Algorithms for Computing QoS Paths with Restoration

Bell-Labs Technical Memorandum

Yigal Bejerano, Yuri Breitbart, Ariel Orda, Rajeev Rastogi, Alexander Sprintson

Abstract

There is a growing interest among service providers to offer new services with *Quality of Service* (QoS) guaranties that are also resilient to failures. Supporting QoS connections requires the existence of a routing mechanism, that computes the *QoS paths*, *i.e.*, paths that satisfy QoS constraints (*e.g.*, delay or bandwidth). Resilience to failures, on the other hand, is achieved by providing, for each primary QoS path, a set of alternative QoS paths used upon a failure of either a link or a node. The above objectives, coupled with the need to minimize the global use of network resources, imply that the cost of both the primary path and the restoration topology should be a major consideration of the routing process.

We undertake a comprehensive study of problems related to finding suitable restoration topologies for QoS paths. We consider both bottleneck QoS constraints, such as bandwidth, and additive QoS constraints, such as delay and jitter. This is the first study to provide a rigorous solution, with proven guaranties, to the combined problem of computing *QoS paths* with *restoration*. It turns out that the widely used approach of disjoint primary and restoration paths is not an optimal strategy. Hence, the proposed algorithms construct a *restoration topology*, *i.e.*, a set of *bridges*, each bridge protecting a portion of the primary QoS path. This approach guaranties to find a restoration topology with low cost when one exists.

In addition to analysis, we test our approach also by way of simulations. The simulation results demonstrate that our proposed approximation algorithms identify QoS restoration paths whose cost is significantly smaller than those provided by alternative approaches.

Index Terms

Restoration, OoS routing, Approximation algorithms.

I. Introduction

There is a growing interest among service providers to offer their customers new revenue-generating services with *Quality of Service* (QoS) guarantees. This is facilitated by current efforts to provide resource reservations and explicit path routing, *e.g.*, *MultiProtocol Label Switching* (MPLS). A key requirement for such services is that they also be resilient to failures. This goal can be achieved by provisioning *primary* and *restoration* paths that satisfy the QoS constraints. The primary QoS path is used during normal network operation; upon failure of a network element (node or link) in the primary path, the traffic is immediately switched to a restoration path. To facilitate this seamless recovery to a restoration path in the event of a failure, it is necessary to reserve network resources (*e.g.*, bandwidth) on both the primary and restoration QoS paths. Thus, since optimizing the utilization of network resources is an important requirement for service providers, it is crucial that the *cost* of the computed primary and restoration QoS paths be as small as possible, where the cost of a path is some measure related to the characteristics of its links, their degree of utilization, *etc*.

QoS constraints occur naturally in a number of practical settings involving bandwidth and delay sensitive applications such as voice over IP, audio and video conferencing, multimedia streaming *etc*. For instance, voice (*e.g.*, telephone conversations) requires a certain bandwidth allocation along the connection path (currently, around 16–64 Kbs) and the end-to-end path delay to be below a certain threshold (typically between

Bell Labs, Lucent Technologies, email: bej@research.bell-labs.com
Bell Labs, Lucent Technologies, email:yuri@research.bell-labs.com
Department of Electrical Engineering, Technion, Israel, email: ariel@ee.technion.ac.il
Bell Labs, Lucent Technologies, email: rastogi@research.bell-labs.com
Correspondence author, Department of Electrical Engineering, Technion, Israel, email: spalex@tx.technion.ac.il

100 and 300 ms). QoS constraints can be divided into *bottleneck* constraints, such as bandwidth and *additive* constraints, such as delay or jitter.

QoS routing has been the subject of several recent studies and proposals (see, *e.g.*, [3], [13], [15], [16] and references therein). However, none of the prior studies on QoS routing consider the problem of provisioning QoS paths with restoration. Similarly, path restoration and routing over alternate paths has also attracted a large body of research (see, *e.g.*, [7], [8], [9], [10], [11]). Most of the proposed solutions, however, consider only bottleneck QoS constraints. The few studies that do consider additive constraints, focus on heuristic approaches and do not provide proven performance guarantees.

Bottleneck QoS constraints can be efficiently handled by pruning infeasible links. However, additive QoS constraints are more difficult to handle. Indeed, the basic problem of finding an optimal path that satisfies an additive QoS constraint is $\mathcal{NP}$-hard [5]. Moreover, it turns out that, in the presence of additive QoS constraints, the widely used approach of disjoint primary and restoration paths is not an optimal strategy. A better solution is to provide a *restoration topology*, *i.e.*, a set of *bridges*, with each bridge protecting a portion of the primary path. The advantage of the disjoint paths strategy is its ability to switch promptly from the primary path to the backup path in the event of a failure. While a restoration topology requires more sophisticated switching with a proper signaling mechanism, it has several advantages over the disjoint paths strategy. First, it provides a cheaper solution in terms of resource consumption. Second, it may find a solution when one does not exist for the disjoint paths strategy [10]. Third, a restoration topology strategy uses fewer backup links upon a failure, which facilitates more efficient sharing of backup bandwidth [9]. Finally, the restoration topology strategy enables the network to recover from a failure by simply activating a local bridge, rather than switching to a completely new path.

Accordingly, this study investigates the problem of provisioning primary and restoration paths that satisfy QoS constraints. Since this problem is $\mathcal{NP}$-hard, we present solutions that are guaranteed to be within a certain factor of the optimum. The paper makes two major contributions. First, we address the issue of link sharing by different bridges, which complicates the process of finding the set of optimal bridges, and we prove that there is an optimal restoration topology in which each link is shared by at most two bridges, both in directed and undirected graphs. This enables us to identify restoration topologies whose cost is at most 2 times more than the optimum. The second contribution is the novel concept of *adjusted delay*, which allows us to represent the set of bridges that compose a restoration topology as a single *walk*[1] between the source and the destination nodes. This concept makes it possible to adapt standard schemes such as Bellman-Ford's Shortest Path algorithm [12] for identification of restoration topologies. To further illustrate the effectiveness of our proposed solutions, we conduct some simulation experiments. The simulation results demonstrate that our proposed algorithms perform well in practice, and in a number of cases, return a restoration topology even when prior approaches fail to do so.

The remainder of the paper is organized as follows. In Section II, we present the network model and formulate the main problems considered in this paper. In Section III, we discuss a fundamental property of optimal restoration topologies, namely, the sharing of links by several bridges. In Section IV, we introduce the basic concepts of adjusted delay and feasible walk. In Section V, we provide approximation schemes for provisioning of the restoration topologies. In Section VI, we extend our results for directed networks. Simulation results are presented and discussed in Section VII. Finally, conclusions are presented in Section VIII.

II. Model and Problem Formulation

In this section, we describe the network model and the two problems addressed in this paper. For simplicity of exposition, we use the terms *bandwidth* and *delay* requirements in order to generically refer to *bottleneck QoS constraints* and *additive QoS constraints*, respectively.

[1] In contrast to a path, a walk may include loops.

A. The Network Model

We respesent the *network* by an undirected graph $G(V, E)$, where V is the set of nodes and E is the set of links. We denote by N and M the number of network nodes and links, respectively, *i.e.*, $N = |V|$ and $M = |E|$. An (s, t)-*walk* is a finite sequence of nodes $\mathcal{W} = (s = v_0, v_1, \cdots, t = v_n)$, such that, for $0 \leq i \leq n-1$, $(v_i, v_{i+1}) \in E$. Here, $n = |\mathcal{W}|$ is the *hop count* of $\mathcal{W}$. Note that nodes and links may appear in a walk several times. An (s, t)-*path* $\mathcal{P}$ is an (s, t)-walk whose nodes are distinct. The subwalk (subpath) of $\mathcal{W}$ ($\mathcal{P}$) that extends from v_i to v_j is denoted by $\mathcal{W}_{(v_i,v_j)}$ ($\mathcal{P}_{(v_i,v_j)}$). Let $\mathcal{W}_1$ be a (v, u)-walk and $\mathcal{W}_2$ be a (u, w)-walk; then, $\mathcal{W}_1 \circ \mathcal{W}_2$ denotes the (v, w)-walk formed by the concatenation of $\mathcal{W}_1$ and $\mathcal{W}_2$.

Each link $l \in E$ offers a bandwidth guarantee b_l (which is typically the available bandwidth on l), and a delay guarantee d_l. The bandwidth $B(\mathcal{W})$ of a walk $\mathcal{W}$ is identical to the bandwidth of its worst link, *i.e.*, $B(\mathcal{W}) = \min_{l \in \mathcal{W}}\{b_l\}$. The delay $D(\mathcal{W})$ of a walk $\mathcal{W}$ is the sum of the QoS requirements of its links, *i.e.*, $D(\mathcal{W}) = \sum_{l \in \mathcal{W}} d_l$.

In order to satisfy QoS constraints, certain resources such as bandwidth and buffer space must be reserved along QoS paths. In order to optimize the global resource utilization, we need to identify QoS paths that consume as few network resources as possible. Accordingly, we associate with each link l a nonnegative cost c_l, which estimates the quality of the link in terms of resource utilization. The link cost may depend on various factors, *e.g.*, the link's available bandwidth and its location. The cost $C(\mathcal{W})$ of a walk $\mathcal{W} = \{v_0, \cdots, v_n\}$ is defined to be the sum of the costs of its links, *i.e.*, $C(\mathcal{W}) = \sum_{i=0}^{n-1} c_{(v_i,v_{i+1})}$.

In order to model networks with nodes connected by asymmetric or unidirectional links, we also consider directed network graphs in this paper (see Section VI). For instance, in such networks, for a pair of connected nodes (v_i, v_j), the bandwidth provisioned on the link in the direction from v_i to v_j may be much larger than the allocated bandwidth in the opposite direction. In addition, the delay and cost characteristics of link (v_i, v_j) may be very different from those of link (v_j, v_i).

B. QoS paths

A fundamental problem in QoS routing is to identify a minimum cost path between a source s and a destination t that satisfies some delay and bandwidth constraints. Bandwidth requirements can be efficiently handled by simply pruning infeasible links from the graph, *i.e.*, links l whose bandwidth b_l is lower than the constraint. Thus, in the rest of the paper, we only consider delay requirements. Accordingly, the fundamental problem is to find a minimum cost path that satisfies a given delay constraint. This can be formulated as the *Restricted Shortest Path* problem.

Problem RSP (Restricted Shortest Path): Given a source node s, a destination node t and a delay constraint $\hat{d}$, find an (s, t)-path $\hat{\mathcal{P}}$ such that

1) $D(\hat{\mathcal{P}}) \leq \hat{d}$, and
2) $C(\hat{\mathcal{P}}) \leq C(\mathcal{P})$ for every other (s, t)-path $\mathcal{P}$ that satisfies $D(\mathcal{P}) \leq \hat{d}$.

In general, Problem RSP is intractable, *i.e.*, $\mathcal{NP}$-hard [5]. However, there exist pseudo-polynomial solutions, which give rise to fully polynomial approximation schemes[2] (FPAS), whose computational complexity is reasonable (see [4], [6], [12]). The most efficient algorithm, presented in [12], has a computational complexity of $O(MH(\frac{1}{\varepsilon} + \log\log H))$, and computes a path with delay at most $\hat{d}$ and cost at most $(1 + \varepsilon)$ times the optimum. We refer to this algorithm as Algorithm RSP.

C. Bridges and a Restoration Topology

As mentioned earlier, our study focuses on provisioning QoS paths with restoration. The QoS path that is used during normal network operation is referred to as the *primary* path. Upon failure of a network element (node or link) in the primary path, the traffic is immediately switched to a *restoration* path. Thus, we require that in addition to the primary path, the restoration paths also satisfy the delay constraint $\hat{d}$. In this paper,

[2] A *Fully Polynomial Approximation Scheme* (FPAS) provides a solution whose cost is at most $(1 + \varepsilon)$ times more than the optimum with a time complexity that is polynomial in the size of the input and $1/\varepsilon$.

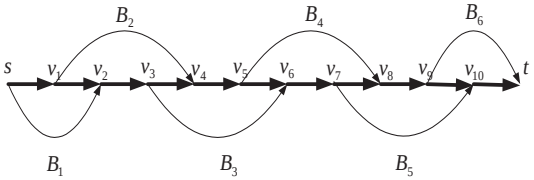

Fig. 1. An example of a network. The delay of all links is 10, except for bold links whose delay is 0.

we primarily focus on link failures, but our results can be easily extended to deal with node failures by using standard node splitting techniques (see, *e.g.*, [17]).

A common approach for path restoration is to provision two disjoint paths that satisfy the delay constraint. However, as we illustrate below (see also [10]), in some cases, such disjoint paths do not exist, although it is possible to provision a primary path with a set of restoration paths. Consider the network depicted in Fig. 1. Here, the delay of all links is 10, except for the links marked by bold lines, whose delay is 1. The only two disjoint paths between the source node s and the destination node t are $\mathcal{P}_1 = \{s, v_1, v_4, v_5, v_8, v_9, t\}$ and $\mathcal{P}_2 = \{s, v_2, v_3, v_6, v_7, v_{10}, t\}$. For a delay constraint $\hat{d} = 20$, $\mathcal{P}_1$ and $\mathcal{P}_2$ cannot be used as primary and restoration paths, because $D(\mathcal{P}_1) = D(\mathcal{P}_2) = 33 > 20$. However, it is possible to provision a primary path $\hat{\mathcal{P}}$ and a set of restoration paths that satisfy the delay constraint $\hat{d} = 20$. Specifically, we use the primary path $\hat{\mathcal{P}} = \{s, v_1, v_2, \cdots, v_{10}, t\}$ and restoration paths $\{\hat{\mathcal{P}}_1, \cdots, \hat{\mathcal{P}}_6\}$ defined as follows. Upon failure of links (s, v_1) or (v_1, v_2) we use restoration path $\hat{\mathcal{P}}_1 = \mathcal{B}_1 \circ \hat{\mathcal{P}}_{(v_2,t)}$ with $D(\hat{\mathcal{P}}_1) = 19$, while upon failure of link (v_2, v_3) path $\hat{\mathcal{P}}_2 = \hat{\mathcal{P}}_{(s,v_1)} \circ \mathcal{B}_2 \circ \hat{\mathcal{P}}_{(v_4,t)}$ with $D(\hat{\mathcal{P}}_2) = 18$ is used. Similarly, we construct restoration paths $\hat{\mathcal{P}}_3, \cdots, \hat{\mathcal{P}}_6$. As demonstrated in the example above, a restoration path comprises of portions of the primary path and a *bridge*, which serves as a backup for the failed segment of the primary path. For example, in Fig. 1, the restoration paths $\hat{\mathcal{P}}_1$ and $\hat{\mathcal{P}}_2$ include the bridges $\mathcal{B}_1 = \{s, v_2\}$ and $\mathcal{B}_2 = \{v_1, v_4\}$, respectively.

Definition 1 (Bridge for a link failure): Let $\hat{\mathcal{P}} = \{s = v_0, \cdots, t = v_n\}$ be a QoS path and $\hat{\mathcal{P}}_{(v_i,v_j)}$ be a subpath of $\hat{\mathcal{P}}$. A path $\mathcal{B}$ between $v_i \in \hat{\mathcal{P}}$ and $v_j \in \hat{\mathcal{P}}$ that has no common links with $\hat{\mathcal{P}}_{(v_i,v_j)}$ is referred to as a *bridge*. We say that bridge $\mathcal{B} = \{v_i, \cdots, v_j\}$ *protects* the subpath $\hat{\mathcal{P}}_{(v_i,v_j)}$ of $\hat{\mathcal{P}}$.

