Diversity Coding-Based Survivable Routing with QoS and Differential Delay Bounds

Alija Pašić[a], Péter Babarczi[a], Attila Kőrösi[b]

[a] *MTA-BME Future Internet Research Group, Budapest University of Technology and Economics, Hungary*
[b] *MTA-BME Information Systems Research Group, Budapest University of Technology and Economics, Hungary*

Abstract

Survivable routing with instantaneous recovery gained much attention in the last decade, as in optical backbone networks even the shortest disruption of a connection may cause tremendous loss of data. Recently, strict delay requirements emerges with the growing volume of multimedia and video streaming applications, which have to be ensured both before and after a failure. Diversity coding provides a nice trade-off between the simplicity of dedicated protection and bandwidth-efficiency of network coding to ensure instantaneous recovery for the connections. Hence, in this paper we thoroughly investigate the optimal structure of diversity coding-based survivable routing, which has a well-defined acyclic structure of subsequent paths and disjoint path-pairs between the communication end-points. We define the delay of these directed acyclic graphs, and investigate the effect of Quality-of-Service and differential delay bounds on the solution cost. Complexity analysis and integer linear programs are provided to solve these delay aware survivable routing problems. We discuss their approximability and provide some heuristic algorithms, too. Thorough experiments are conducted to demonstrate the benefits of diversity coding on randomly generated and real-world optical topologies.

Keywords: survivable routing, delay-aware routing, QoS routing, diversity coding, instantaneous recovery

1. Introduction

With the proliferation of multi-media and streaming applications, the end-to-end delay characteristics of the connections are getting more and more into the spotlight. Besides these new applications (e.g., telesurgery,

Email addresses: pasic@tmit.bme.hu (Alija Pašić), babarczi@tmit.bme.hu (Péter Babarczi), korosi@tmit.bme.hu (Attila Kőrösi)

stock market, VoIP, etc.) are highly delay sensitive, they require high resilience from the underlying network. Satisfying both constraints at the same time in a bandwidth-efficient way is unquestionably one of the most challenging tasks of service providers in transport networks.

Survivable routing [1] approaches – e.g., dedicated protection [2] methods – ensure high resilience through sending the connection's data on disjoint (end-to-end) paths, hence, they survive the failure of an arbitrary link, which is the most common fault in transport networks [3]. Although dedicated protection approaches use excessive amount of network resources (e.g., bandwidth), they satisfy the high resilience requirement of these new applications by providing *instantaneous recovery*[1]. However, while maintaining instantaneous recovery, network resources can be used in an efficient manner only if further techniques are applied above routing, e.g., *network coding* [5, 6].

Optimal bandwidth efficiency of dedicated protection approaches with network coding was investigated [4], and it was shown that in order to reach that the user data might be split into arbitrary many parts. Although suitable for a theoretical lower bound, from a practical point of view (e.g., network equipment and management complexity) this can not be implemented. Hence, survivable routing with diversity coding (SRDC) was introduced in [7] where *user data is divided into at most two parts* in order to ensure instantaneous recovery, while approaching the theoretical lower bound in bandwidth efficiency. In [8] it has been proven that *every minimum cost SRDC solution can be decomposed into three end-to-end directed acyclic graphs (DAGs)*, forwarding the two data parts (A and B) along some redundancy data ($A \oplus B$, i.e., the eXclusive OR), respectively. Thus, SRDC applies coding only at the end-nodes of the connection while avoids any complicated hardware deployment in the core network [9].

Combining the results of [7, 8], the bandwidth-efficient survivable routing problem with diversity coding turns into finding three appropriate DAGs between the communication endpoints s and t. With SRDC instantaneous recovery is ensured as the data transmitted on the DAGs is unchanged upon an arbitrary failure occurs. This high resilience of SRDC was demonstrated in [10] through a video streaming application scenario, but it was also noted that the end-to-end delay and the delay difference should be considered in the routing problem [11]. Although several works are dealing with Quality-of-Service (QoS) routing [12] and differential delay (DD) aware [13] survivable routing in optical networks, all of the methods are designed for disjoint paths only. In this paper we generalize these results for diversity coding-based survivable routing using DAGs. Hence, we define the before- and after-failure delay of an end-to-end DAG, and investigate the effect of QoS

[1]Note that, as no flow rerouting or packet retransmission is required upon failure, the after-failure signaling is completely eliminated from their recovery process [4].

routing and DD-aware routing delay constraints on these optimal survivable routing structures ensuring instantaneous recovery.

The rest of the paper is organized as follows. In Section 2 the related work is presented focusing on previous survivable QoS routing and differential delay aware routing results. In Section 3 the preliminaries and problem formulation for survivable routing is discussed in details, and the delay of a DAG is defined. We show a graph transformation to an equivalent survivable routing problem, too, which traces back our problem to finding disjoint paths with delay constraints in a special graph. Section 4 introduces our complexity analysis, integer linear program and approximability result for the survivable QoS routing problem, while the same is presented together with a heuristic solution for the differential delay aware routing problem in Section 5. Experimental results are shown in Section 6, and the paper is concluded in Section 7.

2. Related work

2.1. QoS Routing

Finding a minimum cost (or shortest) path while minimizing a single metric (e.g., cost or length) can be solved in polynomial time with Dijkstra's algorithm. On the other hand, satisfying an additional constraint (e.g., delay or jitter) along the path while minimizing its cost is a fundamental problem and arises in several application scenarios, referred to as Quality-of-Service (QoS) routing [14]. Note that, this problem is already NP-hard [15], called the shortest weight-constrained path problem (or restricted shortest path problem), where a minimum cost path is required between the source s and destination t, such that the delay of the path is lower than a pre-defined bound D. An exact solution can be found by pseudo-polynomial algorithms, thus, if the input parameters are bounded it can be solved in polynomial time through a standard dynamic programming approach [16]. Furthermore, ϵ-optimal fully polynomial approximation schemes (FPTAS) exists for the problem both for acyclic [17] and general graphs [18].

The problem of finding two link-disjoint paths while minimizing the total cost of the paths is solvable in polynomial-time, e.g., with the Suurballe-Tarjan [19] algorithm. In order to extend this work to QoS routing, in [12] the 2-Restricted Link Disjoint Paths (2DP) problem was introduced where the the total cost of the paths was minimized while both paths have to obey a specific delay bound D. Note that, even if we assume that each link has zero cost, it is NP-hard to find a solution that does not violate the delay constraint D of at least one of the paths [12] (following from the fact that finding a disjoint path-pair while minimizing the delay of the longer path is NP-hard [20]). Furthermore, no polynomial-time algorithm exists which can approximate the delay within a factor of $1 \leq \alpha < 2$ [12]. However, a 2-approximation on the delay-bound can be provided with minimizing the

total delay of the disjoint path-pair, e.g., with the Suurballe-Tarjan [19] algorithm. Thus, in [12] four bifactor approximation algorithms were presented, i.e., both the total cost and the total delay of the path-pair can be bounded with a constant factor. The first approximation algorithm 2DP-1 uses standard flow techniques, i.e., employs the path augmenting approach to find the restricted shortest paths [18] in the residual network with a delay bound of D and $2D$ for the first and second path, respectively. Thus, the approximation ratio is 1.5 on the delay, and it is a slightly larger factor on the cost. The subsequent approximation algorithms reduce the computational complexity and the delay violation at the expense of higher total cost.

The problem of finding a set of k link-disjoint paths from s to t, such that the total cost of these paths is a minimum and that the delay for each path is not greater than a specified bound D was introduced in [21]. Of course, this problem contains the problem of 2DP [12], thus, it is also NP-hard. The approximation ratios of [12] were generalized for k paths and improved as well in [22]. Besides obeying a delay bound for each individual path, in [21] a more general network programming based approach was presented for finding k constrained shortest link-disjoint paths, such that the *overall delay of these paths* should be lower than a specified bound (kD). Note that, the approximation algorithms in [12] also relax the problem of 2DP with delay constraint D on both paths to a minimum constrained flow problem with a total delay constraint of $2D$ on the solution.

