Nardelli, G. Proietti, and P. Widmayer. Swapping a failing edge of a single source shortest paths tree is good and fast. *Algorithmica*, 35:2003, 1999.

[21] E. Nardelli, G. Proietti, and P. Widmayer. A faster computation of the most vital edge of a shortest path. *Information Processing Letters*, 79(2):81–85, 2001.

[22] E. Nardelli, G. Proietti, and P. Widmayer. Finding the most vital node of a shortest path. In Jie Wang, editor, *Computing and Combinatorics*, volume 2108 of *Lecture Notes in Computer Science*, pages 278–287. Springer Berlin / Heidelberg, 2001.

[23] Evdokia Nikolova and David R. Karger. Route planning under uncertainty: The canadian traveller problem. In *AAAI*, pages 969–974, 2008.

[24] Evdokia Nikolova, Jonathan A. Kelner, Matthew Brand, and Michael Mitzenmacher. Stochastic shortest paths via quasi-convex maximization. In *ESA*, pages 552–563, 2006.

[25] N. Nisan and A. Ronen. Algorithmic mechanism design (extended abstract). STOC '99, pages 129–140, New York, NY, USA, 1999. ACM.

[26] Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theor. Comput. Sci.*, 84(1):127–150, July 1991.

[27] L. Roditty. On the k-simple shortest paths problem in weighted directed graphs. SODA '07, pages 920–928, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[28] L. Roditty and U. Zwick. Replacement paths and k simple shortest paths in unweighted directed graphs. ICALP'05, pages 249–260. Springer-Verlag, 2005.

[29] Samitha Samaranayake, Sebastien Blandin, and Alexandre M. Bayen. Speedup techniques for the stochastic on-time arrival problem. In *ATMOS*, pages 83–96, 2012.

[30] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975.

[31] O. Weimann and R. Yuster. Replacement paths via fast matrix multiplication. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:655–662, 2010.

[32] C. Wulff-Nilsen. Solving the replacement paths problem for planar directed graphs in o(n log n) time. SODA '10, pages 756–765, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[33] Peng Xiao, Yinfeng Xu, and Bing Su. Finding an anti-risk path between two nodes in undirected graphs. *J. Comb. Optim.*, 17(3):235–246, 2009.

[34] G. Yu and J. Yang. On the robust shortest path problem. *Computers & Operations Research*, 25(6):457 – 468, 1998.