Recall that each restoration path must satisfy the delay constraint $\hat{d}$. This implies that the delay $D(\mathcal{B})$ of a bridge $\mathcal{B}$ must also be constrained. Specifically, the delay of a bridge $\mathcal{B}_i = \{s_i, \cdots, t_i\}$ must be at most $D(\hat{\mathcal{P}}_{(s_i,t_i)}) + \hat{d} - D(\hat{\mathcal{P}})$, where $\hat{\mathcal{P}}_{(s_i,t_i)}$ is the subpath of $\hat{\mathcal{P}}$ protected by $\mathcal{B}$. We denote the quantity $\hat{d} - D(\hat{\mathcal{P}})$ by Δ. Clearly, for larger values of Δ, it is possible to find cheaper bridges $\mathcal{B}_i = \{s_i, \cdots, t_i\}$ that satisfy $D(\mathcal{B}_i) \leq D(\hat{\mathcal{P}}_{(s_i,t_i)}) + \Delta$.

A set of bridges that provides a restoration path for the failure of any link $l \in \hat{\mathcal{P}}$ is referred to as a *restoration topology*.

Definition 2 (Restoration topology for link failures): Let $\hat{d}$ be a QoS constraint and $\hat{\mathcal{P}} = \{s = v_0, \cdots, v_n = t\}$ be a QoS path that satisfies $\hat{d}$ (*i.e.,* $D(\hat{\mathcal{P}}) \leq \hat{d}$). Then, a *restoration topology* $\mathcal{R}$ for $(\hat{\mathcal{P}}, \hat{d})$ is a set of bridges $\{\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_h\}$ such that:

1) for each bridge $\mathcal{B}_i = \{s_i, \cdots, t_i\}$, it holds that $D(\mathcal{B}_i) \leq D(\hat{\mathcal{P}}_{(s_i,t_i)}) + \hat{d} - D(\hat{\mathcal{P}})$, and
2) for each link $l \in \hat{\mathcal{P}}$, there exists a bridge $\mathcal{B}_i = \{s_i, \cdots, t_i\}$ that "protects" l, *i.e.,* l is included in the subpath $\hat{\mathcal{P}}_{(s_i,t_i)}$ of $\hat{\mathcal{P}}$ protected by $\mathcal{B}_i$.

We refer to $\mathcal{R}$ as a *feasible restoration topology*, in order to emphasize that each restoration path satisfies the delay constraint $\hat{d}$.

Let $E(\mathcal{R})$ be the set of links that belong to bridges of $\mathcal{R}$, *i.e.,* $E(\mathcal{R}) = \{l | l \in \mathcal{B}_i, \mathcal{B}_i \in \mathcal{R}\}$. The cost of a restoration topology is defined as the total cost of links in $E(\mathcal{R})$, *i.e.,* $C(\mathcal{R}) = \sum_{l \in E(\mathcal{R})} c_l$. Note that the cost of each link is counted only once, even if it belongs to several bridges. We denote by $|\mathcal{R}| = |E(\mathcal{R})|$ the number of links in the restoration topology.

We seek restoration topologies that minimize the usage of network resources. Since the cost of a link is

a measure of its desirability for routing (with lower cost links being more desirable), our goal is to find a (feasible) restoration topology $\mathcal{R}$ with minimum cost $C(\mathcal{R})$. We will denote by OPT the minimum cost of a restoration topology for $(\hat{\mathcal{P}}, \hat{d})$. Note that, depending on how costs are assigned to links, our approach enables a wide range of restoration topologies to be selected. For example, associating unit costs with all links would translate into computing restoration topologies with a minimum number of links. Also observe that in an optimal restoration topology, the subpaths of $\hat{\mathcal{P}}$ protected by two bridges $\mathcal{B}_i$ and $\mathcal{B}_j$ are not nested, one within the other. Thus, for any two bridges $\mathcal{B}_i = \{s_i, \cdots, t_i\}$ and $\mathcal{B}_j = \{s_j, \cdots, t_j\}$ in $\mathcal{R}$, if s_i precedes s_j in $\hat{\mathcal{P}}$ then t_i also precedes t_j in $\hat{\mathcal{P}}$, and vice versa. For clarity of presentation, we assume that the bridges in $\mathcal{R}$ are enumerated such that the source s_i of bridge $\mathcal{B}_i$ is either identical to or a predecessor of the destination t_{i-1} of bridge $\mathcal{B}_{i-1}$ in $\hat{\mathcal{P}}$.

D. Problem Statement

We are now ready to formulate the two problems that we consider in this study. The first problem seeks to compute a suitable restoration topology for a given QoS path.

Problem RT (Restoration topology for a QoS Path): Given an (s,t)-path $\hat{\mathcal{P}}$ and a QoS constraint $\hat{d}$, such that $D(\hat{\mathcal{P}}) \leq \hat{d}$, find a minimum cost restoration topology $\mathcal{R}$ for $(\hat{\mathcal{P}}, \hat{d})$.

Next, we consider the problem of provisioning a QoS path with a restoration topology.

Problem P+RT (QoS Path and Restoration topology): Given a source s, a destination t and a QoS constraint $\hat{d}$, find an (s,t)-path $\hat{\mathcal{P}}$ that satisfies $D(\hat{\mathcal{P}}) \leq \hat{d}$ and a restoration topology $\mathcal{R}$ for $(\hat{\mathcal{P}}, \hat{d})$ such that their total cost $C(\hat{\mathcal{P}}) + C(\mathcal{R})$ is minimum.

Each of the above problems, namely RT and P+RT, includes Problem RSP as a special case; hence, they are both $\mathcal{NP}$-hard. Furthermore, as discussed below, in most cases we cannot provide an efficient solution without violating the delay constraint in the restoration paths. Accordingly, we introduce the following definition of (α, β)-*approximations*.

Definition 3 ((α, β)*-approximation):* For constants α and β, an (α, β)-approximate solution to, either Problem RT or Problem P+RT is a solution, for which:

1) the cost is at most α times more than the optimum;
2) the primary path satisfies the delay constraint;
3) each restoration path violates the delay constraint by a factor of at most β.

E. Our results

In the following table, we summarize the approximation ratios that we obtain for the above two problems. Below, ε is a constant that captures the trade-off between the quality of the approximations and the running time of the algorithms. Specifically, the time complexity of the algorithms is proportional to $\frac{1}{\varepsilon}$; thus smaller values of ε yield better approximate solutions at the expense of higher running times.

Problem	Undirected	Directed
RT	$(2 \cdot (1+\varepsilon), 1)$	$(2 \cdot (1+\varepsilon), 2)$
P+RT	$(3 \cdot (1+\varepsilon), 2)$	$(3 \cdot (1+\varepsilon), 3)$

The solution of Problem P+RT is the main contribution of this paper. We emphasize that our solutions may violate the delay constraint only for restoration paths, while *the primary paths always satisfy the QoS constraints*. Therefore, such delay violations have no effect during normal network operation. Moreover, many time-sensitive applications can tolerate short-term delay violations (until the failed link is repaired), *e.g.*, by way of buffering.

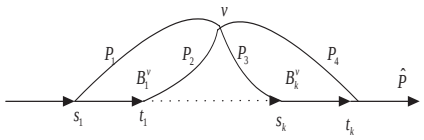

Fig. 2. Node v is shared by bridges $\mathcal{B}_1^v = \mathcal{P}_1 \circ \mathcal{P}_2$ and $\mathcal{B}_k^v = \mathcal{P}_3 \circ \mathcal{P}_4$.

III. Properties of Restoration Topologies

Finding an optimal restoration topology is a complicated problem due to the fact that bridges may share links. However, in this section we establish the existence of an optimal restoration topology in which the number of appearances of a node or a link is bounded by 2. This bound is then used to obtain an approximate scheme within a factor of $2 \cdot (1 + \varepsilon)$ of the optimal solution.

Lemma 1: Given an undirected graph G, a delay constraint $\hat{d}$, an (s,t)-path $\hat{\mathcal{P}}$, $D(\hat{\mathcal{P}}) \leq \hat{d}$, and a restoration topology $\mathcal{R}$ for $(\hat{\mathcal{P}}, \hat{d})$, there exists a restoration topology $\hat{\mathcal{R}}$ for $(\hat{\mathcal{P}}, \hat{d})$ such that $C(\hat{\mathcal{R}}) \leq C(\mathcal{R})$, and each node $v \in \hat{\mathcal{R}}$ or link $l \in \hat{\mathcal{R}}$ is included in at most two bridges.

Proof: Let $\hat{\mathcal{R}}$ be a restoration topology for $(\hat{\mathcal{P}}, \hat{d})$ such that $E(\hat{\mathcal{R}}) \subseteq E(\mathcal{R})$ and $|\hat{\mathcal{R}}|$ is minimum. We prove that each node of $\hat{\mathcal{R}}$ is included in at most two bridges.

By way of contradiction, let $\{\mathcal{B}_1^v, \mathcal{B}_2^v, \cdots, \mathcal{B}_k^v\}$, $k \geq 3$ be the set of bridges of $\hat{\mathcal{R}}$ that contain v, sorted according to their indexes. Since $|\hat{\mathcal{R}}|$ is minimum, it follows that the subpaths $\hat{\mathcal{P}}_{(s_1,t_1)}$ and $\hat{\mathcal{P}}_{(s_k,t_k)}$ of $\hat{\mathcal{P}}$ protected by bridges $\mathcal{B}_1^v$ and $\mathcal{B}_k^v$, respectively, are disjoint (*i.e.*, t_1 is a predecessor of s_k in $\hat{\mathcal{P}}$), otherwise we can omit from $\hat{\mathcal{R}}$ the bridges $\mathcal{B}_2^v, \cdots, \mathcal{B}_{k-1}^v$. We denote the subpaths $\{s_1, \cdots, v\}$ and $\{v, \cdots, t_1\}$ of $\mathcal{B}_1^v$ by $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively (see Fig. 2). The subpaths $\{s_k, \cdots, v\}$ and $\{v, \cdots, t_k\}$ of $\mathcal{B}_k^v$ are denoted by $\mathcal{P}_3$ and $\mathcal{P}_4$, respectively.

The delay of each bridge $\mathcal{B}$ exceeds the delay of the subpath of $\hat{\mathcal{P}}$ protected by $\mathcal{B}$ by at most Δ, *i.e.*:

$$D(\mathcal{P}_1) + D(\mathcal{P}_2) \leq D(\hat{\mathcal{P}}_{(s_1,t_1)}) + \Delta$$
$$D(\mathcal{P}_3) + D(\mathcal{P}_4) \leq D(\hat{\mathcal{P}}_{(s_k,t_k)}) + \Delta$$

It is easy to verify that one of the following two conditions must hold:

$$D(\mathcal{P}_1) + D(\mathcal{P}_3) \leq D(\hat{\mathcal{P}}_{(s_1,t_1)}) + \Delta$$
$$D(\mathcal{P}_2) + D(\mathcal{P}_4) \leq D(\hat{\mathcal{P}}_{(s_k,t_k)}) + \Delta$$

If the first condition holds, then the substitution of $\mathcal{B}_1^v$ in $\hat{\mathcal{R}}$ by $\mathcal{P}_1 \circ \mathcal{P}_3$ yields a valid restoration topology with fewer links than in $\hat{\mathcal{R}}$ which contradicts our assumption that $|\hat{\mathcal{R}}|$ is minimum. Note that this substitution yields a restoration topology with fewer links even if $\mathcal{P}_2$ has common links with $\mathcal{P}_3$ or $\mathcal{P}_4$. Similarly, we can show a contradiction if the second condition holds.

We thus conclude that each node $v \in \hat{\mathcal{R}}$ is shared by at most two bridges. It follows that each link $l \in \hat{\mathcal{R}}$ is also shared by at most two bridges. ■

In Section VI, we derive a similar bound of 2 on the degree of sharing for each link in a directed network graph. But first, in the following two sections, we focus on developing approximation algorithms for Problems RT and P+RT for the undirected graph scenario.

IV. Adjusted Delay and Feasible Walk Concepts

In this section, we introduce the basic concepts of *adjusted delay* and *feasible walk*, which lay the foundations for our efficient approximation algorithms for Problems RT and P+RT, presented in Section V.

A. A Simple Algorithm

In order to set the stage for the concept of adjusted delay, we first present a simple algorithm for Problem RT. The algorithm, at a high level, consists of the following steps. First, we compute for each node pair (v_i, v_j) in $\hat{\mathcal{P}}$, the cheapest bridge $\mathcal{B}_{(i,j)} = \{v_i, \cdots, v_j\}$ whose delay is at most $D(\hat{\mathcal{P}}_{(v_i,v_j)}) + \Delta$. For this end we delete all the links of the path $\hat{\mathcal{P}}$ from G and apply Algorithm RSP [12] to the resulting graph. Next, we construct a restoration topology by selecting a subset of bridges $\{\mathcal{B}_{(i,j)}\}$ such that each link of $\hat{\mathcal{P}}$ is protected and the total cost is minimum. To achieve this, we construct an auxiliary directed graph $\tilde{G}$ whose nodes are essentially the nodes of $\hat{\mathcal{P}}$. Further, for each link $(v_i, v_{i+1}) \in \hat{\mathcal{P}}$ we add to $\tilde{G}$ a link (v_{i+1}, v_i) and assign it a zero cost. Also, for each pair (v_i, v_j) of nodes of $\hat{\mathcal{P}}$, such that $j > i$, we add to $\tilde{G}$ a link $l = (v_i, v_j)$ whose cost is identical to the cost of the bridge $\mathcal{B}_{(i,j)}$. We now show that each (s,t)-path in $\tilde{G}$ corresponds to a feasible restoration topology. Consider an (s,t)-path $\mathcal{P}$ in $\tilde{G}$, and let $\mathcal{S}$ be the set of bridges that correspond to links in $\mathcal{P}$. Consider two successive bridges $\mathcal{B}_i = \{s_i, t_i\}$ and $\mathcal{B}_{i+1} = \{s_{i+1}, t_{i+1}\}$ in $\mathcal{S}$. Note that s_{i+1} either precedes t_i or else coincides with t_i, while t_{i+1} succeeds t_i in the path $\hat{\mathcal{P}}$. This ensures that every link in $\hat{\mathcal{P}}$ is protected by a bridge, and thus $\mathcal{S}$ corresponds to a feasible restoration topology. Hence, a near-optimal restoration topology can be determined by finding the shortest (s,t)-path in $\tilde{G}$. Specifically, Lemma 1 and the fact that Algorithm RSP returns a $(1+\varepsilon)$-approximation for each $\mathcal{B}_{(i,j)}$, jointly imply that a shortest path algorithm (run on $\tilde{G}$) provides a $(2 \cdot (1+\varepsilon), 1)$-approximate solution.

The above algorithm, while conceptually simple, is computationally expensive, because it applies Algorithm RSP [12] for each pair of nodes in $\hat{\mathcal{P}}$. Since the time complexity of the RSP algorithm is $O(MN(1/\varepsilon + \log\log N))$, it requires a total of $O(MN^3(1/\varepsilon + \log\log N))$ time. In the following sections, we describe Algorithm RT, which employs similar ideas, but whose computational complexity is comparable to that of Algorithm RSP.