2.2. Differential Delay Aware Routing

Although the delays of individual paths in QoS routing is an important question, from a practical point of view the difference between the path delays could be a more serious issue in some application environments. For example, with the deployment of next-generation SONET/SDH technology virtual concatenation (VC) enabled service providers to split the traffic of a single circuit into multiple finer granularity parts, and route these parts along multiple paths. However, besides of the several advantages the application of VC provides, it introduces *differential delay (DD)* among the diversely routed paths as well, which boiled down to the issue of increased buffer size at the destination node for DD compensation. In order to avoid service degradation, differential delay of the paths should be considered in the routing problem, as in optical networks the maximal DD compensation is about 125 ms with off-chip SDRAM technology [13].

The authors in [23] introduced the Two-Sided Constrained Path (TSCP) problem, where the task is to decide whether a new VC can be added to a VC group. Formulating with the DD constraint, the task is finding a path with overall delay of D between a given minimum and a maximum bound, i.e., $D_{min} < D < D_{max}$. It was proved that the TSCP problem is NP-hard [23]. In [24], the DD is defined as the difference between the delay of the highest and smallest delay paths. In their Differential Delay Routing

(DDR) problem the task is to find a given number of paths, while their DD is lower than a pre-defined delay bound. It was shown that minimizing the delay difference of paths in DDR is not only NP-hard but provably hard to approximate within a constant factor. In [25] the same authors introduce the cumulative differential delay, which is the sum of the differences of delays of all the paths of a solution compared to the highest delay path. Also heuristic approaches were introduced like the Sliding-Window Algorithm and the Cost-Based Algorithm, which are based on finding k-shortest paths.

In the previous works the objective function was to minimize the differential delay, while neither link costs nor the disjointness of the paths were considered in the optimization. The study in [13] extended DD aware multi-path routing with survivability. Contrary to the differential delay problems above, their goal is to minimize the total cost of the paths while the disjointness of these paths is required in order to ensure single link failure resilience. The mathematical formulation of the survivable multi-path DD constrained routing problem has been presented in [13], where a DD bound have to be satisfied between each pair of the k paths. The NP-completeness of this problem follows from the DDR problem [24]. Thus, two heuristic approaches were introduced based on k-shortest link-disjoint paths, both of them following the Shared Protection of the Largest Traversed link (SPLIT) approach.

3. Problem Formulation

In the *Delay Aware Routing with Coding (DARC)* problem [11] the network is represented by a directed graph $G = (V, E, k, c, d)$ with node set V, arc set E, and three additional attributes for each arc $e \in E$:

- the capacity $k(e) \in \mathbb{N}$, i.e., the number of bandwidth units available for data transmission;
- a non-negative cost function $c(e) \in \mathbb{R}^+$, which is defined as the cost of using one unit of bandwidth along arc e;
- and the delay $d(e) \in \mathbb{R}^+$, which corresponds to the time transmitting data between the end nodes of the arc.

Note that, arc defines the direction of the communication between two adjacent nodes. We will use the more general term "link" if the direction of the arc or (anti-)parallel arcs between two nodes is not relevant. We consider a dynamic routing scenario, where traffic demands arrive one after the other, without any knowledge of further incoming requests, i.e., each request is routed independently. Thus, as part of the input of the delay aware survivable routing problem a single *connection request* $C = (s, t, b, D)$ is given, which consists of the source node $s \in V$, destination node $t \in V$,

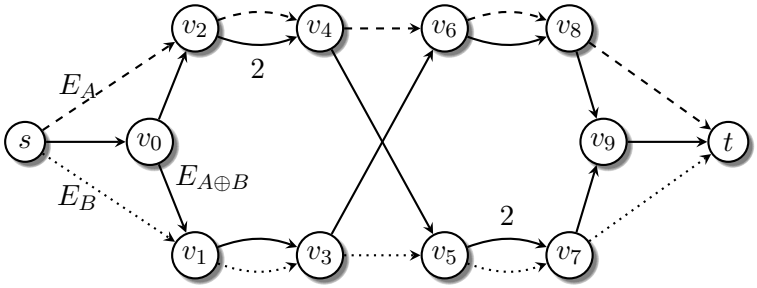


Figure 1: A survivable SRDC solution for request $C = (s, t, 2, -)$ with the corresponding routing DAGs E_A, E_B and $E_{A\oplus B}$. Arcs with $f(e) = 2$ are duplicated. Arc costs are unit. The arc delays are $d(e) = 1$, otherwise written next to the arc.

the number of bandwidth units b requested for data transmission, and a maximal delay bound D which have to be satisfied by the routing of the request to ensure a given QoS or differential delay for the connection.

3.1. Preliminaries

In this section we recall the main findings of SRDC, which gives the starting point of our work. Throughout this paper, we build on the following definition of survivable routing [7]:

Definition 1. *We say that $R = (V^R, E^R, f)$ is a **survivable routing** of a connection $C = (s, t, b, D)$ in G with flow values f (where $V^R \subseteq V$, $E^R \subseteq E$), if there is an $s - t$ flow of value $F \geq b$ in R, even if we delete any single arc of R.*

In SRDC, our objective is to minimize the total bandwidth cost of the survivable routing solution for connection request[2] $C = (s, t, b = 2, -)$, formally:

$$\min_R \sum_{e \in E^R} c(e) \cdot f(e). \tag{1}$$

A minimum cost SRDC solution for the input $C = (s, t, 2, -)$ with respect to Eq. (1) has $\forall e \in E^R : f(e) \leq 2$ (consequence of [8, Theorem 1]). Furthermore, E^R always can be decomposed into three arc sets $E_A, E_B, E_{A\oplus B}$, respectively transmitting data parts A and B and $A \oplus B$, called **routing DAGs**. Thus, if we delete an arbitrary arc $e \in E^R$ from an SRDC solution, there *remains at least two routing DAGs in which a path connects s to t* (satisfying [8, Theorem 2]) in order to guarantee instantaneous recovery (i.e., no flow rerouting is required on the intact arcs).

[2]Remember that in SRDC we divide user data into two equal-sized parts A and B – e.g., packets with odd and even sequence number – represented by $b = 2$, while no delay bounds are given.

An example is presented in Fig. 1. If an arc with $f(e) = 1$ fails, e.g., (s, v_1), it disrupts only a single routing DAG (E_B) and the above claim trivially holds. Further note, that the failure of an arc with $f(e) = 2$, e.g., (v_1, v_3) affects two routing DAGs, but in different ways: while routing DAG E_B is disrupted, only the end-to-end delay increases for the other routing DAG $E_{A\oplus B}$. This is because E_B is simple path $s \to v_1 \to v_3 \to v_5 \to v_7 \to t$ in Fig. 1, and when arc (v_1, v_3) fails, the path is disrupted. On the other hand, $E_{A\oplus B}$ is not a simple path, it contains a segment between v_0 and v_9 which is a link-disjoint path-pair. In the operational state (i.e., no failure occurs) the data is transmitted along the lower delay path between v_0 and v_9, that is $v_0 \to v_1 \to v_3 \to v_6 \to v_8 \to v_9$ with total delay of 5 units. If (v_1, v_3) fails this path can not be used anymore, but the data flow of $E_{A\oplus B}$ still reaches destination t on the other path of the segment between v_0 and v_9, that is $v_0 \to v_2 \to v_4 \to v_5 \to v_7 \to v_9$ with total delay of 7 units. Thus, $s-t$ connectivity is still ensured without any reconfiguration of the network. Note that, in this example the end-to-end delay between s and t increased from 7 to 9 units on routing DAG $E_{A\oplus B}$ in Fig. 1 upon the failure of arc (v_1, v_3).