B. The Adjusted Delay Concept

The algorithm presented in the previous section exploited the relationship between the shortest path in an auxiliary graph and the restoration topology. In this section, we use this idea again, but for devising a more efficient algorithm. We construct a directed auxiliary graph G' from G by reversing each link $l \in \hat{\mathcal{P}}$ and assigning it a zero cost. We denote by $\hat{\mathcal{P}}'$, the path in G' formed from $\hat{\mathcal{P}}$ by reversing its links. In addition, we also substitute each link $l = (u,v) \in G, l \notin \hat{\mathcal{P}}$ by two directed links $l_1 = (u,v)$ and $l_2 = (v,u)$ such that $c_{l_1} = c_{l_2} = c_l$ and $d_{l_1} = d_{l_2} = d_l$. Clearly, each (s,t)-walk in the auxiliary graph G' corresponds to a set of bridges that protects each link $l \in \hat{\mathcal{P}}$. For example, Fig. 3 depicts the auxiliary graph for the network depicted in Fig. 1 and the primary path $\hat{\mathcal{P}} = \{s, v_1, v_2, \cdots, t\}$. The walk $\mathcal{W} = \{s, v_2, v_1, v_4, v_3, \cdots, t\}$ in auxiliary graph corresponds to a set of bridges $\{\mathcal{B}_1, \cdots, \mathcal{B}_6\}$. In general, however, as explained below, not every (s,t)-walk in the auxiliary graph corresponds to a *feasible* restoration topology, *i.e.*, one that satisfies the delay constraint.

One of the key contributions of this study is an efficient method for verifying, during its construction, whether a given walk represents a feasible restoration topology. This method is used as a basic building block in our algorithm, and enables us to find a low-cost feasible restoration topology. In order to identify a walk in G' that corresponds to a feasible restoration topology, we introduce the notion of *adjusted delay* for a walk $\mathcal{W}$ in G'.

Consider a walk $\mathcal{W} = \{s = u_0, \cdots, u_{k-1}, u_k, \cdots, t\}$ in G' that defines a set $\mathcal{S} = \{\mathcal{B}_i = \{s_i, \cdots, t_i\}\}$ of bridges, and let $\mathcal{P}_i$ be the restoration path obtained by activating the bridge $\mathcal{B}_i$; thus, $\mathcal{P}_i = \hat{\mathcal{P}}_{(s,s_i)} \circ \mathcal{B}_i \circ \hat{\mathcal{P}}_{(t_i,t)}$. We refer to nodes s_i and t_i as the start-point and termination-point of bridge $\mathcal{B}_i$, respectively. Recall that a bridge $\mathcal{B}_i$ satisfies the delay constraint only if $D(\mathcal{B}_i) \leq D(\hat{\mathcal{P}}_{(s_i,t_i)}) + \Delta$, or, equivalently,

$$D(\mathcal{P}_{i(s,t_i)}) \leq D(\hat{\mathcal{P}}_{(s,t_i)}) + \Delta. \tag{1}$$

Furthermore, every subwalk $\mathcal{W}_{(s,u)}$ of $\mathcal{W}$ corresponds to a subset $\mathcal{S}'$ of complete bridges in $\mathcal{S}$ and, possibly, a subpath of an additional bridge $\mathcal{B}_j \in \mathcal{S}$. The adjusted delay $\tilde{D}(\mathcal{W}_{(s,u)})$ of the walk $\mathcal{W}_{(s,u)}$ maintains the

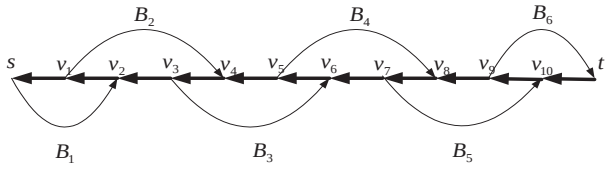

Fig. 3. An auxiliary graph for the network depicted on Fig. 1.

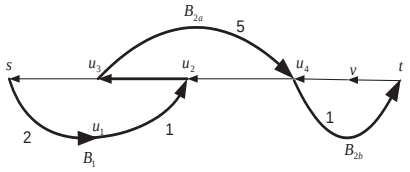

Fig. 4. An example of a walk in the auxiliary graph G'.

following invariant: if all the bridges in $\mathcal{S}'$ satisfy the delay constraint, then $\tilde{D}(\mathcal{W}_{(s,u)})$ represents the delay between the source node s and the node $u \in \mathcal{B}_j$ along $\mathcal{P}_j$, *i.e.*, $\tilde{D}(\mathcal{W}_{(s,u)}) = D(\mathcal{P}_{j\,(s,u)})$. Otherwise, if there is a bridge in $\mathcal{S}'$ that does not satisfy the delay constraint, $\tilde{D}(\mathcal{W}_{(s,u)})$ is set to infinity, thus indicating that the restoration topology formed by the walk $\mathcal{W}_{(s,u)}$ is infeasible. Thus, by applying Condition (1), the adjusted delay enables us to check easily whether bridge $\mathcal{B}_j$ satisfies the delay constraint, when its termination-point t_j is reached.

The adjusted delay $\tilde{D}(\mathcal{W})$ of a walk $\mathcal{W} = \{s = u_0, \cdots, u_{k-1}, u_k, \cdots, t\}$ is calculated in a recursive manner. The adjusted delay of an empty walk is zero, that is $\tilde{D}(\{s\}) = 0$. Now, let us turn to compute the adjusted delay of $\mathcal{W}_{(s,u_k)}$ and suppose that we have already calculated the adjusted delay of the sub-walk $\mathcal{W}_{(s,u_{k-1})}$. Let $D = \tilde{D}(\mathcal{W}_{(s,u_{k-1})}) + d_{(u_{k-1},u_k)}$, where $d_{(u_{k-1},u_k)}$ is the delay of the link (u_{k-1}, u_k). Generally speaking, the adjusted delay $\tilde{D}(\mathcal{W}_{(s,u_k)}) = D$, except for the case when $u_k \in \hat{\mathcal{P}}$. In this case, a special procedure is required for verifying if node u_k is the termination-point of a bridge and whether the newly formed bridge satisfies the delay constraint. Recall that $u_k \in \hat{\mathcal{P}}$ is not necessarily a termination-point of a bridge, since a bridge may have several common nodes with the primary path. For instance, in Fig. 4, bridge $\mathcal{B}_2$ comprises of the two segments $\mathcal{B}_{2a}$ and $\mathcal{B}_{2b}$, and node $u_4 \in \hat{\mathcal{P}}$ is not a termination-point of a bridge. However, if for link (u_k, u_{k+1}) of $\mathcal{W}$, it holds that $(u_k, u_{k+1}) \in \hat{\mathcal{P}}'$, then u_k must be the termination-point of a bridge since a bridge cannot share links with the primary path. Recall that $\hat{\mathcal{P}}' \in G'$ is a path from path $\hat{\mathcal{P}}$ by reversing its links. As illustrated in Fig. 4, node u_2 must be the termination-point of the bridge $\mathcal{B}_1$, since its successor node in the walk, node u_3, is also included in $\hat{\mathcal{P}}$.

Based on the above observations, the adjusted delay of a walk ending at node $u_k \in \hat{\mathcal{P}}$ is defined as follows. (Below, $D = \tilde{D}(\mathcal{W}_{(s,u_{k-1})}) + d_{(u_{k-1},u_k)}$).

- **Case 1**: If $u_{k-1} \notin \hat{\mathcal{P}}$, $u_k \in \hat{\mathcal{P}}$ and $D > D(\hat{\mathcal{P}}_{(s,u_k)}) + \Delta$, then node u_k cannot be the termination-point of a valid bridge and it may only be an internal node of a bridge. Thus, $\tilde{D}(\mathcal{W}_{(s,u_k)}) = D$.
- **Case 2**: If $u_{k-1} \notin \hat{\mathcal{P}}$, $u_k \in \hat{\mathcal{P}}$ and $D \leq D(\hat{\mathcal{P}}_{(s,u_k)}) + \Delta$ then node u_k may be either the termination-point or an internal node of a bridge. As a result, since u_k may be the starting-point of a new bridge, we set $\tilde{D}(\mathcal{W}_{(s,u_k)}) = \min\{D(\hat{\mathcal{P}}_{(s,u_k)}), D\}$.
- **Case 3**: If $(u_{k-1}, u_k) \in \hat{\mathcal{P}}'$, and $\tilde{D}(\mathcal{W}_{(s,u_{k-1})}) \leq D(\hat{\mathcal{P}}_{(s,u_{k-1})})$, then it follows that node u_k is not a termination-point of a bridge and all the bridges included by the walk satisfy the delay constraint. Since u_k may be the starting-point of a new bridge, its adjusted delay $\tilde{D}(\mathcal{W}_{(s,u_k)}) = D(\hat{\mathcal{P}}_{(s,u_k)})$.
- **Case 4**: If $(u_{k-1}, u_k) \in \hat{\mathcal{P}}'$, and $\tilde{D}(\mathcal{W}_{(s,u_{k-1})}) > D(\hat{\mathcal{P}}_{(s,u_{k-1})}) + \Delta$, then it follows that, if node u_{k-1} is the termination-point of a bridge (that is, the predecessor of u_{k-1} in walk $\mathcal{W}$ does not belong to $\hat{\mathcal{P}}$), then the bridge ending at u_{k-1} does not satisfy the delay constraint (due to Condition (1)). On the other

hand, if u_{k-1} is not the termination-point of a bridge, then an induction argument can be used to show that some bridge preceding u_{k-1} in walk $\mathcal{W}$ does not satisfy the delay constraint. Hence, $\tilde{D}(\mathcal{W}_{(s,u_k)})$ is set to ∞.

We proceed to present a formal definition of adjusted delay that considers the four cases mentioned above.

Definition 4 (Adjusted Delay): The *adjusted delay* $\tilde{D}(\mathcal{W})$ of a walk $\mathcal{W} = \{s = u_0, \cdots, u_{k-1}, u_k\}$ in the auxiliary graph G' is defined recursively as follows:

1) The adjusted delay of an empty walk (*i.e.,* $k = 0$) is 0: $\tilde{D}(\{s\}) = 0$;
2) Otherwise, the adjusted delay $\tilde{D}(\mathcal{W})$ of a walk $\mathcal{W} = \{u_0, \cdots, u_{k-1}, u_k\}$ is

$$\tilde{D}(\mathcal{W}) = \begin{cases} \min\{D(\hat{\mathcal{P}}_{(s,u_k)}), D\} & \textbf{if } u_{k-1} \notin \hat{\mathcal{P}}\ , u_k \in \hat{\mathcal{P}}\ and\ D \leq D(\hat{\mathcal{P}}_{(s,u_k)}) + \Delta, \\ D(\hat{\mathcal{P}}_{(s,u_k)}) & \textbf{if } (u_{k-1}, u_k) \in \hat{\mathcal{P}}'\ and\ \tilde{D}(\mathcal{W}_{(s,u_{k-1})}) \leq D(\hat{\mathcal{P}}_{(s,u_{k-1})}), \\ \infty & \textbf{if } (u_{k-1}, u_k) \in \hat{\mathcal{P}}'\ and\ \tilde{D}(\mathcal{W}_{(s,u_{k-1})}) > D(\hat{\mathcal{P}}_{(s,u_{k-1})}), \\ D & otherwise, \end{cases}$$

where $D = \tilde{D}(\mathcal{W}_{(s,u_{k-1})}) + d_{(u_{k-1},u_k)}$ and $\hat{\mathcal{P}}' \in G'$ is a path formed from $\hat{\mathcal{P}}$ by reversing its links.

We illustrate the calculation of the adjusted delay using the walk shown in bold in Fig. 4. Here, the primary path $\hat{\mathcal{P}} = \{s, u_3, u_2, u_4, v, t\}$. The delay of every link $l \in \hat{\mathcal{P}}$ is $d_l = 1$ and the delay constraint $\hat{d} = 7$. Thus, $D(\hat{\mathcal{P}}) = 5$ and $\Delta = 2$. The delay of every other link $l \notin \hat{\mathcal{P}}$ is depicted in the figure. Let us calculate the adjusted delay of the walk $\mathcal{W}_{(s,t)}$ and its various prefixes. At the base of the recursion, $\tilde{D}(\{s\}) = 0$. Since, node $u_1 \notin \hat{\mathcal{P}}$, $\tilde{D}(\mathcal{W}_{(s,u_1)}) = 2$. For computing the adjusted delay of node u_2 we calculate the value $D = \tilde{D}(\mathcal{W}_{(s,u_1)}) + d_{(u_1,u_2)} = 3$. Since $D \leq D(\hat{\mathcal{P}}_{(s,u_2)}) + \Delta = 4$, node u_2 may be either the termination-point of a bridge that satisfies the delay constraint or an internal node of a bridge. To allow these two possibilities, we set $\tilde{D}(\mathcal{W}_{(s,u_2)}) = \min\{D(\hat{\mathcal{P}}_{(s,u_2)}), D\} = \min\{2, 3\} = 2$. Since $(u_2, u_3) \in \hat{\mathcal{P}}'$, node u_2 must be the termination-point of the bridge $\mathcal{B}_1$. Since the bridge $\mathcal{B}_1$ satisfies the delay constraint, we have $\tilde{D}(\mathcal{W}_{(s,u_3)}) = D(\hat{\mathcal{P}}_{(s,u_3)}) = 1$. Now, to compute $\tilde{D}(\mathcal{W}_{(s,u_4)})$, we calculate the corresponding value $D = \tilde{D}(\mathcal{W}_{(s,u_3)}) + d_{(u_3,u_4)} = 6$. Because $D > D(\hat{\mathcal{P}}_{(s,u_4)}) + \Delta = 3 + 2 = 5$, node u_4 can only be an internal node inside a bridge and its adjusted delay is $\tilde{D}(\mathcal{W}_{(s,u_4)}) = 6$. Finally, when computing the adjusted delay of $\mathcal{W}_{(s,t)}$, $D = \tilde{D}(\mathcal{W}_{(s,u_4)}) + d_{(u_4,t)} = 7$. Node t is the termination-point of a valid bridge $\mathcal{B}_2$ and $\tilde{D}(\mathcal{W}_{(s,t)}) = \min\{D(\hat{\mathcal{P}}), D\} = 5$. We conclude that the given walk represents a feasible restoration topology.

C. Feasible Walk Concept

An (s,t)-walk whose adjusted delay does not exceed the delay $D(\hat{\mathcal{P}})$ of the primary path $\hat{\mathcal{P}}$ is referred to as a *feasible* (s,t)*-walk*. From Lemma 2 below, it follows that there is a one-to-one correspondence between feasible walks and feasible restoration topologies.

Lemma 2:

1) Let $\mathcal{R} = \{\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_h\}$, $\mathcal{B}_i = \{s_i, \cdots, t_i\}$, be a feasible restoration topology for $(\hat{\mathcal{P}}, \hat{d})$ and let $\mathcal{W}$ be an (s,t)-walk in G' that corresponds to $\mathcal{R}$, *i.e.,* $\mathcal{W} = \{\mathcal{B}_1 \circ \hat{\mathcal{P}}_{(t_1,s_2)} \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_h\}$. Then, $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$.
2) Let $\mathcal{W}$ be an (s,t)-walk in G' such that $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$. Then, it is possible to decompose $\mathcal{W}$ into a set of bridges $\{\mathcal{B}_1, \cdots, \mathcal{B}_h\}$ that forms a feasible restoration topology.

Proof: See Appendix. ■

In general, there may be more than one way to decompose a walk into a set of bridges, *i.e.,* there are several sets of bridges that can be constructed out of a single walk. For example, in Fig. 4 there are two sets of bridges S_1 and S_2 that correspond to walk $\mathcal{W}$: S_1 is formed by bridges $\{\mathcal{B}_1, \mathcal{B}_{2a}, \mathcal{B}_{2b}\}$, while in S_2, the bridges $\mathcal{B}_{2a}$ and $\mathcal{B}_{2b}$ are combined into a single bridge $\mathcal{B}_2$. Note that only some of these sets constitute feasible restoration

topologies. One can construct a feasible restoration topology from a feasible walk by simply choosing as the termination-point for a bridge, the first node $u_k \in \hat{\mathcal{P}}$ in the bridge for whom $\tilde{D}(\mathcal{W}_{(s,u_k)}) = D(\hat{\mathcal{P}}_{(s,u_k)})$.

We denote by $\mathcal{W}^{opt}$ the minimum cost feasible (s,t)-walk in the auxiliary graph G' and by c^{opt} the cost of $\mathcal{W}^{opt}$. There is a relationship between $\mathcal{W}^{opt}$ and the optimum restoration topology (whose cost is denoted by OPT). Indeed, by Lemmas 1 and 2, there exists a feasible walk $\mathcal{W}$ such that $C(\mathcal{W}) \leq 2 \cdot OPT$. Thus, $c^{opt} \leq 2 \cdot OPT$. Further, note that the cost of a restoration topology constructed from a walk never exceeds the cost of the walk itself. Thus, if we could compute the optimal feasible walk, then we could compute a (2,1)-approximation to the optimal restoration topology.

V. Approximation Schemes for Provisioning of Restoration Topologies

We are now in a position to present efficient approximation schemes for Problems RT and P+RT. We begin with a pseudo-polynomial algorithm for Problem RT, which serves as the basic building block for the approximation scheme for Problem RT presented in Section V-B. Finally, in Section V-C we present the approximation scheme for Problem P+RT.

A. Pseudo-polynomial Solution for Problem RT

The fundamental concepts of adjusted delay and feasible (s,t)-walk give rise to a pseudo-polynomial algorithm for Problem RT, *i.e.*, an algorithm whose running time is proportional to the cost of the optimal solution. The algorithm, referred to as Algorithm PP is presented in this section. Because of its simplicity, the algorithm can be easily implemented in practice. However, in the worst case, its running time can be very high.

The purpose of Algorithm PP is to check, for a given value of upper bound U, whether there exists an (s,t)-walk $\mathcal{W}$ in G', such that $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$ and $C(\mathcal{W}) \leq U$, and if so, to find a minimum cost (s,t)-walk $\mathcal{W}$ such that $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$ and the corresponding restoration topology. The algorithm is a natural extension of the well-known Bellman-Ford algorithm and uses the dynamic-programming technique of *relaxation* [2]. The algorithm assumes that the costs of links are integer values greater than 0, and an upper bound U on the cost of the solution is given.

We first describe the *relaxation* technique used by Algorithm PP. For each node $v_i \in V$, we maintain an array $D_{v_i}[c]$ of *minimum delay estimates*. The array $D_{v_i}[c]$ stores, for each cost c, the minimum adjusted delay of an (s,v_i)-walk, whose cost is at most c. Initially, $D_{v_i}[c] = \infty$ for every $v_i \in G' \backslash s$ and $c \geq 0$. We only relax links whose cost does not exceed the current budget restriction c. The process of relaxing a link (v_i, v_j) consists of testing whether we can improve the best (s,v_j)-walk (*i.e.*, the walk whose adjusted delay is minimum) found so far to v_j by going through v_i without exceeding the current budget restriction c and if so, updating $D_{v_j}[c]$. The relaxation technique is implemented by Procedure Relax (see Fig. 5).

Next, we proceed to describe Algorithm PP, whose pseudo-code appears in Fig. 5. We start with a zero budget for c and increment it by a value of 1 in each iteration until either $D_t[c] \leq D(\hat{\mathcal{P}})$, *i.e.*, there exists a walk between s and t whose adjusted delay is at most $D(\hat{\mathcal{P}})$, or else $c = U$. In each iteration, we process each node $v_j \in G'$ by relaxing all links entering v_j.

As discussed below, in our approximation scheme, Algorithm PP is applied to graphs in which $c_l \geq 1$ for each link l. Thus, the cost of each link $l \in G'$ is at least 1, except for links in G' that originate from the primary path $\hat{\mathcal{P}}$. Note that for each link (v_i, v_j) with zero cost, node v_i must be processed before v_j. Accordingly, the nodes $v_j \in G'$ are processed in an order such that v_j is before $v_{j'}$ if v_j is a successor of $v_{j'}$ in $\hat{\mathcal{P}}$.

Also, in Step 15, the algorithm identifies a walk $\mathcal{W} = \{s = u_0, \cdots, u_k = t\}$ whose adjusted delay is at most $D(\hat{\mathcal{P}})$ using backtracking. Suppose that c is the value at which Algorithm PP breaks out of the for loop (in Step 16), that is, $D_t[c] \leq D(\hat{\mathcal{P}})$. Then, beginning with node t, for each node u_i, the backtracking procedure adds to the returned walk, the link (u_{i-1}, u_i) that resulted in the value $D_{u_i}[c_i]$ until node s is reached, where the c_i values for u_i are computed as follows: for the initial node $u_k = t, c_k = c$ and for every

Algorithm PP $(G(V,E), \hat{\mathcal{P}}, \hat{d}, U)$*:*

parameters:
$G(V,E)$ - network,
$\{d_l, c_l\}_{l \in E}$ - delays and costs of the network links,
$\hat{\mathcal{P}} = \{s = v_0, v_1, ..., t = v_n\}$ - QoS path,
$\hat{d}$- delay constraint,
U- the upper bound on the cost of $\mathcal{R}$.

```
1  Δ ← d̂ − D(P̂)
2  E' ← E
3  for each link l = (v_i, v_{i+1}) ∈ P̂ do
4     E' ← E' \ {(v_i, v_{i+1}) ∈ P̂}
5     E' ← E' ∪ {(v_{i+1}, v_i) ∈ P̂}, c_{(v_{i+1}, v_i)} ← 0
6  for all v_i ∈ V do
7     D_{v_i}[0] ← ∞
8  D_s[0] ← 0
9  for c = 1, 2, ..., U do
10    for each v_j ∈ V in order such that v_j is before v_{j'} if
          v_j is a successor of v_{j'} in P̂ do
11       D_{v_j}[c] ← D_{v_j}[c − 1]
12       for each link l = (v_i, v_j) ∈ E' do
13          RELAX(l(v_i, v_j), c, Δ)
14    if D_t[c] ≤ D(P̂) then
15       determine walk W by backtracking
16       return W.
17 return FAIL

Procedure RELAX (l = (v_i, v_j), c, Δ)
1  if v_j ∈ P̂ and v_i ∈ P̂ then
2     if D_{v_i}[c] ≤ D(P̂_{(s,v_i)}) then
3        D_{v_j}[c] ← min{D_{v_j}[c], D(P̂_{(s,v_j)})}
4  else
5     if c_l ≤ c then
6        D_{v_j}[c] ← min{D_{v_j}[c], D_{v_i}[c − c_l] + d_l)}
7     if v_j ∈ P̂ and D_{v_j}[c] ≤ D(P̂_{(s,v_j)}) + Δ
        then
8        D_{v_j}[c] ← min{D_{v_j}[c], D(P̂_{(s,v_j)})}
```

Fig. 5. Algorithm PP

subsequent node u_{i-1}, $c_{i-1} = c_i - c_{(u_{i-1},u_i)}$. Thus, the cost of the walk $\mathcal{W}$ can be shown to be at most c. The computational complexity of Algorithm PP is $O(M \cdot U)$.

Theorem 1: Let c^{opt} denote the minimum cost of a feasible (s,t)-walk in G'. If $U \geq c^{opt}$ then Algorithm PP returns a minimum cost feasible walk $\mathcal{W}$.

Proof: See Appendix. ■

B. Approximation scheme for Problem RT

In this section, we develop an FPAS for computing the minimum cost feasible (s,t)-walk, and use this near-optimal walk to construct a near-optimal restoration topology. The technique we use is similar to the one presented in [12].

We begin with a high-level overview of the approximation scheme. A basic building block of the scheme is Procedure SCALE (Fig. 6), which uses *scaling* and *rounding* in order to efficiently find an approximate solution. The efficiency of Procedure SCALE depends on the tightness of the lower and upper bounds, L, U, on the cost of the optimal solution. Thus, to compute sufficiently tight lower and upper bounds, we rely on two procedures, namely Procedure BOUND and Procedure TEST. The former is used for obtaining initial

values of L, U such that $U/L < 2 \cdot N$, while the latter performs iterations to tighten the bounds further. Finally, we combine all the ideas in Procedure RT.

1) Scaling and Rounding: We proceed to describe Procedure SCALE (see Fig. 6). The main idea is to scale and round the cost c_l of each link $l \in E$, replacing it by c'_l, as follows:

$$c'_l = \left\lfloor \frac{c_l}{S} \right\rfloor + 1,$$

where $S = \frac{L\varepsilon}{2N}$. Then, we apply Algorithm PP on the resulted graph. Clearly, with the new costs c'_l, there must exist a feasible walk with cost at most $\left\lfloor \frac{c^{opt}}{S} \right\rfloor + 2N$ and no more than $2N$ links (due to Lemma 1). Thus, by Theorem 1 the actual cost of the path returned by Algorithm PP is no more than $c^{opt} + 2NS \leq (1+\varepsilon) \cdot c^{opt}$. We summarize this discussion in the following lemma:

Lemma 3: If Procedure SCALE is invoked with valid upper bound, *i.e.*, $c^{opt} \leq U$, then it returns a walk whose cost is at most $(1+\varepsilon)$ times more than the optimum.

We note that Procedure SCALE might return a feasible walk even if it is invoked with non valid upper bound, *i.e.*, $c^{opt} > U$. However, we show that the cost of a walk $\mathcal{W}$ returned by the algorithm always satisfies $C(\mathcal{W}) \leq L\varepsilon + U$. We use this property later to compute tight lower and upper bounds on c^{opt}.

Lemma 4: Any walk $\mathcal{W}$ returned by Procedure SCALE is feasible and satisfies $C(\mathcal{W}) \leq L\varepsilon + U$.

Proof: Let $\hat{\mathcal{W}}$ be a walk returned by Procedure SCALE. Note that $\hat{\mathcal{W}}$ was returned by Algorithm PP. By Theorem 1 (Part 1), $\hat{\mathcal{W}}$ is a feasible walk. Since $C'(\hat{\mathcal{W}}) \leq \tilde{U} = \left\lfloor \frac{U}{S} \right\rfloor + 2N$, and since $c_l \leq c'_l \cdot S$ for each $l \in E$, we have $C(\hat{\mathcal{W}}) \leq U + 2NS = L \cdot \varepsilon + U$. ■

The running time of Procedure SCALE is $O(\frac{MNU}{\varepsilon L})$. Thus, if we can compute tight lower and upper bounds on c^{opt} such that the ratio U/L is a constant, then we can reduce the computation time of Procedure SCALE to $O(\frac{MN}{\varepsilon})$. We next show how to compute these tight bounds.

2) Lower and upper bounds: In this subsection, we present Procedure BOUND (see Fig. 6), which identifies lower and upper bounds L, U on the minimum cost c^{opt} of a feasible walk c^{opt} such that $U/L \leq 2N$.

We denote by $c^1 < c^2 < \cdots < c^r$ the distinct costs values of the links. Our goal is to find the maximum cost value $c^* \in \{c^i\}$ such that the graph G'' derived from G' by omitting all links whose cost is greater than c^*, does not contain a feasible (s,t)-walk. Clearly, a feasible (s,t)-walk contains at least one link whose cost is c^* or more, hence c^* is a lower bound on c^{opt}. In addition, there exists a feasible (s,t)-walk that comprises of links whose cost is c^* or less, and whose hop count is, by Lemma 1, at most $2N$. We conclude that $2N \cdot c^*$ is an upper bound on c^{opt}.

Procedure BOUND performs a binary search on the values $c^1, c^2, \cdots, c^r$. At each iteration, we need to check whether $c \leq c^*$, where c is the current estimate of c^*. For this purpose, we remove from G all links whose cost is more than c, and assign the unit cost to the remaining links. Then, we apply Algorithm PP on the resulting graph, with the parameter $U = 2N$. If Algorithm PP returns a feasible walk, then $c \geq c^*$; otherwise, $c < c^*$. The computational complexity of Procedure BOUND is $O(MN \log N)$.

3) A testing procedure: In order to tighten the bounds further, we make use of Procedure TEST (shown in Fig. 6). Procedure TEST performs the following 2-approximation test: if the procedure returns a positive answer, then definitely $c^{opt} < 2B$; otherwise, it is the case that $c^{opt} \geq B$.

Procedure TEST is implemented by invoking Procedure SCALE with $U = L = B$ and $\varepsilon = 1$.

Lemma 5: If Procedure TEST returns a positive answer, then $c^{opt} \leq B(1+\varepsilon)$. Otherwise, $c^{opt} > B$.

Proof: If Procedure TEST returns a positive answer, then Procedure SCALE does not fail. Thus, by Lemma 4, Procedure SCALE returns a feasible walk $\hat{\mathcal{W}}$, for which $C(\hat{\mathcal{W}}) \leq B(1+\varepsilon)$, hence $c^{opt} \leq C(\hat{\mathcal{W}}) \leq B(1+\varepsilon)$.

We proceed to prove the second part of the lemma. By way of contradiction, assume that $c^{opt} \leq B$. By Lemma 3, Procedure SCALE does not return FAIL, hence Procedure TEST returns a positive answer, which contradicts the condition of the lemma.

■

4) Putting it all together: We are now in a position to combine the results of the previous subsections in order to present our final approximation algorithm, referred to as Algorithm RT (see Fig. 6).

The algorithm begins by applying Procedure BOUND, which provides the lower and upper bounds L and U on c^{opt} such that $U/L \leq 2N$. Then, we iteratively apply Procedure TEST to improve these bounds until the ratio U/L falls below 8. In each iteration, we invoke Procedure TEST with $B = \sqrt{L \cdot U}$. If Procedure TEST returns a positive answer, then, $c^{opt} < 2B$, hence U is set to $2B$. Otherwise, it is the case that $c^{opt} > B$, hence L is set to B. Note that, if the ratio U/L is equal to x at the beginning of an iteration, then at the end of the iteration we have $(U/L) \leq 2\sqrt{x}$. Thus, since the above process terminates once $U/L \leq 8$, the number of iterations performed can be shown to be $O(\log \log N)$.

Having obtained lower and upper bounds L, U such that $U/L \leq 8$, we use Procedure SCALE to find a feasible walk $\mathcal{W}$, whose cost is at most $(1+\varepsilon) \cdot c^{opt}$. Finally, we return the restoration topology corresponding to $\mathcal{W}$. The computational complexity of Algorithm RT is $O(MN(1/\varepsilon + \log N))$.