It has been proven in [7] that the structure shown in Figure 1 is general for each *minimum cost SRDC solution*, i.e., each routing DAG $E_j : j \in \{A, B, A\oplus B\}$ consists of a series of paths (denoted as $\mathcal{P}_{E_j}$) and disjoint path-pairs, called **islands** (referred to as $\mathcal{I}_{E_j}$). Furthermore, each island is part of at most one routing DAG. In Figure 1 routing DAGs E_A and E_B consist of a single $\mathcal{P}_{E_A} = \mathcal{P}_{E_B} = \{s \to t\}$ path. On the other hand, routing DAG $E_{A\oplus B}$ consist of $\mathcal{P}_{E_{A\oplus B}} = \{s \to v_0, v_9 \to t\}$ paths, and $\mathcal{I}_{E_{A\oplus B}} = \{v_0 \to v_9\}$ disjoint path-pair (island with *splitter node* v_0 and *merger node* v_9). As the same routing structure must be satisfied by DARC, we assume that the routing DAGs in a DARC solution can be decomposed into subsequent paths and islands, too. After defining the delay of a routing DAG in Section 3.2, we introduce a graph transformation in Section 3.3 which traces back the survivable routing problem to finding minimal cost routing DAGs composed of paths and islands obeying specific delay bounds.

3.2. Defining the Delay of a Routing DAG

Note that, the implementation of the merger node (i.e., which selects among the two identical copies of the same data part arriving on the two paths of an island, e.g., v_9 of $E_{A\oplus B}$ in Fig. 1) is crucial to ensure instantaneous recovery [4]. In order to eliminate signaling from the recovery process, a link failure should be oblivious to the merger node, i.e., the merger has to switch from the failed path autonomously to the operating path. The merger node implementation in [4] solves this issue by tracking of the highest sequence number of the forwarded packets (SEQ_{FW}). A packet p is only forwarded on the merger's outgoing arc if its sequence number SEQ_p is larger than SEQ_{FW}, and it sets $\text{SEQ}_{FW} = \text{SEQ}_p$ upon forwarding. As a result,

a merger forwards the packets from the "faster" path from the two disjoint paths of an island (I_{min}) in a failure-less state, and discards the duplicates that arrive on the "slower" path of the island (I_{max}). On the other hand, if a failure occurs on the faster path (as we have seen in Fig. 1 after the failure of (v_1, v_3)), the merger will forward the packets arriving on the slower path on its outgoing arc automatically. Note that, at switching some jitter could occur on the routing DAG owing to the delay difference between the last packet arrived on I_{min} and the first forwarded packet from I_{max}.

In order to capture the before- and after-failure delay characteristics of the routing DAGs, we introduce two delay values for each island I: $d^I_{min} = \sum_{e \in I_{min}} d(e)$ corresponding to the delay of the island in the failure-less state (i.e., the faster path); and the delay difference between the two disjoint paths $\Delta^I = \sum_{e \in I_{max}} d(e) - \sum_{e \in I_{min}} d(e)$ corresponding to the delay increase between the splitter and merger node of the island upon a failure occurs on the faster path. Thus, the *end-to-end delay of a routing DAG* can be modeled as

$$\delta_{E_j} = \sum_{P \in \mathcal{P}_{E_j}} \sum_{e \in P} d(e) + \sum_{I \in \mathcal{I}_{E_j}} d^I_{min} \tag{2}$$

in the failure-less state, while it increases to

$$\Delta_{E_j} = \delta_{E_j} + \max_{I \in \mathcal{I}_{E_j}} \Delta^I \tag{3}$$

in worst case upon a failure along the island with the largest delay difference between its two paths[3].

In the example in Figure 1, $\delta_{E_{A \oplus B}} = 7$ because the delay of $s \to v_0 \to v_1 \to v_3 \to v_6 \to v_8 \to v_9 \to t$ is 7 units. For $E_{A \oplus B}$ $\max \Delta^I = 2$, because the delay difference between the two disjoint paths $v_0 \to v_1 \to v_3 \to v_6 \to v_8 \to v_9$ and $v_0 \to v_2 \to v_4 \to v_5 \to v_7 \to v_9$ of the single island in $\mathcal{I}_{E_{A \oplus B}}$ is 2 units. Of course this means $\Delta_{E_{A \oplus B}} = 9$.

Further note that, besides instantaneous recovery diversity coding provides additional benefits to the connections in the failure-less state as well. On one hand, in multi-path routing (or shared path protection approaches [13, 6]) we have to wait for the highest delay path to reconstruct user data. On the other hand, the additional redundancy provided by diversity coding ensures that the two lower delay paths determine the delay of the connection (as we can *reconstruct user data from arbitrary two of the three routing DAGs* with low additional coding complexity owing to the applied simple XOR codes [4]). This may lead to lower overall end-to-end delay for the applications in the failure-less state, and also results lower jitter and differential delay increase caused by protection switching.

[3]Note that, $\Delta_{E_j} - \delta_{E_j}$ gives also the largest jitter we might expect along the routing DAG caused by protection switching.

3.3. An Equivalent Survivable Routing Problem

Using the polynomial-time graph transformation proposed in [7] the minimum cost survivable routing problem of SRDC was traced back to finding three link-disjoint $s-t$ paths with minimum total cost in an auxiliary graph G^*. In this section we extend the transformation with the delays of the routing DAGs, which enables the incorporation of additional delay constraints for survivable QoS routing and DD aware routing.

The input of DARC is the graph $G = (V, E, k, c, d)$ with delay values $d(e)$ for every arc e. An auxiliary (multi-)graph $G^* = (V, E^*, c^*, d_{min}, \Delta)$ is created, where:

- node set V in G^* is the same as in G, while
- arc set E^* contains the *original arcs* of G. Additional *virtual arcs* $e_{(u,v)}$ are added between every pair of distinct node-pairs for which a link-disjoint path-pair exist, i.e., representing a potential island I with splitter node u and merger node v.
- The cost of $c^*(e_{(u,v)})$ is set to the cost of a minimum cost link-disjoint path-pair between nodes u and v in G (calculated with Suurballe's algorithm [26]). The original arcs of G have the same cost ($\forall e \in E : c^*(e) = c(e)$).
- In addition to the previous transformation for SRDC, in DARC we have to capture the routing DAG delays in Eq (2)-(3). Thus, two variables d^I_{min} and Δ^I are introduced for each virtual arc (island) $I = e_{(u,v)}$ (i.e., the delay of I_{min} and the delay difference between I_{min} and I_{max}). For original arcs $e \in E$ we define $d^e_{min} = d(e)$, and $\Delta^e = 0$.

For $G = (V, E)$ in Fig. 1 ten virtual arcs should be added during the transformation. For example, a virtual arc $e = (v_0, v_9)$ represents potential island with splitter v_0 and merger v_9, and it has $c^*(e) = 10$, $d^e_{min} = 5$, $\Delta^e = 2$. Note that, if the original graph G was 2-link-connected, then G^* contains the original arcs and a full mesh of virtual arcs.

An optimal survivable routing in G^* – that is, three link-disjoint paths with minimum total cost – either with or without obeying delay bounds can be easily transformed back to the routing DAGs in G. The arcs of the routing DAGs are the arcs of the three paths in G^*, with the difference that the virtual arcs $e_{(u,v)}$ in G^* are replaced with the original arcs of the corresponding disjoint path-pairs (islands). As in a minimum cost survivable routing solution $\forall e \in E^{\mathcal{R}} : f(e) \leq 2$ (consequence of [8, Theorem 1]) [7], the replacement of virtual arcs with original arcs can always be done if $\forall e \in E : k(e) = 2$ (or greater). We argue that this assumption is reasonable, as the bandwidth of the connection request itself is $b = 2$ units (or can be scaled accordingly). In other words we only assume that each arc of

the network is capable of carrying the whole data of the connection request (can be removed from the input graph G otherwise). Thus, without loss of generality, in the rest of the paper we assume that $\forall e \in E : k(e) \geq 2$ at the arrival of each connection request.