Algorithm RT $(G(V,E), \hat{\mathcal{P}}, \hat{d}, \varepsilon)$*:*

```
1  L, U ← BOUND(G(V,E), P̂, d̂)
2  do
3     B ← sqrt(L(c^opt) · U(c^opt))
4     if TEST(G(V,E), P̂, d̂, B, ε) returns YES then
5        L ← B
6     else
7        U ← 2 · B
8  until U/L ≤ 8.
9  W ← SCALE(G(V,E), P̂, d̂, L, U, ε)
10 return the restoration topology that corresponds to W.
```

Procedure SCALE$(G(V,E), \hat{\mathcal{P}}, \hat{d}, L, U, \varepsilon)$

```
1  S ← Lε/(2N)
2  for each link l ∈ E do
3     c'_l ← ⌊c_l/S⌋ + 1
4  Ũ ← ⌊U/S⌋ + 2N
5  return PP(G(V,E), {d_l, c'_l}_{l∈E}, P̂, d̂, Ũ)
```

Procedure TEST$(G(V,E), \hat{\mathcal{P}}, \hat{d}, B)$

```
1  Apply Procedure SCALE for (G(V,E), P̂, d̂, B, B, 2)
2  if Algorithm SCALE returned FAIL then
3     return NO
4  else
5     return YES
```

Procedure BOUND$(G(V,E), \hat{\mathcal{P}}, \hat{d})$

```
1  let c^1 < c^2 < ··· < c^r the distinct costs values of the
   links.
2  low ← 0; high ← r
3  while low < high − 1
4     j ← ⌊(high + low)/2⌋
5     E' ← {l | c_l ≤ c^j}
6     set c_l ← 1 for each l ∈ E'
7     apply Algorithm PP on (G'(V,E'), P̂, d̂, 2N)
8     if Algorithm PP returned FAIL then
9        high ← j
10    else
11       low ← j
12 U ← 2N · c^high; L ← c^high;
13 return L, U;
```

Fig. 6. Algorithm RT

Lemma 6: The computational complexity of Algorithm RT is $O(MN(1/\varepsilon + \log N))$.

Proof: Procedure BOUND requires $O(MN \log N)$ time; the execution of Procedure SCALE in line 9 requires $O(MN/\varepsilon)$ time.

We proceed to analyze the computational complexity of the loop of lines 2-8. We denote by x_i the ratio U/L at the beginning of iteration i. Initially, we have $x_1 \leq 2N$. As discussed above at iteration i, $x_i \leq 2\sqrt{x_{i-1}}$. It follows that

$$x_i \leq 2 \cdot 2^{1/2} \cdot 2^{1/4} \cdot \dots \cdot 2^{1/2^i} \cdot (2N)^{1/2^i} \leq 4 \cdot (2N)^{1/2^i}.$$

At iteration $k = \log\log(2N)$ we have $x_k \leq 8$. We conclude that the loop performs $O(\log\log N)$ iterations. At each iteration we execute Procedure TEST, which requires $U = O(MN/\varepsilon)$ time. We conclude that the total running time of the loop is $O(MN \log\log N)$.

We conclude that the total running time of Algorithm RT is $O(MN(1/\varepsilon + \log N))$. ■

Theorem 2: Given an undirected graph G, a primary QoS path $\hat{\mathcal{P}} \in G$, a delay constraint $\hat{d}$ and an approximation ratio ε, Algorithm RT identifies, a feasible restoration topology $\mathcal{R}$ for $(\hat{\mathcal{P}}, \hat{d})$, whose cost is at most $2 \cdot (1+\varepsilon)$ times more than the optimum.

Proof: By Lemma 5, lines 4-7 ensure that at each iteration L and U are valid bounds, *i.e.*, $L \leq c^{opt} \leq U$. Thus, Procedure SCALE is called at line 9 with valid bounds, hence by Lemma 3 it finds a feasible walk $\mathcal{W}$ whose cost $C(\mathcal{W}) \leq (1+\varepsilon) \cdot c^{opt}$. Further, from Lemmas 1 and 2, it follows that $c^{opt} \leq 2 \cdot OPT$. Thus, $C(\mathcal{W}) \leq 2(1+\varepsilon) \cdot OPT$ and the cost of the restoration topology $\mathcal{R}$ corresponding to $\mathcal{W}$ satisfies $C(\mathcal{R}) \leq 2(1+\varepsilon) \cdot OPT$. ■

C. Approximation Scheme for Problem P+RT

The approximation scheme for simultaneous provisioning of a primary QoS path and the restoration topology is implemented as follows. First, using Algorithm RSP, we identify a $\hat{d}$-delay constrained (s,t)- path $\hat{\mathcal{P}}$ in G whose cost is at most $(1+\varepsilon)$ times the optimum. Then we apply Algorithm RT with parameters G, $\hat{\mathcal{P}}$, $(\hat{d} + D(\hat{\mathcal{P}}))$ and ε. The resulting algorithm is referred to as Algorithm P+RT.

Theorem 3: Algorithm P+RT identifies, in $O(MN(1/\varepsilon + \log N))$ time, a $(3 \cdot (1+\varepsilon), 2)$-approximate solution for Problem P+RT.

Proof: The computational complexity of Algorithm P+RT is identical to that of Algorithm RT.

For a path $\mathcal{P}$ we denote by $E(\mathcal{P})$ the set of links in $\mathcal{P}$. Similarly, $E(\mathcal{B})$ denotes the set of links in bridge $\mathcal{B}$.

Let $(\hat{\mathcal{P}}, \hat{\mathcal{R}})$ be the output of Algorithm P+RT and let $(\hat{\mathcal{P}}^{opt}, \hat{\mathcal{R}}^{opt})$ be the optimal solution to Problem P+RT. We prove that there exists a restoration topology $\mathcal{R}'$ for $(\hat{\mathcal{P}}, \hat{d} + D(\hat{\mathcal{P}}))$ such that $E(\mathcal{R}') \subseteq E(\hat{\mathcal{P}}^{opt}) \cup E(\hat{\mathcal{R}}^{opt})$.

For each link $l \in \hat{\mathcal{P}}$ we identify a bridge $\mathcal{B}_l = \{s_l, \cdots, t_l\}$ such that $\mathcal{B}_l$ protects l and $E(\mathcal{B}_l) \subseteq E(\hat{\mathcal{P}}^{opt}) \cup E(\hat{\mathcal{R}}^{opt})$. We consider the following two cases.

- **Case 1.** If $l = (v_i, v_{i+1}) \in \hat{\mathcal{P}}$ and $l \notin \hat{\mathcal{P}}^{opt}$ then we choose $\mathcal{B}_l$ to be the subpath $\hat{\mathcal{P}}^{opt}_{(s_l,t_l)}$ of $\hat{\mathcal{P}}^{opt}$ for which $s_l, t_l \in \hat{\mathcal{P}}$, s_l is a predecessor of v_i in $\hat{\mathcal{P}}$ and t_l is a successor of v_{i+1} in $\hat{\mathcal{P}}$ (see Fig. 7 (a)). If more than one such subpath exists, then we choose the one with minimum hop count.
- **Case 2.** If $l = (v_i, v_{i+1}) \in \hat{\mathcal{P}}$ and $l \in \hat{\mathcal{P}}^{opt}$ then let $\mathcal{B}' = \{s', \cdots, t'\}$ be a bridge of $\hat{\mathcal{R}}^{opt}$ that protects l. We denote by $\mathcal{P}'$ the path $\hat{\mathcal{P}}^{opt}_{s,s'} \circ \mathcal{B}' \circ \hat{\mathcal{P}}^{opt}_{t,t'}$, *i.e.*, the restoration path for link l in the optimal solution. Then, we choose $\mathcal{B}_l$ to be the subpath $\mathcal{P}'_{(s_l,t_l)}$ of $\mathcal{P}'$ for which $s_l, t_l \in \hat{\mathcal{P}}$, s_l is a predecessor of v_i in $\hat{\mathcal{P}}$ and t_l is a successor of v_{i+1} in $\hat{\mathcal{P}}$ (see Fig. 7 (b)). If more than one such subpath exists, then we choose the one with minimum hop count.

Let $\mathcal{R}'$ be a restoration topology formed by bridges $\{\mathcal{B}_l | l \in \hat{\mathcal{P}}\}$. We observe that each link $l \in \mathcal{P}$ is protected by a bridge in $\mathcal{R}'$, and for each bridge $\mathcal{B}_l \in \mathcal{R}'$ it holds that $D(\mathcal{B}_l) \leq \hat{d}$, hence $\mathcal{R}'$ is a feasible restoration topology for $(\hat{\mathcal{P}}, \hat{d} + D(\hat{\mathcal{P}}))$.

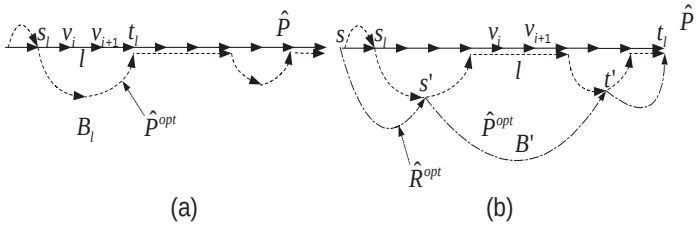

Fig. 7. The optimal solution $(\hat{\mathcal{P}}^{opt}, \hat{\mathcal{R}}^{opt})$ to Problem P+RT and an (s,t)-path $\hat{\mathcal{P}}$

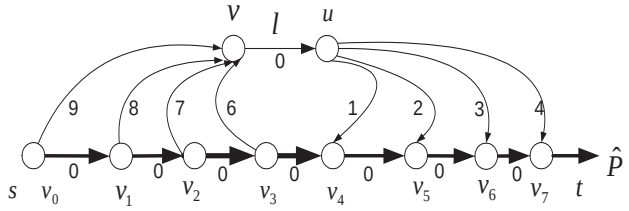

Fig. 8. An example of a restoration topology $\mathcal{R}$ formed by bridges $\mathcal{B}_1 = \{v_0, v, u, v_4\}$, $\mathcal{B}_2 = \{v_1, v, u, v_5\}$, $\mathcal{B}_3 = \{v_2, v, u, v_6\}$ and $\mathcal{B}_4 = \{v_3, v, u, v_7\}$. Link $l(v,u)$ is shared by bridges $\mathcal{B}_1, \cdots, \mathcal{B}_4$.

We also note that $C(\mathcal{R}') \leq C(\hat{\mathcal{P}}^{opt}) + C(\hat{\mathcal{R}}^{opt})$, since, $\mathcal{R}'$ contains only links from $\hat{\mathcal{P}}^{opt}$ and $\hat{\mathcal{R}}^{opt}$. By Theorem 2, the cost $C(\hat{\mathcal{R}})$ of $\hat{\mathcal{R}}$ is at most $2 \cdot (1+\varepsilon)$ times more than the cost of the optimal restoration topology for $(\hat{\mathcal{P}}, \hat{d}+D(\hat{\mathcal{P}}))$, thus $C(\hat{\mathcal{R}}) \leq 2(1+\varepsilon) \cdot C(\mathcal{R}')$. As a result, $C(\hat{\mathcal{R}}) \leq 2(1+\varepsilon)(C(\hat{\mathcal{P}}^{opt}) + C(\hat{\mathcal{R}}^{opt}))$. Since $C(\hat{\mathcal{P}}) \leq (1+\varepsilon)C(\hat{\mathcal{P}}^{opt})$, we have $C(\hat{\mathcal{P}}) + C(\hat{\mathcal{R}}) \leq 3(1+\varepsilon)(C(\hat{\mathcal{P}}^{opt}) + C(\hat{\mathcal{R}}^{opt}))$. Since $\hat{d} + D(\hat{\mathcal{P}}) \leq 2 \cdot \hat{d}$, it follows that $\hat{\mathcal{R}}$ is a $(3 \cdot (1+\varepsilon), 2)$-approximate solution for Problem P+RT. ■

VI. Directed Networks

In this section, we extend our results from the previous sections for directed networks, which we model as a directed graph $G(V,E)$. In such networks, for a pair of connected nodes (v_i, v_j), the bandwidth provisioned on the link in the direction from v_i to v_j may be much larger than the allocated bandwidth in the opposite direction. In addition, the delay and cost characteristics of link (v_i, v_j) may be very different from those of link (v_j, v_i).

We observe that, with such asymmetric links, it is possible that a node or a link is shared by several bridges, which constitutes a major obstacle for identifying efficient solutions. For example, consider the network depicted on Fig. 8. The numbers show delays of the links. For delay constraint $\hat{d} = 10$, there exists only one feasible restoration topology $\mathcal{R} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4\}$, $\mathcal{B}_1 = \{v_0, v, u, v_4\}$, $\mathcal{B}_2 = \{v_1, v, u, v_5\}$, $\mathcal{B}_3 = \{v_2, v, u, v_6\}$ and $\mathcal{B}_4 = \{v_3, v, u, v_7\}$. Note that nodes v, u and link (v,u) are shared by all bridges of $\mathcal{R}$.

We overcome this obstacle by combining bridges. Specifically, let $\mathcal{R}$ be an optimal restoration topology for $(\hat{\mathcal{P}}, \hat{d})$ and let $\{\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_k\}$, $k \geq 3$ be the set of $\mathcal{R}$'s bridges that contain the node v, sorted according to their indexes, where $\mathcal{B}_i = \{s_i, \cdots, t_i\}$. We combine the bridges $\mathcal{B}_1$ and $\mathcal{B}_k$ into a single bridge $\mathcal{B}$, such that $\mathcal{B} = \mathcal{B}_{1(s_1,v)} \circ \mathcal{B}_{k(v,t_k)}$, as depicted in Fig. 9. Since the subpaths of $\hat{\mathcal{P}}$ protected by the bridges $\mathcal{B}_1$ and $\mathcal{B}_k$ are disjoint, the delay of bridge $\mathcal{B}$ may exceed the delay of the subpath of $\hat{\mathcal{P}}$ protected by $\mathcal{B}$ by at most $2\Delta = 2(\hat{d} - D(\hat{\mathcal{P}}))$. We use this idea in order to prove the following lemma, which is the counterpart of Lemma 1 for undirected networks.

Lemma 7: Given a directed graph G, a delay constraint $\hat{d}$ and an (s,t)-path, $D(\hat{\mathcal{P}}) \leq \hat{d}$, there exists a restoration topology $\hat{\mathcal{R}}$ for $(\hat{\mathcal{P}}, 2\hat{d} - D(\hat{\mathcal{P}}))$ such that $C(\hat{\mathcal{R}}) \leq OPT$, and each node $v \in \hat{\mathcal{R}}$ or link $l \in \hat{\mathcal{R}}$ is shared by at most two bridges.

Proof: See Appendix. ■

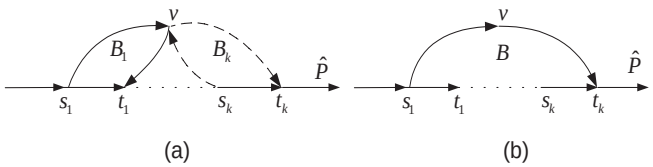

Fig. 9. (a) Node v is shared by bridges $\mathcal{B}_1$ and $\mathcal{B}_k$. (b) Bridges $\mathcal{B}_1$ and $\mathcal{B}_k$ are replaced by a single bridge $\mathcal{B}$.

Note that in the above lemma $\hat{\mathcal{R}}$ is a feasible restoration topology with respect to $\Delta = 2\hat{d} - D(\hat{\mathcal{P}})$. Thus, in order to achieve an efficient solution we relax the delay constraint and use $\Delta = 2\hat{d} - D(\hat{\mathcal{P}})$ and not $\Delta = \hat{d} - D(\hat{\mathcal{P}})$, as in the case of undirected networks. For example, the (simple) algorithm presented in Section IV-A is invoked with $\Delta = 2(\hat{d} - D(\hat{\mathcal{P}}))$. By Lemma 7, the algorithm provides a $(2 \cdot (1+\varepsilon), 2)$-approximate solution for Problem RT.

In this section we denote by $\mathcal{W}^{opt}$ the optimal walk with respect to $\Delta = 2\hat{d} - D(\hat{\mathcal{P}})$. The cost of $\mathcal{W}^{opt}$ denoted by c^{opt}.

Generally, the approximation scheme for directed networks is similar to undirected network case, except for the following:

1) The adjusted delay is defined with respect to $\Delta = 2(\hat{d} - D(\hat{\mathcal{P}}))$.
2) Algorithm PP is applied with $\Delta = 2(\hat{d} - D(\hat{\mathcal{P}}))$ (instead of $\Delta = \hat{d} - D(\hat{\mathcal{P}})$).
3) A more elaborate procedure is required for finding the lower and upper bounds L, U on c^{opt}. We provide more details below.

A. Approximation scheme for Problem RT

Recall that our approach for the undirected case was first to identify lower and upper bounds L and U on the cost of an optimum walk c^{opt} such that $U/L \leq 2N$, and then to iteratively improve the these bounds by using scaling of link costs. For directed networks, however, computing lower bound L on c^{opt} incurs high complexity. This is because any optimum walks might have high hop count(recall that in the undirected case by Lemma 1 there exists an optimum walk whose hop count is at most $2N$). Our approach is to consider, for the purpose of computing lower bound, only walks whose hop count do not exceed $2N$. More specifically, we denote by $\mathcal{W}_{2N}^{opt}$ the feasible walk of minimum cost whose hop count is at most $2N$ and by c_{2N}^{opt} the cost of $\mathcal{W}_{2N}^{opt}$. We then use the lower bound L' on c_{2N}^{opt} instead of L. The bounds L' and U are iteratively improved till either a suitable walk is found or $U/L' \leq 8$. In the latter case we apply Procedure SCALE with parameters L' and U to find a suitable walk.

Our algorithm is based on the two following lemmas.

Lemma 8: $c_{2N}^{opt} \leq 2 \cdot OPT$.

Proof: By Lemmas 7 and 2, there exists a feasible walk $\hat{\mathcal{W}}$ such that $|\hat{\mathcal{W}}| \leq 2N$ and $C(\hat{\mathcal{W}}) \leq 2 \cdot OPT$. Hence, $c_{2N}^{opt} \leq C(\hat{\mathcal{W}}) \leq 2 \cdot OPT$. ■

Lemma 9: If $c_{2N}^{opt} \leq U$ then Procedure SCALE returns a feasible walk $\mathcal{W}$ whose cost is at most $(1+\varepsilon)$ times more than c_{2N}^{opt}, *i.e.*, $C(\mathcal{W}) \leq (1+\varepsilon) \cdot c_{2N}^{opt}$.

Proof: Similar to the proof of Lemma 3, but using Lemma 8 instead of Lemma 1. ■

We proceed to describe the FPAS in more detail. First, we invoke Procedure BOUND with parameters $G(V,E), \hat{\mathcal{P}}, \hat{d}'$. It is easy to verify that the procedure returns lower and upper bounds L' and U' on c_{2N}^{opt} such that $U'/L' \leq 2N$. Next, we set $U = U'$ and use the following iterative process in order to improve improve the bounds U and L'. At each iteration we compute $B \leftarrow \sqrt{L' \cdot U}$ and apply Procedure SCALE for $L = U = B$. There are several possible cases.

- **Case 1**: Procedure SCALE returns FAIL. Then, due to Lemma 9, there exists no feasible walk $\mathcal{W}$ such $C(\mathcal{W}) \leq B$ and $|\mathcal{W}| \geq 2N$. Accordingly, we set $L' \leftarrow B$.

- **Case 2**: Procedure SCALE returns a feasible walk $\mathcal{W}$ such that $C(\mathcal{W}) \leq L'$. Then, the algorithm halts and returns the walk $\mathcal{W}$. Note that $C(\mathcal{W}) \leq c_{2N}^{opt} \leq 2 \cdot OPT$.
- **Case 3**: Procedure SCALE returns a feasible walk $\mathcal{W}$ such that $L' < C(\mathcal{W}) \leq 2B$. Since it is possible that $|\mathcal{W}| > 2N$, we set $U \leftarrow B$.

Note that by Lemma 4, if Procedure SCALE does not fail, it returns a feasible walk $\mathcal{W}$, whose cost is at most $2B$, hence all possible cases are covered. The process stops when a suitable walk is found or $U/L' \leq 8$.

Having obtaining lower bound L' on c_{2N}^{opt} and upper bound U on c^{opt} such that $U/L' \leq 8$, we apply Procedure SCALE for $L = L'$ and $U = U$. If the algorithm fails then $L' > U$, hence we return a walk $\mathcal{W}$ due to which the upper bound U was assigned its current value. Note that due to Lemma 8, $C(\mathcal{W}) \leq L' \leq c_{2N}^{opt} \leq 2 \cdot OPT$. Otherwise, due to Lemma 9, Procedure SCALE returns a walk $\mathcal{W}$ such $C(\mathcal{W}) \leq (1+\varepsilon)c_{2N}^{opt}$. In both cases we identify a feasible restoration topology $\hat{\mathcal{R}}$ that corresponds to $\mathcal{W}$. It follows that $C(\hat{\mathcal{R}}) \leq 2(1+\varepsilon)OPT$. The FPAS is implemented by Algorithm DRT, whose formal description appears on Fig. 10.

Algorithm DRT $(G(V,E), \hat{\mathcal{P}}, \hat{d}, \varepsilon)$*:*

parameters:
$G(V,E)$ - network
$\hat{\mathcal{P}} = \{s = v_1, v_2, ..., t = v_n\}$ - QoS path,
$\hat{d}$- delay constraint
ε- approximation ratio

1 $\hat{d}' \leftarrow 2\hat{d} - D(\hat{\mathcal{P}})$
2 $L', U \leftarrow$BOUND$(G(V,E), \hat{\mathcal{P}}, \hat{d}')$
3 **do**
4 $B \leftarrow \sqrt{L' \cdot U}$
5 Apply Procedure SCALE for $G(V,E), \hat{\mathcal{P}}, \hat{d}', B, B, \varepsilon$)
6 **if** Procedure SCALE return FAIL **then**
7 $L' \leftarrow B$
8 **else**
9 Set $\mathcal{W}$ be the walk returned by Procedure SCALE
10 **if** $C(\mathcal{W}) \leq L'$ **then**
11 **return** the restoration topology $\mathcal{R}$ that corresponds to $\mathcal{W}$.
12 **else**
13 $U \leftarrow 2 \cdot B$,
14 **until** $U/L' \leq 8$.
15 Apply Procedure SCALE for $(G(V,E), \hat{\mathcal{P}}, \hat{d}', L', U, \varepsilon)$
16 **if** Procedure SCALE does not fail **then**
17 Set $\mathcal{W}$ be the walk returned by Procedure SCALE
18 **return** the restoration topology $\mathcal{R}$ that corresponds to $\mathcal{W}$.

Fig. 10. Algorithm DRT

We summarize our results in the following theorem.

Theorem 4: Given are a directed graph G, a primary QoS path $\hat{\mathcal{P}} \in G$, a delay constraint $\hat{d}$ and an approximation ratio ε. Then, Algorithm DRT identifies, in $O(MN(1/\varepsilon + \log N))$ a feasible restoration topology $\mathcal{R}$ for $(\hat{\mathcal{P}}, 2\hat{d} - D(\hat{\mathcal{P}}))$, whose cost is at most $2(1+\varepsilon)$ times more than OPT, *i.e.*, a $(2(1+\varepsilon), 2)$-approximate solution to Problem RT.

B. Approximation scheme for Problem P+RT

The approximation scheme for of primary QoS path and the restoration topology is similar to the undirected case. Namely, we first identify a $\hat{d}$-delay constrained (s,t)- path $\hat{\mathcal{P}}$ in G whose cost is at most $(1+\varepsilon)$ times more than the optimum. Then, we apply Algorithm DRT with parameters G, $\hat{\mathcal{P}}$, $(\hat{d} + D(\hat{\mathcal{P}}))$ and ε.

Theorem 5: The above algorithm identifies, in $O(MN(1/\varepsilon + \log N))$ time, a $(3(1+\varepsilon), 3)$-approximate solution for Problem P+RT in directed graphs.

Proof: The proof follows similar lines as in Theorem 3. Let $(\hat{\mathcal{P}}, \hat{\mathcal{R}})$ be the output of Algorithm P+RT and let $(\hat{\mathcal{P}}^{opt}, \hat{\mathcal{R}}^{opt})$ be the optimal solution to Problem P+RT. By Theorem 4, Algorithm DRT returns a restoration topology $\hat{\mathcal{R}}$ for $(\hat{\mathcal{P}}, 2\hat{d} + D(\hat{\mathcal{P}}))$ such that $C(\hat{\mathcal{R}}) \leq 2(1+\varepsilon) \cdot C(\hat{\mathcal{R}}^{opt})$. Since $2\hat{d} + D(\hat{\mathcal{P}}) \leq 3 \cdot \hat{d}$, $(\hat{\mathcal{P}}, \hat{\mathcal{R}})$ is a $(3(1+\varepsilon), 3)$-approximate solution for Problem P+RT.

The running time of the algorithm is dominated by Algorithm DRT, hence it is $O(MN(1/\varepsilon + \log N))$. ■

VII. Simulation Results

In order to further illustrate the efficiency of our proposed solutions, we conducted simulation experiments. We compared the following two algorithms for undirected networks (with respect to the total cost of the computed primary and restoration paths):

• **Algorithm 1** - Two Disjoint Paths. Provision a $\hat{d}$-delay constrained path $\mathcal{P}_1$, which serves as the primary path, then delete all its links from the graph, and finally, provision a $\hat{d}$-delay constrained path $\mathcal{P}_2$ in the resulting graph, which serves as the restoration path. Both paths are provisioned using Algorithm RSP of [12].

• **Algorithm 2** - Algorithm P+RT with $\beta = 1$. Provision the primary path $\mathcal{P}_1$ satisfying $\hat{d}$ (using Algorithm RSP of [12]), then use Algorithm RT to provision the restoration topology $\mathcal{R}$ for $(\mathcal{P}_1, \hat{d})$. Note that the delay constraint is strictly satisfied for both the primary and restoration paths.

Recall that Algorithm RSP returns a $(1+\varepsilon)$-approximation to the minimum cost path satisfying delay constraints. In both algorithms, we choose ε to be a fairly small constant.

A. Network Generation Models

We used two different methods for generating the undirected network topologies using the BRITE topology generation tool [14]. The first is Waxman's method [18], and the second is Barabasi and Albert's [1], described below. Both network generators assign delays to links based on the distance between the link's endpoints. Further, we assigned costs to links uniformly and randomly from a fixed interval.

• **Waxman model [18].** In this model, nodes are placed on a plane; the probability of interconnecting two nodes decreases exponentially with the Euclidean distance between them. We set the value for parameters α and β to 0.15 and 0.2, respectively.

• **Barabasi-Albert model [1].** In this model, the node connectivity follows a power-law rule: very few nodes have high connectivity, and the number of nodes with lower connectivity increases exponentially as the connectivity decreases.

B. Experimental Results

In our experiments, we compared the provisioning costs (*i.e.*, the total cost of the primary path and the restoration topology) computed by the two algorithms for a randomly selected source and destination node pair. We studied the effect of varying the delay constraint. More specifically, let the delay ratio be $x = \hat{d}/d_{sp}$, where d_{sp} is the minimum delay of an (s,t)-path. Fig. 11 depicts the provisioning costs for Algorithms 1 and 2 for the (a) Waxman and (b) Barabasi-Albert models in a network of 7000 nodes, and as the delay ratio is increased from 1.2 to 1.6.

The findings of our study can be summarized as follows:

• In many cases, for which Algorithm 1 fails to find a pair of disjoint paths, Algorithm 2 still computes a (feasible) primary path and restoration topology solution with a low cost. For example, for the Barabasi-Albert model, for values $x = 1.5$ and $x = 1.6$ Algorithm 1 fails, while Algorithm 2 still provides a feasible low-cost solution. The same occurs for $x = 1.2$ in the Waxman model. This clearly demonstrates the superiority of the restoration topology strategy over the disjoint path approach.

• Algorithm P+RT (even with $\beta = 1$!) always exhibits superior performance (*i.e.*, finds paths with lower cost) compared to Algorithm 1.

• The cost benefits due to Algorithm 2 are particularly significant (around 15-20%) when delay constraints are tight, *i.e.*, closer to the minimum delay of an (s,t)-path.

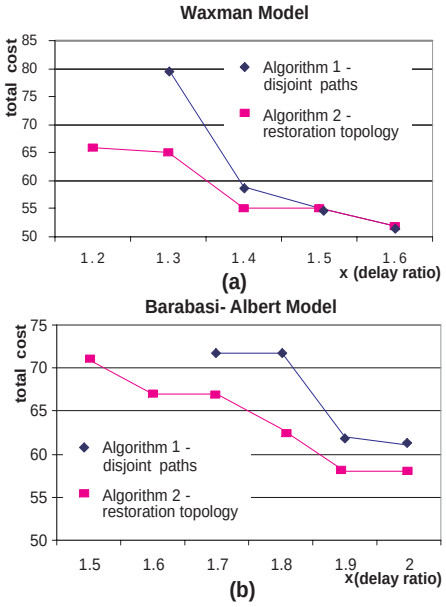

Fig. 11. Effect of delay ratio on the algorithm performance

VIII. Conclusion

In this paper, we investigated the problem of provisioning QoS paths with restoration. Specifically, we developed algorithms that compute a *primary QoS path* and a *restoration topology* comprising of a set of *bridges*, each of which protects a different part of the primary QoS path.

A major contribution of this paper is the concept of *adjusted delays*, which allows existing path algorithms (*e.g.*, Bellman-Ford [2], Hassin's [6]) to be adapted in order to identify suitable restoration topologies. This enabled us to devise efficient approximation schemes with proven performance guarantees. Specifically, we presented an $O(MN(1/\varepsilon + \log N))$ approximation scheme (Algorithm P+RT) that provides $(3 \cdot (1+\varepsilon), 2)$-approximate solutions for link failures. We extended the scheme for directed networks and achieved a $(3 \cdot (1+\varepsilon), 3)$-approximate solution. We emphasize that, in our schemes, the delay violation may occur only in the restoration paths, while *the primary path always satisfies the QoS constraint*.