Finding link-disjoint paths with minimum total cost in G^*, thus, the SRDC problem is polynomial-time solvable [26]. However, this won't be true for DARC with additional delay constraints, as we have seen that finding link-disjoint paths while obeying some additional delay constraint is already an NP-complete problem [12, 15, 24, 13]. We note here that, we can use the transformed graph to run algorithms, but owing to the correlation between the arcs and arc parameters of G^* *the hardness results can not be directly transformed back to our original survivable routing problem.* Thus, further investigation is required to show the complexity of our survivable QoS routing and DD aware routing problems, done in Section 4 and Section 5, respectively.

4. Computing Routing DAGs for QoS Routing

In this section we investigate the survivable routing problem when delay constraints are given for the individual DAGs to ensure QoS routing (DARC-QoS). Formally, the task is to minimize the total cost in terms of Eq. (1), while the after-failure delay for each routing DAG is less than a given bound D [12]:

$$\forall j \in \{A, B, A \oplus B\} : \Delta_{E_j} \leq D \ .$$

First, the complexity of DARC-QoS is discussed in Section 4.1, and it is shown that DARC-QoS is NP-complete. Thus, in Section 4.2 we present an Integer Linear Program (ILP), while in Section 4.3 the approximability of the problem is discussed.

4.1. Complexity Analysis of DARC-QoS

In this section we show that DARC-QoS is NP-complete. The proof is based on the reduction from the Three-Way Partition problem [15], which is known to be NP-hard (following also from the fact that in an optimal three-way partition the items in arbitrary two subsets are partitioned into two equal parts).

Theorem 1. *To decide whether a $\leq Z$ cost solution for DARC-QoS exists is NP-complete.*

Proof. DARC-QoS is in NP, a solution with $\leq Z$ cost with delay $\leq D$ on each routing DAG is a proof.

Assuming we are given an instance of the Three-Way Partition Problem [15], [27], that is, a finite set A of items with size $s(a) \in \mathbb{Z}^+$ for

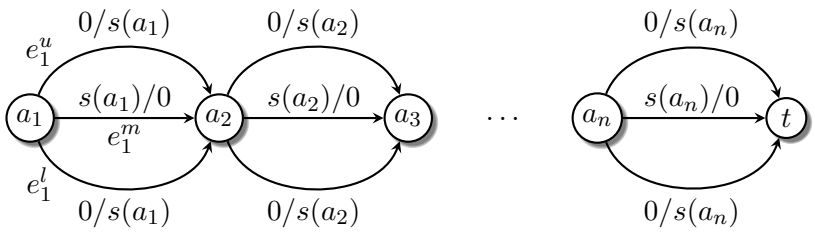


Figure 2: Transformation of Three-Way Partition [15] to DARC-QoS. Arc delays/arc costs are shown next to the arcs. Each arc has $k(e) = 2$ units of free capacity.

each $a \in A$. Let us denote $T = \sum_{a \in A} s(a)$. Is there a perfect three-way partition of set A, i.e., partitions P_1, P_2, P_3 such that $P_1 \cup P_2 \cup P_3 = A$, $\sum_{a \in P_1} s(a) = \sum_{a \in P_2} s(a) = \sum_{a \in P_3} s(a) = T/3$ and $P_1 \cap P_2 = \emptyset$, $P_1 \cap P_3 = \emptyset$, $P_2 \cap P_3 = \emptyset$.

The polynomial time transformation for Three-Way Partition with $|A| = n$ to DARC-QoS is given as follows (shown in Figure 2). We construct a graph with $n+1$ nodes using the following gadget for each a_i: we add three arcs (upper, middle and lower) $e_i^u = e_i^m = e_i^l = (a_i, a_{i+1}), i = 1, 2, \ldots, n$ (with $t = a_{n+1}$). We define arc delay and arc cost values according to $s(a_i)$, as shown in Figure 2. We define $Z = 2T$ and $D = T/3$ for the connection request $C = (s = a_1, t, 2, D)$. Next, we show that the two instances are solvable at the same time.

($\Rightarrow$) First, we show how to convert a DARC-QoS solution with $\leq Z$ cost to a Three-Way Partition of A. From the survivability aspect we know that for every gadget $f(e_i^u) + f(e_i^l) \geq 2$, as the remaining flow should be at least two after the failure of e_i^m. If we sum up this for all gadgets, this leads to $\sum_{e \in E} f(e_i^u) + f(e_i^l) \geq 2T$. However, the total cost is $\leq 2T$, thus, $\forall i : f(e_i^u) + f(e_i^l) = 2$ follows. Furthermore, $f(e_i^m) \geq 1$ follows as well because of the survivability aspect (i.e., the remaining flow should be at least two either e_i^u or e_i^l fails). On the other hand, from $D = T/3$ on the individual routing DAGs we know that the total delay suffered by the three routing DAGs is $\leq T$. Thus, $f(e_i^m) \leq 1$ follows, because otherwise this total delay bound, hence, the delay bound of the individual routing DAGs could not be satisfied. So at the end $f(e_i^m) = 1$, which leads to $f(e_i^u) = 1$ and $f(e_i^l) = 1$ to maintain survivability. This means that a DARC-QoS solution with $\leq Z$ cost, i.e., the three routing DAGs E_A E_B and $E_{A \oplus B}$ are three $s - t$ paths in Figure 2. As the total delay of these routing DAGs is T (as e_i^m is traversed exactly by one routing DAG in each gadget), their delay is exactly $T/3$. As the delay of e_i^m was set as the weight $s(a_i)$, the partitions of a_i are defined by which routing DAG traverses e_i^m.

($\Leftarrow$) In the other direction, it is easy to convert a Three-Way Partition into three routing DAGs with $\leq D$ delay, i.e., in this case to three $s - t$ paths. Let assume the three partitions P_1, P_2 and P_3 are given. Without

loss of generality, we assume that

- the single $s-t$ path in $\mathcal{P}_{E_A}$ is using $\forall a_i \in P_1 : e_i^m$, and e_i^u otherwise,
- the single $s-t$ path in $\mathcal{P}_{E_B}$ is using $\forall a_i \in P_2 : e_i^m$, and e_i^l otherwise,
- and the single $s-t$ path in $\mathcal{P}_{E_{A\oplus B}}$ is using $\forall a_i \in P_3 : e_i^m$, and otherwise the remaining available arc of the gadget.

Thus, a Three-Way Partition gives a DARC-QoS a solution with $\leq Z$ cost. This concludes the proof as the two problems are solvable at the same time.

4.2. Integer Linear Program for DARC-QoS

Here, we present an ILP formulation for DARC-QoS based on [7, 11] for finding three minimum cost link-disjoint paths in $G^* = (V, E^*, c^*, d_{min}, \Delta)$ with additional delay bounds required for QoS routing. The connection request is $C = (s, t, b = 2, D)$. The *three paths corresponding to the routing DAGs* are denoted as $w \in \{A, B, A \oplus B\} = \mathcal{W}$, respectively. Binary variables $x^w(e)$ are used to indicate the paths for each routing DAG. Our objective is to minimize the total bandwidth cost in terms of Eq. (1):

$$\min \sum_{w \in \mathcal{W}} \sum_{e \in E} c^*(e) \cdot x^w(e). \tag{4}$$

The following constraints are required to find a survivable routing:

$$\forall w \in \mathcal{W}, \forall i \in V\text{:}$$

$$\sum_{(i,j)\in E} x^w(i,j) - \sum_{(j,i)\in E} x^w(j,i) = \begin{cases} 1 & \text{, if } i = s \\ -1 & \text{, if } i = t \\ 0 & \text{, otherwise} \end{cases}, \tag{5}$$

$$\forall e \in E\text{:} \sum_{w \in \mathcal{W}} x^w(e) \leq 1, \tag{6}$$

$$\forall w \in \mathcal{W},\ \forall e \in E\text{:}\ x^w(e) \cdot \Delta^e \leq y^w. \tag{7}$$

Constraint (5) formulates the flow conservation for each path w. Constraint (6) ensures the disjointness of the paths. Constraint (7) gives a lower bound for the integer variable y^w, which captures the worst case delay increase of path (routing DAG) w upon a single link failure formulated in Eq. (3). For DARC-QoS, we have to add Constraint (8), which ensures that the maximal delay of the routing DAGs is less than the specified bound:

$$\forall w \in \mathcal{W}\text{:} \sum_{e \in E} x^w(e) \cdot d_{min}^e + y^w \leq D. \tag{8}$$

From the paths defined by the binary variables $x^w(e)$ in G^* the routing DAGs in G can be easily obtained by replacing the virtual arcs by the original arcs of the corresponding link-disjoint path-pair.

Table 1: The end-to-end delay difference on the two fastest routing DAGs upon a single link failure (wlog $\delta_{E_A} \leq \delta_{E_B} \leq \delta_{E_{A\oplus B}}$).

disrupted \ inc. delay	$\emptyset$	E_A	E_B	$E_{A\oplus B}$
$\emptyset$	$\lvert\delta_{E_A} - \delta_{E_B}\rvert$	$\lvert\delta_{E_B} - \min\{\Delta_{E_A}, \delta_{E_{A\oplus B}}\}\rvert$	$\lvert\delta_{E_A} - \min\{\Delta_{E_B}, \delta_{E_{A\oplus B}}\}\rvert$	$\lvert\delta_{E_A} - \delta_{E_B}\rvert$
E_A	$\lvert\delta_{E_B} - \delta_{E_{A\oplus B}}\rvert$	–	$\lvert\Delta_{E_B} - \delta_{E_{A\oplus B}}\rvert$	$\lvert\delta_{E_B} - \Delta_{E_{A\oplus B}}\rvert$
E_B	$\lvert\delta_{E_A} - \delta_{E_{A\oplus B}}\rvert$	$\lvert\Delta_{E_A} - \delta_{E_{A\oplus B}}\rvert$	–	$\lvert\delta_{E_A} - \Delta_{E_{A\oplus B}}\rvert$
$E_{A\oplus B}$	$\lvert\delta_{E_A} - \delta_{E_B}\rvert$	$\lvert\Delta_{E_A} - \delta_{E_B}\rvert$	$\lvert\delta_{E_A} - \Delta_{E_B}\rvert$	–

4.3. Approximability of DARC-QoS

It was shown that it is NP-hard to find two link-disjoint paths that does not violate the delay constraint D of at least one of the paths [12], and this problem even can not be approximated within a factor of 2 on the delay-bound. However, Suurballe's [26] algorithm which minimizes the total delay of the two paths provides a 2-approximation on the delay. On the other hand, for our survivable routing problem – finding three link-disjoint paths which minimizes the average (or total) delay of the routing DAGs – the same algorithm [26] provides only a 3-approximation on the delay, without considering the solution cost in the optimization problem. Although the solutions of the bifactor approximation algorithms [12, 22] gives bound on the total cost of the solution, the two delay parameters on the arcs in DARC-QoS can not be easily incorporated in the path augmentation approach for finding restricted shortest paths. Thus, we cannot hope an efficient algorithm which obeys the delay bound in G^* on each of the three paths. Hence, if strict delay bounds have to be satisfied in DARC-QoS, we suggest the usage of the ILP in Section 4.2, which runs in reasonable time for several instances owing to the small number of constraints on the path delays.

5. Computing Routing DAGs for Differential Delay Aware Routing

In this section we investigate the survivable routing problem when delay bounds are given on the end-to-end delay differences of the routing DAGs (DARC-DD). The task is to minimize the total cost in terms of Eq. (1), while all possible DD bounds in Table 1 have to be satisfied. These constraints ensure that the delay difference between the two fastest routing DAGs is under a specific bound $\leq D$ [13] corresponding to the maximal buffer size when an arbitrary single link failure occurs. Note that, a single link failure can cause the disruption of a routing DAG (if an arc in a path is failed) or increase it's end-to-end delay (if the arc failure affects one of its islands). Furthermore, if an arc with $f(e) = 2$ fails (e.g., (v_1, v_3) in Fig. 1), it could happen that one routing DAG is disrupted while the end-to-end delay increased for another routing DAG.

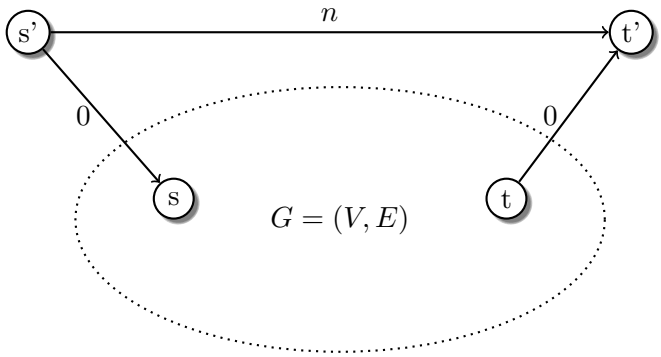


Figure 3: Polynomial-time reduction of the Longest Path problem to DARC-DD. Arc delays are depicted next to the additional arcs, while it is set to the length of the arcs in $G = (V, E)$. Each arc has $k(e) = 2$ units of free capacity.

First, the complexity of the problem is discussed in Section 5.1, and it is shown that DARC-DD is NP-complete. Thus, in Section 5.2 we present an Integer Linear Program (ILP), while in Section 5.3 its approximability is discussed. Finally, in Section 5.4 heuristic approaches are proposed to solve the DD-aware survivable routing problem on the transformed graph introduced in Section 3.3.

5.1. Complexity Analysis of DARC-DD

In this section the NP-completeness proof of DARC-DD is provided. The proof is based on the polynomial reduction from the Longest Path Problem [15], which is known to be NP-complete.

Theorem 2. *To decide whether a $\leq Z$ cost solution for DARC-DD exists is NP-complete.*

Proof. DARC-DD is in NP, a solution with $\leq Z$ cost with $\leq D$ delay difference between the routing DAGs is a proof.

Assuming we are given an instance of the Longest Path Problem [15], that is, a directed graph $G = (V, E)$ with length $l(e) \in Z^+$ for each $e \in E$, a positive integer K, and specified vertices s and t. It should be decided whether there is a simple path in G from s to t of length K or more? We use the version of the problem when $\forall e \in E : l(e) = 1$, which is still NP-complete [15].

The polynomial time transformation for Longest Path Problem to DARC-DD is given as follows (shown in Figure 3). We add two nodes s' and t' to graph G, and connect them to node s and t with arcs $e_s = (s', s)$ and $e_t = (t, t')$, respectively. Also an arc $e_n = (s', t')$ is added between s' and t'. The transformed graph is denoted as $G' = (V', E')$. The cost of all arcs in G' is $\forall e \in E' : c(e) = 1$. The delay of the arcs in G' is set to the length of the arcs in G (i.e., $\forall e \in E : d(e) = 1$), while the delay of additional arcs e_s

and e_t is set to zero. The delay of $e_n = (s', t')$ is set to value $n = |V|$, which is larger than the length of the longest simple path in G (if the graph is Hamiltonian the longest path has length of $n-1$). The DARC-DD problem can be formulated in $G'(V', E')$ as follows: Does there exist a DARC-DD solution with $\leq Z = 2n + 4$ cost and $\leq D = n - K$ differential delay bound for all possible failure scenarios in Table 1 for the connection request $C = (s', t', 2, D)$? We show that the two problems have a positive answer at the same time.