The simulation results indicate that in many cases, for which the disjoint path strategy fails, Algorithm P+RT still computes a (feasible) primary path and a low-cost restoration topology. This clearly demonstrates the superiority of the restoration topology strategy over the disjoint path approach. The cost benefits due to Algorithm P+RT are particularly significant (around 15-20%) when delay constraints are tight.

References

[1] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
[3] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet – RFC No. 2386. Internet RFC, August 1998.
[4] L. Zhang F. Ergun, R. Sinha. An Improved FPTAS for Restricted Shortest Path. In *Technical Report*, Bell Labs, 2001.
[5] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1979.
[6] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, February 1992.
[7] G. Italiano, R. Rastogi, and B. Yener. Restoration Algorithms for Virtual Private Networks in the Hose Model. In *Proceedings of IEEE INFOCOM'02*, New York, NY, June 2002.
[8] K. Kar, M. Kodialam, and T. V. Lakshman. Routing restorable bandwidth guaranteed connections using maximum 2-route flows. In *Proceedings of IEEE INFOCOM'02*, New York, NY, June 2002.
[9] Murali S. Kodialam and T. V. Lakshman. Dynamic routing of bandwidth guaranteed tunnels with restoration. In *Proceedings of IEEE INFOCOM'2000*, Tel-Aviv, Israel, March 2000.
[10] G. Krishna, M. Pradeep, and C. Ram Murthy. A segmented backup scheme for dependable real time communication in multihop networks. In *Proceedings of Workshop on Parallel and Distributed Real-Time Systems (WPDRTS'2000)*, Cancun, Mexico, May 2000.

[11] G. Li, D. Wang, C. Kalmanek, and R. Doverspike. Efficient distributed path selection for shared restoration connections. In *Proceedings of IEEE INFOCOM'02*, New York, NY, June 2002.
[12] D.H. Lorenz and D. Raz. A Simple Efficient Approximation Scheme for the Restricted Shortest Path Problem. *Operations Research Letters*, 28(5):213–219, June 2001.
[13] Q. Ma and P. Steenkiste. Quality of Service Routing for Traffic with Performance Guarantees. In *Proceedings of International Workshop on Quality of Service (IWQoS'97)*, Columbia University, New York, NY, May 1997.
[14] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: Universal topology generation from a user's perspective. In *Proceedings of Workshop the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '01)*, Cincinnati, OH, August 2001.
[15] A. Orda. Routing with end to end QoS guarantees in broadband networks. *IEEE/ACM Transactions on Networking*, 7(3):365–374, June 1999.
[16] J. Sobrinho. Algebra and Algorithms for Qos Path Computation and Hop-by-Hop Routing in the Internet. In *Proceedings of IEEE INFOCOM'01*, Anchorage, Alaska, April 2001.
[17] J. Suurballe. Disjoint Path in Networks. *Networks*, 4:125–145, 1974.
[18] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1671–1622, 1988.

APPENDIX

Lemma 2:

1) Let $\mathcal{R} = \{\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_h\}$, $\mathcal{B}_i = \{s_i, \cdots, t_i\}$, be a feasible restoration topology for $(\hat{\mathcal{P}}, \hat{d})$ and let $\mathcal{W}$ be an (s,t)-walk in G' that corresponds to $\mathcal{R}$, *i.e.*, $\mathcal{W} = \{\mathcal{B}_1 \circ \hat{\mathcal{P}}_{(t_1,s_2)} \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_h\}$. Then, $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$.
2) Let $\mathcal{W}$ be an (s,t)-walk in G' such that $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$. Then, it is possible to decompose $\mathcal{W}$ into a set of bridges $\{\mathcal{B}_1, \cdots, \mathcal{B}_h\}$ that forms a feasible restoration topology.

Proof: (1) We first prove that if $\tilde{D}(\mathcal{W}_{(s,s_i)}) \leq D(\hat{\mathcal{P}}_{(s,s_i)})$ for $i \leq h$ then $\tilde{D}(\mathcal{W}_{(s,t_i)}) \leq D(\hat{\mathcal{P}}_{(s,t_i)})$. Since no link in bridge $\mathcal{B}_i$ belongs to $\hat{\mathcal{P}}'$, for each link $(u_{j-1}, u_j) \in \mathcal{B}_i$ it holds that $\tilde{D}(\mathcal{W}_{(s,u_j)}) \leq \tilde{D}(\mathcal{W}_{(s,u_{j-1})}) + d_{(u_{j-1},u_j)}$. As a result, for the immediate predecessor u^* of t_i in $\mathcal{B}_i$ it follows that $\tilde{D}(\mathcal{W}_{(s,u^*)}) \leq \tilde{D}(\mathcal{W}_{(s,s_i)}) + D(\mathcal{W}_{(s_i,u^*)})$. The feasibility of $\mathcal{R}$ implies that $D(\mathcal{W}_{(s_i,t_i)}) \leq D(\hat{\mathcal{P}}_{(s_i,t_i)}) + \Delta$. Thus, $\tilde{D}(\mathcal{W}_{(s,u^*)}) + d_{(u^*,t_i)} \leq \tilde{D}(\mathcal{W}_{(s,s_i)}) + D(\mathcal{W}_{(s_i,t_i)}) \leq D(\hat{\mathcal{P}}_{(s,t_i)}) + \Delta$. Hence, according to the definition of adjusted delay, $\tilde{D}(\mathcal{W}_{(s,t_i)}) \leq D(\hat{\mathcal{P}}_{(s,t_i)})$.

Next, we show that if $\tilde{D}(\mathcal{W}_{(s,t_i)}) \leq D(\hat{\mathcal{P}}_{(s,t_i)})$ for $i \leq h$ then $\tilde{D}(\mathcal{W}_{(s,s_{i+1})}) \leq D(\hat{\mathcal{P}}_{(s,s_{i+1})})$. Note that each link (u_j, u_{j+1}) of the subwalk $\mathcal{W}_{(t_i,s_{i+1})}$ of $\mathcal{W}$ belongs to $\hat{\mathcal{P}}'$. Thus, since $\tilde{D}(\mathcal{W}_{(s,t_i)}) \leq D(\hat{\mathcal{P}}_{(s,t_i)})$, for each node $u \in \mathcal{W}_{(t_i,s_{i+1})}$ it holds that $\tilde{D}(\mathcal{W}_{(s,u)}) \leq D(\hat{\mathcal{P}}_{(s,u)})$. We conclude that $\tilde{D}(\mathcal{W}_{(s,s_{i+1})}) \leq D(\hat{\mathcal{P}}_{(s,s_{i+1})})$.

Next, we show, by induction on i, that $\tilde{D}(\mathcal{W}_{(s,s_i)}) \leq D(\hat{\mathcal{P}}_{(s,s_i)})$ for $i \leq h$. Clearly, $\tilde{D}(\mathcal{W}_{(s,s_1)}) = 0$ forms the base case. If $\tilde{D}(\mathcal{W}_{(s,s_i)}) \leq D(\hat{\mathcal{P}}_{(s,s_i)})$ for some $i < h$, then, as proven above, $\tilde{D}(\mathcal{W}_{(s,t_i)}) \leq D(\hat{\mathcal{P}}_{(s,t_i)})$, which in turn implies that $\tilde{D}(\mathcal{W}_{(s,s_{i+1})}) \leq D(\hat{\mathcal{P}}_{(s,s_{i+1})})$.

We proved that $\tilde{D}(\mathcal{W}_{(s,s_h)}) \leq D(\hat{\mathcal{P}}_{(s,s_h)})$, hence $\tilde{D}(\mathcal{W}_{(s,t_h)}) \leq D(\hat{\mathcal{P}}_{(s,t_h)})$, which completes the proof of part (1).

(2) We enumerate the nodes of $\mathcal{W}$ by u_1, u_2, *etc*, *i.e.*, $\mathcal{W} = \{u_1, \cdots, u_n\}$.

Let S_1, S_2, S_3 be subsets of $\mathcal{W}$'s nodes, as follows:

$$S_1 = \{u_i \in \mathcal{W} | (u_{i-1}, u_i) \in \hat{\mathcal{P}}', (u_i, u_{i+1}) \notin \hat{\mathcal{P}}'\},$$
$$S_2 = \{u_i \in \mathcal{W} | (u_{i-1}, u_i) \notin \hat{\mathcal{P}}', (u_i, u_{i+1}) \in \hat{\mathcal{P}}'\},$$
$$S_3 = \{u_i \in \mathcal{W} | u_i \in \hat{\mathcal{P}}', (u_{i-1}, u_i) \notin \hat{\mathcal{P}}', (u_i, u_{i+1}) \notin \hat{\mathcal{P}}', \tilde{D}(\mathcal{W}_{(s,u_i)}) = D(\hat{\mathcal{P}}_{(s,u_i)})\}.$$

The restoration topology $\mathcal{R}$ is then formed by taking the portions of the walk. More specifically, we decompose $\mathcal{W}$ into a set of bridges $\mathcal{R}$, each bridge $\mathcal{B}_i = \{s_i, \cdots, t_i\} \in \mathcal{R}$ is a subwalk of $\mathcal{W}$ such that:

1) $s_i \in \{s\} \cup S_1 \cup S_3$;
2) $t_i \in \{t\} \cup S_2 \cup S_3$;
3) $\mathcal{B}_i$ does not contain nodes that belong to S_1, S_2 or S_3, except for s_i and t_i.

We prove that $\mathcal{R}$ is a feasible restoration topology. First, we show that for each $u_i \in S_1$ it holds that $\tilde{D}(\mathcal{W}_{(s,u_i)}) = D(\hat{\mathcal{P}}_{(s,u_i)})$. Indeed, according to the definition of adjusted delay, for each $u_i \in S_1$, $\tilde{D}(\mathcal{W}_{(s,u_i)})$

can be either $D(\hat{\mathcal{P}}_{(s,u_i)})$ or ∞. If $\tilde{D}(\mathcal{W}_{(s,u_i)})$ is set to ∞, it stays at ∞ for any extension of $\mathcal{W}_{(s,u_i)}$, which would lead to a contradiction. Similarly, for each $u_i \in S_2$ we have $\tilde{D}(\mathcal{W}_{(s,u_i)}) \leq D(\hat{\mathcal{P}}_{(s,u_i)})$, otherwise, according to the definition, the adjusted delay of the successor $s(u_i)$ of u_i in $\mathcal{W}$ would be set to ∞, resulting in a contradiction.

We proceed to show that the delay of each bridge $\mathcal{B}_i = \{s_i, ..., t_i\} \in \mathcal{R}$ exceeds the delay of the corresponding subpath $\hat{\mathcal{P}}_{(s_i,t_i)}$ of $\hat{\mathcal{P}}$ by at most Δ. Since $\mathcal{B}_i$ does not contain nodes that belong to S_1, S_2 or S_3, except for s_i and t_i, it follows that, for each link $l = (u_i, u_{i+1}) \in \mathcal{B}_i$, except for the last one, it holds that $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) = \tilde{D}(\mathcal{W}_{(s,u_i)}) + d_{(u_{i-1},u_i)}$. Clearly, this condition holds if $u_{i+1} \notin \hat{\mathcal{P}}$. If $u_{i+1} \in \hat{\mathcal{P}}$, then either $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) < D(\hat{\mathcal{P}}_{(s,u_{i+1})})$ or $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) > D(\hat{\mathcal{P}}_{(s,u_{i+1})})$, otherwise u_{i+1} would belong to S_3. Hence, according to the definition $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) = \tilde{D}(\mathcal{W}_{(s,u_i)}) + d_{(u_{i-1},u_i)}$. Let u^* be the immediate predecessor of t_i. It follows that $\tilde{D}(\mathcal{W}_{(s,u^*)}) = \tilde{D}(\mathcal{W}_{(s,s_i)}) + D(\mathcal{W}_{(s_i,u^*)})$. Since $s_i \in \{s\} \cup S_1 \cup S_3$ it holds that $\tilde{D}(\mathcal{W}_{(s,s_i)}) = D(\hat{\mathcal{P}}_{(s,s_i)})$, hence $\tilde{D}(\mathcal{W}_{(s,u^*)}) = D(\hat{\mathcal{P}}_{(s,s_i)}) + D(\mathcal{W}_{(s_i,u^*)})$. Since $t_i \in \{t\} \cup S_2 \cup S_3$, it holds that $\tilde{D}(\mathcal{W}_{(s,t_i)}) \leq D(\hat{\mathcal{P}}_{(s,t_i)})$, hence $\tilde{D}(\mathcal{W}_{(s,u^*)}) + d_{(u_{i-1},u_i)} \leq D(\hat{\mathcal{P}}_{(s,t_i)}) + \Delta$. Substituting $\tilde{D}(\mathcal{W}_{(s,u^*)})$, yields $D(\hat{\mathcal{P}}_{(s,s_i)}) + D(\mathcal{W}_{(s_i,t_i)}) \leq D(\hat{\mathcal{P}}_{(s,t_i)}) + \Delta$, or $D(\mathcal{W}_{(s_i,t_i)}) \leq D(\hat{\mathcal{P}}_{(s_i,t_i)}) + \Delta$. We conclude that the restoration topology is ***feasible, i.e.,*** delay of each bridge exceeds the delay of the corresponding subpath of $\hat{\mathcal{P}}$ by at most Δ.

Finally, we show that each link $l = (v_i, v_{i+1}) \in \hat{\mathcal{P}}$ is protected by a bridge in $\mathcal{R}$. Let v_j be the last node before v_i or v_i itself that belongs to $\{s\} \cup S_1 \cup S_3$. Also, let v_k be the first node after v_{i+1} or v_{i+1} itself that belongs to $\{t\} \cup S_2 \cup S_3$. Then $\mathcal{B}_i = \{v_j, ..., v_k\} \in \mathcal{R}$ is a bridge that protects l. ■

Theorem 1: Let c^{opt} denote the minimum cost of a feasible (s,t)-walk in G'. Then, the following two conditions hold:

1) Any walk $\mathcal{W}$ returned by Algorithm PP is feasible.
2) If $c^{opt} \leq U$ then Algorithm PP returns a minimum cost feasible walk $\mathcal{W}$.

Proof: (1) We prove by induction of i that, for $i = 0, ..., k$, $\tilde{D}(\mathcal{W}_{(s,u_i)}) = D_{u_i}[c_i]$, where $c_0 = 0$ and $c_i = c_{i-1} + c_{(u_{i-1},u_i)}$. Since $D_t[c_k] \leq D(\hat{\mathcal{P}})$, this implies that $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$.

For $i = 0$, we have $\tilde{D}(\mathcal{W}_{(s,s)}) = 0$, which forms the base step. Suppose that $\tilde{D}(\mathcal{W}_{(s,u_i)}) = D_{u_i}[c_i]$ for $i > 0$. We show that $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) = D_{u_{i+1}}[c_{i+1}]$. We consider the following three cases (recall that $\hat{\mathcal{P}}' \in G'$ is the path formed from $\hat{\mathcal{P}}$ by reversing its links).