($\Rightarrow$) First, we show how to obtain a longest path in G from a DARC-DD solution in G'. Because of the survivability aspect we know that the arcs e_n, e_s and e_t have the flow value of $f(e_n) = f(e_s) = f(e_t) = 2$ in any DARC-DD solution (as upon the removal of either of them there should remain an $s' - t'$ flow with at least value 2). Thus, s' is a splitter node of an island and a starting node of two path segments. Hence, a routing DAG, without loss of generality $E_{A\oplus B}$ consists of a single path segment $\mathcal{P}_{E_{A\oplus B}} = \{e_n\}$ with $\delta_{E_{A\oplus B}} = \Delta_{E_{A\oplus B}} = n$. Note that, $E_{A\oplus B}$ has the highest δ and Δ delay among the routing DAGs (the delay of simple paths not containing e_n is at most $n-1$). Thus, in order to satisfy all constraints in Table 1 it is enough if both $\delta_{E_A} \geq K$ and $\delta_{E_B} \geq K$.

The second routing DAG, e.g., E_A is a single island $\mathcal{I}_A = \{s' \to t'\}$ with link-disjoint path-pair I^A_{max} and I^A_{min}, where $\Delta_{E_A} = n$ (delay of $I^A_{max} = e_n$). In order to satisfy the DD bound the delay δ_{E_A} of the simple $I^A_{min} = s' \to s \to t \to t'$ path in G' should be at least K. This is exactly the delay of the $s \to t$ segment of I^A_{min} in G as both $d(e_s) = d(e_t) = 0$. The third routing DAG E_B has path segments e_s and e_t, and could have an arbitrary structure of paths and islands inside G. However, we only have to show that a routing DAG with $\delta_{E_B} \geq K$ exists. In fact, path $\mathcal{P}_{E_B} = \{I^A_{min}\}$ is always a feasible option as routing DAG E_B, as $\forall e \in E' : k(e) = 2$. Thus, all constraints in Table 1 is satisfied with the above routing DAGs with a maximal cost of $2n + 4$. Furthermore, the $s \to t$ segment of I^A_{min} is a path in G with at least length K.

($\Leftarrow$) In the other direction, it is easy to convert the longest path solution to three routing DAGs. Let the longest path between s and t be denoted with P_{s-t}. The three routing DAGs are the following: $\mathcal{P}_{E_{A\oplus B}} = \{e_n\}$, $\mathcal{P}_{E_B} = \{s' \to P_{s-t} \to t'\}$ and the third DAG is an island $\mathcal{I}_{E_A} = \{s' \to t'\}$ with link-disjoint path-pair e_n and $s' \to P_{s-t} \to t'$. This solution has $\leq Z$ cost and satisfies delay bound D for all routing DAGs, which concludes the proof.

5.2. Integer Linear Program for DARC-DD

Here, we present an ILP for DARC-DD based on the mathematical formulation of [13] and correcting the one in [11] for finding three minimum cost link-disjoint paths in G^* with additional delay bounds required for DD-aware routing on DAGs. The connection request is $C = (s, t, b = 2, D)$. The

notations are the same as in Section 4.2. Our objective is to minimize the total bandwidth cost as formulated Eq. (4). Similarly to DARC-QoS, Constraint (5) formulates the flow conservation for each path w. Constraint (6) ensures that the disjointness of the paths. Constraint (7) gives a lower bound for the integer variable y^w, which captures the worst case delay increase of path (routing DAG) w upon a single link failure formulated in Eq. (3).

In contrast with the ILP for DARC-QoS, in DD aware routing we can not implicitly assume that the minimum cost flow follows a simple path, as loops or disjoint cycles may be added by the ILP to ensure that the DD bound of the routing DAGs is satisfied. Thus, in order to explicitly ensure that no loops in the path or no disjoint cycles are formed we used the voltage analysis [28]. For this, we have to introduce the following variables:

- d_i indicates if node i is the target node, i.e., if yes $d_i = 1$, otherwise zero.
- z_i^w is a binary variable, z_i^w has value 1 if node i is traversed by flow w, otherwise zero.
- $q^w(e)$ is a non-negative continuous fractional variable, i.e., the *voltage* assigned to a given arc e and flow w.
- ϵ is a predefined small positive constant, which determines the minimum step of the voltage. In our case it is set to $1/|V|$, to ensure that even the longest simple path can be found.

First, we set the variables z_i^w to one if flow w traverses node i in Constraint (9). Constraint (10) says that only an arc with a non-zero flow value can have non-zero voltage. The main idea of the voltage constraint, i.e., eliminate cycles is formulated in Constraint (11). It says that for each node in the flow the sum of the voltages on the outgoing arcs has to be higher than on the incoming arcs. Finally, to avoid loops in the path Constraint (12) ensures that each node could have at most one outgoing arc with non-zero flow value. Thus, we restricted the solution to simple paths.

$$\forall w \in \mathcal{W}, \forall i \in V, \forall \{e \in E | e = (i,j) \vee e = (j,i)\}\colon\ z_i^w \geq x^w(e), \tag{9}$$

$$\forall w \in \mathcal{W}, \forall i \in V, \forall \{e \in E | e = (i,j) \vee e = (j,i)\}\colon\ q^w(e) \leq x^w(e), \tag{10}$$

$$\forall w \in \mathcal{W}, \forall i \in V\colon\ d_i + \sum_{(i,j)\in E} q^w(i,j) - \sum_{(j,i)\in E} q^w(j,i) \geq \epsilon \cdot z_i^w, \tag{11}$$

$$\forall w \in \mathcal{W}, \forall i \in V\colon\ \sum_{(i,j)\in E} x^w(i,j) \leq 1, \tag{12}$$

In order to capture the delay difference between the two fastest (i.e., lowest delay) routing DAGs in DARC-DD, we formulate the order of the DAGs (wlog $\delta_{E_A} \leq \delta_{E_B} \leq \delta_{E_{A\oplus B}}$) in Constraints (13)-(14).

$$\sum_{e\in E} x^A(e) \cdot d^e_{min} \leq \sum_{e\in E} x^B(e) \cdot d^e_{min}, \tag{13}$$

$$\sum_{e\in E} x^B(e) \cdot d^e_{min} \leq \sum_{e\in E} x^{A\oplus B}(e) \cdot d^e_{min}, \tag{14}$$

The differential delay bound D corresponds to paths x^A and x^B in the failure-less state, is formulated in Constraint (15).

$$\sum_{e\in E} [x^B(e) - x^A(e)] \cdot d^e_{min} \leq D. \tag{15}$$

In order to formulate all possible delay differences between the two fastest paths upon a single link failure occurs, we have to formulate all possible situations in Table 1. For example, Constraints (16)-(17) formulate when E_B is disrupted while the end-to-end delay of E_A is increased (as we don't know which one has lower delay we need two constraints):

$$y^A + \sum_{e\in E} [x^A(e) - x^{A\oplus B}(e)] \cdot d^e_{min} \leq D, \tag{16}$$

$$-y^A + \sum_{e\in E} [x^{A\oplus B}(e) - x^A(e)] \cdot d^e_{min} \leq D. \tag{17}$$

5.3. Approximability of DARC-DD

The NP-hardness of finding paths with obeying a given differential delay bound (DDR problem) was proven in [24], and in Section IV C of [24] its approximability was investigated. It was shown that the DDR problem can not be approximated within a factor of n^ϵ for any $\epsilon < 1$ for Hamiltonian graphs, where $n = |V|$ is the number of vertices in graph G. Note that, the DDR problem is defined strictly for paths (and not DAGs). Furthermore, it does not require the disjointness of paths, thus, it does not provide any survivability for the connection. Meanwhile, DARC-DD is defined for the three routing DAGs which have to satisfy survivability and delay requirements simultaneously. Despite the relevant differences of DDR and DARC-DD, their NP-completeness proofs follow similar reasoning. Thus, we claim that constant factor approximation of DARC-DD is NP-hard as well based on [24]:

Claim 1. *The DARC-DD problem can not be approximated within n^ϵ for any $\epsilon < 1$ for Hamiltonian graphs.*

Proof. See the reasoning of [24][Section IV C] for DDR built on the hardness of finding paths longer than a given constant in Hamiltonian graphs [29].

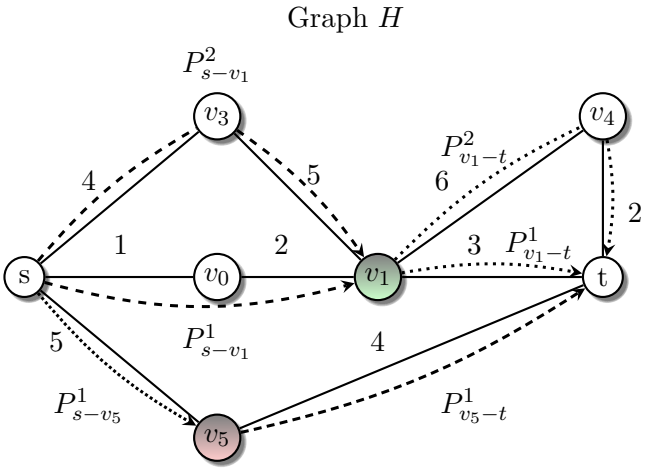


Figure 4: Path generation in the DARC-DD heuristic of Algorithm 1 (illustrating the differences between SPLIT [13] and our approach). The numbers next to the arcs represent the d_{min} value of the arc.

5.4. Heuristic Approach for DARC-DD

Because the problem is hard to approximate, it is legitimate to use a heuristic approach, which leverages the computational complexity of the presented ILP. In [13] two heuristic approaches were introduced based on k-shortest link-disjoint paths, both of them following the SPLIT approach. These methods are based on finding the so called merger nodes (two or more paths crossing at a given node), dividing the paths into segments (starting and ending at these merger nodes), and creating all possible combinations of $s-t$ paths by cascading the path segments. Finally, they distract all maximum disjoint sets of these paths and check whether they satisfy a delay constraint or not. Of course, the SPLIT approach could be modified for finding three disjoint paths in the transformed graph G^* for DARC-DD, which satisfies the delay constrains in Table 1. However, it is obvious that owing to the huge problem space introduced by the investigation of all possible combination of segments, the SPLIT approach would not be a good fit in G^*, where original arcs and a full mesh of virtual arcs are present, leading to a large k in the initial link-disjoint k-shortest path search. This motivated us to create new heuristic in Algorithm 1, which reduces the problem space of SPLIT with a smart and intuitive way of path generation instead of enumerating all possible paths.

Our path generation method is illustrated in Fig. 4 on graph H, which is obtained with a run of the Suurballe algorithm for finding $h = 3$ link-disjoint paths in G^* (Step 4 of Algorithm 1). First the merger node v_1 is processed in Step 7 (this node has the maximum in-degree $m = 2$). In Step 9 the paths $P^1_{s-v_1}$ and $P^2_{s-v_1}$ are found in Fig 4 between s and merger node v_1 with minimum delay 3 and 9, respectively. Next, paths $P^1_{v_1-t}$ and $P^2_{v_1-t}$ are found

Algorithm 1: DARC-DD Heuristic

Input: $G = (V, E, k, c, d)$, $C = (s, t, b, D)$
Result: Routing DAGs E_A, E_B and $E_{A \oplus B}$

begin
  Initialize the number of searched link-disjoint paths $h = 3$.
  `// Create graph` $G^* = (V, E^*, c^*, d_{min}, \Delta)$.
  Create graph G^* according to Section 3.3.
  `// Finding link-disjoint paths in` G^*.
  while *h-link-disjoint paths with minimum total cost exist between s and t in G^** **do**
    Let $H = (V^H, E^H)$ denote the subgraph defined by the link-disjoint paths.
    while *E^H is not empty* **do**
      Find the merger node u with the largest in-degree m in $V^H \setminus \{s, t\}$ (or t if no merger with $m > 0$ exists).
      while *u has incoming arcs in H* **do**
        Find a path P_{s-u} with minimal total d_{min} between s and u.
        Remove the arcs of P_{s-u} from E^H.
      Sort the P_{s-u} paths from lowest delay to the highest (*increasing order*). Denote this sorted set as $\mathcal{P}_{s-u}$.
      while *u has outgoing arcs in H* **do**
        Find a path P_{u-t} with minimal total d_{min} between u and t.
        Remove the arcs of P_{u-t} from E^H.
      Sort the P_{u-t} paths from highest delay to the lowest (*decreasing order*). Denote this sorted set as $\mathcal{P}_{u-t}$
      Take the ith path segments from $\mathcal{P}_{s-u}$ and $\mathcal{P}_{u-t}$ and **create path** P^i_{s-t} from them. Add P^i_{s-t} to set $\mathcal{P}_h$.
    Generate all combination of triplets in $\mathcal{P}_h$, and add the one which satisfies D with minimal cost to $\mathcal{P}$.
    Increase h by 1.
  `// Save routing DAGs in` G.
  Select the solution with minimal cost from $\mathcal{P}$ (if $\mathcal{P} \neq \emptyset$).
  Build routing DAGs from the paths in G^* by replacing virtual arcs with their corresponding islands in G.
  Save the routing DAGs E_A, E_B and $E_{A \oplus B}$.

in Step 13 between nodes v_1 and t with minimum delay 3 and 8, respectively. Next, in Step 16 $s-t$ paths are created from the lowest delay segment between $s-v_1$ and the highest delay segment between v_1-t, from the second lowest and second highest delay segment, etc. Hence, in Fig 4 paths $P^1_{s-t} = \{s \to v_0 \to v_1 \to v_4 \to t\}$ and $P^2_{s-t} = \{s \to v_3 \to v_1 \to t\}$ are created with total delay of 11 and 12, respectively. Note that, the corresponding arcs are removed, leaving node v_5 as the only node $v \in V^H \setminus \{s,t\}$ with in-degree $m > 0$. This means, that the second merger node which is processed in Step 7 has to be v_5. Paths $P^1_{s-v_5}$ and $P^1_{v_5-t}$ are found, and glued together in Step 16 to create $P^3_{s-t} = \{s \to v_5 \to t\}$ with delay minimum of 9. As P^1_{s-t}, P^2_{s-t} and P^3_{s-t} is the only triplet which can be created for $h = 3$ paths, we save it to $\mathcal{P}$ as the minimal cost solution which satisfies the delay bound for $h = 3$ in Step 17. After repeating this procedure while h-link-disjoint paths exist, we select from $\mathcal{P}$ the minimum cost solution in G^*, and obtain the routing DAGs from it.

Note that, the rearrangement of paths in Step 16 of Algorithm 1 is based on $d_{min}(e)$. However, we have two other attributes on the arcs in DARC which can be used in Step 4 to calculate H, i.e., the initial minimum "cost" h-link-disjoint paths. If the $c(e)$ is used as the metric, then we speak about the DD-CH (*Differential Delay - Cost Heuristic*). However, if the delay maximum, i.e., $d_{min}(e) + \Delta(e)$ is considered as the cost metric on the arcs in Step 4, then we talk about the DD-DH (*Differential Delay - Delay Heuristic*). Note that, DD-CH minimizes the cost on expense of no control over the delay. Meanwhile DD-DH minimizes the maximal delay, which may result in lower blocking probability but without any control on the cost of the solution. With rearranging the paths at merger nodes based on the delay minimum (d_{min}) value we get a feasible solution with a higher probability than purely using the link-disjoint paths found in Step 4 based on either metric, as the delay differences are reduced to the lowest possible value.