- $(u_i, u_{i+1}) \in \hat{\mathcal{P}}'$: in this case, $c_{i+1} = c_i$. Also, since (u_i, u_{i+1}) results in the value for $D_{u_{i+1}}[c_{i+1}]$, it must be the case that $D_{u_i}[c_i] \leq D(\hat{\mathcal{P}}_{(s,u_i)})$ and $D_{u_{i+1}}[c_{i+1}]$ is set to $D(\hat{\mathcal{P}}_{(s,u_{i+1})})$. Clearly, since $\tilde{D}(\mathcal{W}_{(s,u_i)}) = D_{u_i}[c_i]$, it follows that $\tilde{D}(\mathcal{W}_{(s,u_i)}) \leq D(\hat{\mathcal{P}}_{(s,u_i)})$ and so by the definition of adjusted delay, $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) = D(\hat{\mathcal{P}}_{(s,u_{i+1})})$. Thus, $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) = D_{u_{i+1}}[c_{i+1}]$.
- $(u_i, u_{i+1}) \notin \hat{\mathcal{P}}'$, $D_{u_{i+1}}[c_{i+1}] = D_{u_i}[c_i] + d_{(u_i,u_{i+1})}$: in this case, either $u_{i+1} \notin \hat{\mathcal{P}}$, or $u_{i+1} \in \hat{\mathcal{P}}$ and one of the following holds: $D_{u_i}[c_i] + d_{(u_i,u_{i+1})} > D(\hat{\mathcal{P}}_{(s,u_{i+1})}) + \Delta$ or $D_{u_i}[c_i] + d_{(u_i,u_{i+1})} \leq D(\hat{\mathcal{P}}_{(s,u_{i+1})})$. In all of the cases, $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) = \tilde{D}(\mathcal{W}_{(s,u_i)}) + d_{(u_i,u_{i+1})}$, and thus $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) = D_{u_{i+1}}[c_{i+1}]$ (since $\tilde{D}(\mathcal{W}_{(s,u_i)}) = D_{u_i}[c_i]$).
- $(u_i, u_{i+1}) \notin \hat{\mathcal{P}}'$, $D_{u_{i+1}}[c_{i+1}] < D_{u_i}[c_i] + d_{(u_i,u_{i+1})}$: in this case $u_{i+1} \in \hat{\mathcal{P}}$, $D(\hat{\mathcal{P}}_{(s,u_{i+1})}) < D_{u_i}[c_i] + d_{(u_i,u_{i+1})} \leq D(\hat{\mathcal{P}}_{(s,u_{i+1})}) + \Delta$. Since $\tilde{D}(\mathcal{W}_{(s,u_i)}) = D_{u_i}[c_i]$, the definition of adjusted delay implies that $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) = D(\hat{\mathcal{P}}_{(s,u_{i+1})})$ and since $D_{u_{i+1}}[c_{i+1}]$ is also set to $D(\hat{\mathcal{P}}_{(s,u_{i+1})})$, it holds that $\tilde{D}(\mathcal{W}_{(s,u_{i+1})}) = D_{u_{i+1}}[c_{i+1}]$.

(2) Let $\hat{\mathcal{W}} = \{s = u_0, \cdots, u_k = t\}$ be a feasible (s,t)-walk of minimum cost, *i.e.,* $C(\hat{\mathcal{W}}) = c^{opt}$. We show by induction on i that, for $i = 0, ..., k$, $D_{u_i}[c_i] \leq \tilde{D}(\hat{\mathcal{W}}_{(s,u_i)})$, where $c_0 = 0$ and $c_i = c_{i-1} + c_{(u_{i-1},u_i)}$. Since $D_t[c_k] \leq D(\hat{\mathcal{W}}_{(s,t)}) \leq D(\hat{\mathcal{P}})$, this implies that the algorithm breaks out of the loop (in Step 16) at $c \leq c_k$. Hence, the algorithm returns a walk $\mathcal{W}$ whose cost is most c_k, *i.e.,* $C(\mathcal{W}) \leq c_k = C(\hat{\mathcal{W}})$.

The basis is trivially true since $D_s[0] = 0$. Assume that for $i \geq 0$, it holds that $D_{u_i}[c_i] \leq \tilde{D}(\hat{\mathcal{W}}_{(s,u_i)})$.

We show that for $i+1$ it holds that $D_{u_{i+1}}[c_{i+1}] \leq \tilde{D}(\hat{\mathcal{W}}_{(s,u_{i+1})})$. We need to consider the following three cases:

- $(u_i, u_{i+1}) \in \hat{\mathcal{P}}'$: if $\tilde{D}(\hat{\mathcal{W}}_{(s,u_i)}) \leq D(\hat{\mathcal{P}}_{(s,u_i)})$, then $\tilde{D}(\hat{\mathcal{W}}_{(s,u_{i+1})}) = D(\hat{\mathcal{P}}_{(s,u_{i+1})})$ and also $D_{u_{i+1}}[c_{i+1}] \leq D(\hat{\mathcal{P}}_{(s,u_{i+1})})$ (since $D_{u_i}[c_i] \leq \tilde{D}(\hat{\mathcal{W}}_{(s,u_i)})$ and u_i is processed before u_{i+1} for cost c_i). Thus, $D_{u_{i+1}}[c_{i+1}] \leq \tilde{D}(\hat{\mathcal{W}}_{(s,u_{i+1})})$. If $\tilde{D}(\hat{\mathcal{W}}_{(s,u_i)}) > D(\hat{\mathcal{P}}_{(s,u_i)})$, then $\tilde{D}(\hat{\mathcal{W}}_{(s,u_{i+1})}) = \infty$, hence $D_{u_{i+1}}[c_{i+1}] \leq \tilde{D}(\hat{\mathcal{W}}_{(s,u_{i+1})})$.
- $(u_i, u_{i+1}) \notin \hat{\mathcal{P}}'$, $u_{i+1} \notin \hat{\mathcal{P}}$ or $u_{i+1} \in \hat{\mathcal{P}}$ and one of the following holds: $\tilde{D}(\hat{\mathcal{W}}_{(s,u_i)}) + d_{(u_1,u_{i+1})} \leq D(\hat{\mathcal{P}}_{(s,u_i)})$ or $\tilde{D}(\hat{\mathcal{W}}_{(s,u_i)}) + d_{(u_1,u_{i+1})} > D(\hat{\mathcal{P}}_{(s,u_i)}) + \Delta$. In this case $\tilde{D}(\hat{\mathcal{W}}_{(s,u_{i+1})}) = \tilde{D}(\hat{\mathcal{W}}_{(s,u_i)}) + d_{(u_1,u_{i+1})}$. Thus, since $D_{u_i}[c_i] \leq \tilde{D}(\hat{\mathcal{W}}_{(s,u_i)})$, it follows that $D_{u_{i+1}}[c_{i+1}] \leq \tilde{D}(\hat{\mathcal{W}}_{(s,u_i)}) + d_{(u_1,u_{i+1})} = \tilde{D}(\hat{\mathcal{W}}_{(s,u_{i+1})})$.
- $(u_i, u_{i+1}) \notin \hat{\mathcal{P}}'$, $u_{i+1} \in \hat{\mathcal{P}}$ and $D(\hat{\mathcal{P}}_{(s,u_i)}) < \tilde{D}(\hat{\mathcal{W}}_{(s,u_i)}) + d_{(u_i,u_{i+1})} \leq D(\hat{\mathcal{P}}_{(s,u_i)}) + \Delta$. In this case $\tilde{D}(\hat{\mathcal{W}}_{(s,u_{i+1})}) = D(\hat{\mathcal{P}}_{(s,u_{i+1})})$. Since $D_{u_i}[c_i] \leq \tilde{D}(\hat{\mathcal{W}}_{(s,u_i)})$, it follows that $D_{u_i}[c_i] + d_{(u_i,u_{i+1})} \leq D(\hat{\mathcal{P}}_{(s,u_i)}) + \Delta$, and $D_{u_{i+1}}[c_{i+1}]$ is set to be $D(\hat{\mathcal{P}}_{(s,u_{i+1})})$. As a result, it follows that $D_{u_{i+1}}[c_{i+1}] \leq \tilde{D}(\hat{\mathcal{W}}_{(s,u_{i+1})})$.

We conclude that if $c^{opt} \leq U$ then the algorithm returns a feasible walk $\mathcal{W}$ in G' for which $C(\mathcal{W}) \leq c^{opt}$ and the theorem follows.

■

Lemma 7: Given a directed graph G, a delay constraint $\hat{d}$ and an (s,t)-path, $D(\hat{\mathcal{P}}) \leq \hat{d}$, there exists a restoration topology $\hat{\mathcal{R}}$ for $(\hat{\mathcal{P}}, 2\hat{d} - D(\hat{\mathcal{P}}))$ such that $C(\hat{\mathcal{R}}) \leq OPT$, and each node $v \in \hat{\mathcal{R}}$ or link $l \in \hat{\mathcal{R}}$ is shared by at most two bridges.

Proof: First, we introduce several definitions. Let v and u be two nodes of $\hat{\mathcal{P}} = \{s, \cdots, t\}$, such that $v \neq u$ and v is a predecessor of u. We say that u is *more right* on $\hat{\mathcal{P}}$ than v, and that v is *more left* on $\hat{\mathcal{P}}$ than u.

Let $S = \{\mathcal{B}_1, \cdots, \mathcal{B}_k\}$ be a set of bridges, we denote by $V(S)$ the set of nodes that belong to bridges $\mathcal{B}_i$, *i.e.*, $V(S) = \{v | v \in \mathcal{B}_i, \mathcal{B}_i \in S\}$. For a node $v \in V(S)$, we denote by $\mathcal{B}_v$ a bridge in S such that $v \in \mathcal{B}_v$ and the source node s_v of $\mathcal{B}_v$ is most left on $\hat{\mathcal{P}}$. Similarly, we denote by $\mathcal{B}'_v$ a bridge in S such that $v \in \mathcal{B}'_v$ and the destination node t'_v of $\mathcal{B}'_v$ is most right on $\hat{\mathcal{P}}$. The *hyper-bridge* $\hat{\mathcal{B}}_v$ for (S, v) is a concatenation of paths $\mathcal{B}_{v(s_v,v)}$ and $\mathcal{B}'_{v(v,t'_v)}$. Note that if $\mathcal{B}_v = \mathcal{B}'_v$ then $\hat{\mathcal{B}}_v = \mathcal{B}_v$.

We show that the delay of each hyper-bridge $\hat{\mathcal{B}}_v$ exceeds the delay of its protected segment by at most $2\hat{d} - D(\hat{\mathcal{P}})$. Suppose that hyper-bridge $\hat{\mathcal{B}}_v$ was formed by combining bridges $\mathcal{B}_v$ and $\mathcal{B}'_v$. First, we note that $D(\hat{\mathcal{B}}_v) \leq D(\mathcal{B}_v) + D(\mathcal{B}'_v)$. Then, we observe that the the protected segment of $\hat{\mathcal{B}}_v$ is a union of protected segments of $\mathcal{B}_v$ and $\mathcal{B}'_v$. We also note that the delays of bridges $\mathcal{B}_v$ and $\mathcal{B}_v$ exceed the delays of their protected segments by at most Δ. Next, we observe that protected segments of $\mathcal{B}_v$ and $\mathcal{B}'_v$ may overlap, but the delay of their intersection is at most $d(\hat{\mathcal{P}})$. From the above observations it follows that the delay of hyper-bridge $\hat{\mathcal{B}}_v$ exceeds the delay of its protected segment by at most $2\Delta + d(\hat{\mathcal{P}})$.

Let $\mathcal{R} = \{\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_h\}$ be an optimal restoration topology for $(\hat{\mathcal{P}}, \hat{d})$, *i.e.*, $C(\mathcal{R}) = OPT$. We prove the lemma by constructing a restoration topology $\hat{\mathcal{R}}$ for $(\hat{\mathcal{P}}, 2\hat{d})$, such that $C(\hat{\mathcal{R}}) \leq C(\mathcal{R})$ and no node is included in more than two hyper-bridges of $\hat{\mathcal{R}}$.

The restoration topology $\hat{\mathcal{R}}$ is constructed in an iterative fashion. We maintain set S of bridges and set $\hat{S}$ of hyper-bridges. Initially, S includes all bridges of $\mathcal{R}$ and $\hat{S}$ is empty. At iteration i, $i = 1, 2, \cdots$, we first identify, for each node $v \in V(S)$, the hyper-bridge $\hat{\mathcal{B}}_v = \{s_v, \cdots, t_v\}$ for (S, v). Next, we choose a hyper-bridge $\hat{\mathcal{B}}_i = \{s_i, \cdots, t_i\} \in \{\hat{\mathcal{B}}_v | v \in V(S)\}$ such that s_i is no more right on $\hat{\mathcal{P}}$ than t_{i-1} (s if $i = 1$) and t_i is most right on $\hat{\mathcal{P}}$. Next, we add $\hat{\mathcal{B}}_i$ to $\hat{S}$ and delete from S all bridges whose protected segment is covered by $\hat{\mathcal{P}}_{(s,t_{v^i})}$. If S in nonempty, we proceed to the next iteration. Finally, the restoration topology $\hat{\mathcal{R}}$ is formed from hyper-bridges in $\hat{S}$.

Since the protected segment of each bridge $\mathcal{B}_i$ deleted from S is covered by the protected segments of hyper-bridges in $\hat{S}$, each link $l \in \hat{\mathcal{P}}$ belongs to a protected segment of a hyper-bridge in $\hat{S}$. In addition, as

showed above, the delay of each hyper-bridge $\hat{\mathcal{B}}_i \in \hat{S}$ exceeds the delay of its protected segment $\{s_v, \cdots, t_v\}$ by at most $2\hat{d} - D(\hat{\mathcal{P}})$. Thus, and since $E(\hat{\mathcal{R}}) \subseteq E(\mathcal{R})$, it follows that $\hat{\mathcal{R}}$ is a feasible restoration topology for $(\hat{\mathcal{P}}, 2\hat{d})$ and $C(\hat{\mathcal{R}}) \leq C(\mathcal{R})$.

Let $u \in V(\hat{S})$, we prove that u is included in at most two hyper-bridges of $\hat{S}$. Let i be the first iteration at which a hyper-bridge that includes u was added to $\hat{S}$. We denote by $\hat{\mathcal{B}}_i = \{s_i, \cdots, t_i\}$ and $\hat{\mathcal{B}}_{i+1} = \{s_{i+1}, \cdots, t_{i+1}\}$ the hyper-bridges that were inserted to $\hat{S}$ at iterations i and $i+1$, respectively. We now consider set S at iteration $i+1$. If $u \notin V(S)$, then each hyper-bridge added to $\hat{S}$ after iteration i does not contain u. Otherwise we denote by $\hat{\mathcal{B}}_u = \{s_u, \cdots, t_u\}$ the hyper-bridge for (S, u). Since $u \in \hat{\mathcal{B}}_i$, s_u is no more right on $\hat{\mathcal{P}}$ than t_i. Further, hyper-bridge $\hat{\mathcal{B}}_{i+1}$ was selected in iteration $i+1$ such that t_{i+1} is most right on $\hat{\mathcal{P}}$, hence t_u is no more right on $\hat{\mathcal{P}}$ than t_{i+1}. If follows that the protected segment of any bridge that contains u is covered by $\hat{\mathcal{P}}_{(s,t_{i+1})}$. Thus, at iteration $i+1$ all bridges that contain u are deleted from S, and no hyper-bridge added to $\hat{S}$ at iteration $j > i+1$ contains u. Hence, u is included in at most two hyper-bridges of $\hat{S}$. It follows that each link $l \in \hat{\mathcal{R}}$ is included in at most two bridges of $\hat{\mathcal{R}}$. ■