6. Experimental Results

In this section we investigate the total bandwidth cost of diversity coding based survivable routing DARC, i.e., when not only bandwidth cost is considered but also additional QoS routing and differential delay bounds are given for every connection request. We compare our methods with SRDC, i.e., with the polynomial-time algorithm where only the bandwidth cost is minimized with $\forall e \in E : k(e) \geq 2$, and no additional constraints are given[4]. Of course the solution cost of SRDC increases by introducing additional bounds, but in exchange for that we can guarantee a certain level of QoS,

[4]Note that, the solution cost of SRDC ranges between $1+1$ dedicated path protection (i.e., two link-disjoint paths with flow value two) and $1:2$ shared path protection (i.e., three link-disjoint paths for diversity coding) as the two extremes.

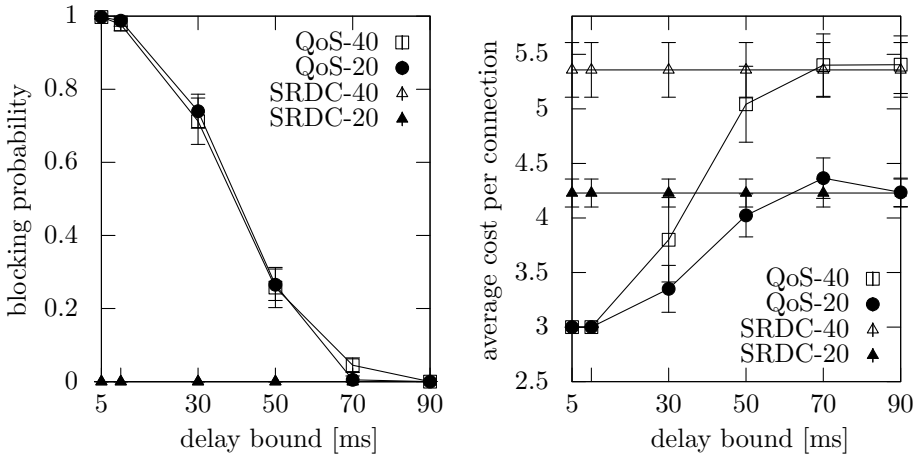
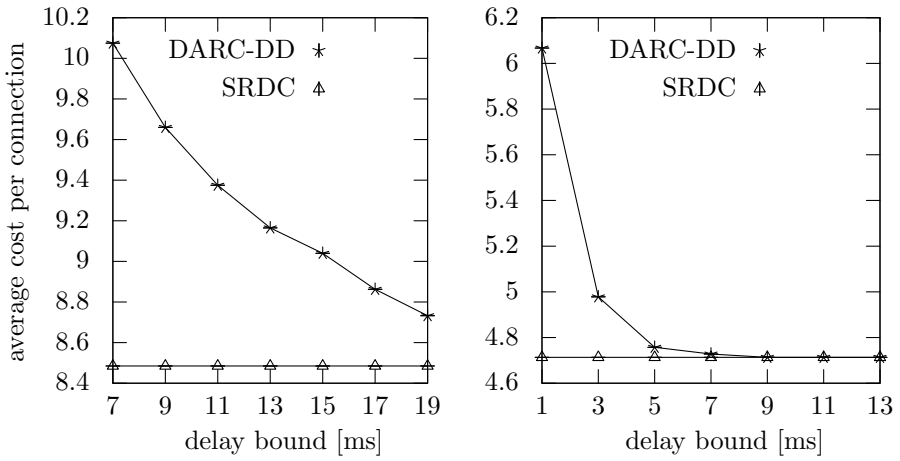


Figure 5: Blocking probability and average cost per connection of the DARC-QoS problem as a function of the delay bound in a maximal planar graph with node number 20 and 40, respectively.

not only in terms of reliability but also in terms of end-to-end and differential delay. This could dramatically improve the user experience of video streaming, which leads in long term to higher revenue and competitive advantage (compared to other providers).

We investigated random generated real-like planar $G = (V, E, k, c, d)$ *network topologies* with different sizes and densities, and some real world optical backbone topologies, too. Bidirectional communication channels are assumed, thus, each link is replaced with two anti-parallel arcs in the topologies[5]. All of the arcs have unit cost ($\forall e \in E : c(e) = 1$). The arc capacities were set high enough so that no blocking occurs due the capacity deficit (i.e., $\forall e \in E : k(e) \geq 2$ satisfied for all requests). Furthermore, the delay of the arcs $d(e)$ is a function of the distance between its adjacent nodes, and scaled into the range of 1 and 25 ms. These values are based on the measurements taken in optical transport networks [4, 13]. Note that, a single random network is generated for each size and density (i.e., 15 different topologies for Sections 6.1-6.2) on the same (unit) square. Up to 600 *connection requests* $C = (s, t, 2, D)$ were generated randomly for each simulation scenario with a given delay bound (while the same demands were considered for SRDC without the delay parameter), thus, we enabled our ILP to run for middle-scale topologies in a reasonable time. Note that, the limited request number is a consequence of the high computational complexity of our ILPs. Furthermore, in order to understand the variability of the results on the traffic in random networks, connection requests were generated in groups of 20 and

[5]Note that, a single link failure disrupts the anti-parallel arcs at the same time.



(a) Cost 266 (37 nodes, 57 links, with diameter 8) [30]

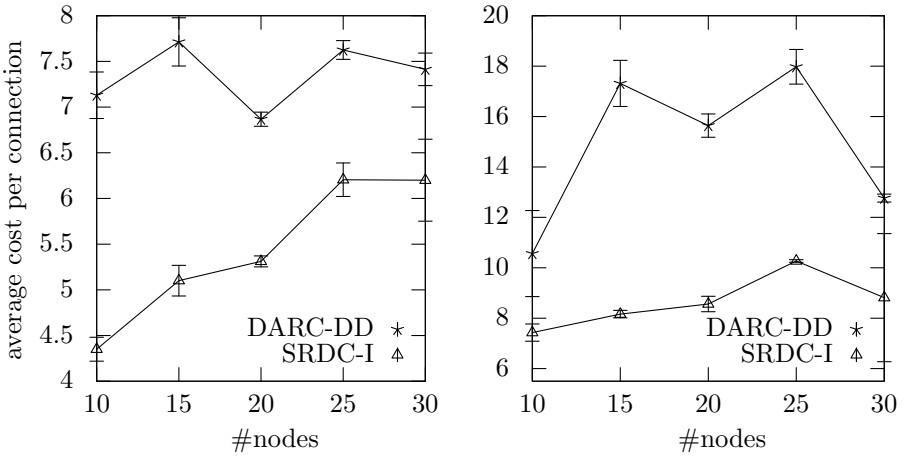
(b) Abovenet (17 nodes, 37 links, with diameter 4) [31]

Figure 6: Bandwidth cost of the DARC-DD problem as a function of the delay bound in real-world topologies.

200 demands, and 95% confidence intervals were added to the corresponding figures.

6.1. Performance Evaluation of DARC-QoS

For the performance evaluation of DARC-QoS total 300 random *connection requests* were generated in groups of 20 demands. In QoS routing the bound given the on path-length might be too strict to satisfy, which depends purely on the physical topology. Thus, if there is no path obeying this bound in the network, then the connection request has to be blocked. This leads us to the recognition that in QoS routing the blocking probability characterizes the problem better than total cost. This value could be an indicator for the network operator, i.e., what percentage of the request can be satisfied with a given QoS level. According to that, in Fig. 5 the blocking probability and the average cost of the routed demands of DARC-QoS is shown for different QoS bounds. One can observe that the blocking probability increases rapidly after a given delay bound is reached. As mentioned before, this is due the fact that if there are no paths shorter than a given bound, then the request gets blocked. If the delay bound is not so strict (e.g., 70 ms), than the blocking probability of DARC-QoS is low, as it is already able to route almost all requests. However, only for the price of longer paths, hence, for higher average cost than SRDC. Finally, if we set D large enough (i.e., 90 ms in the investigated topologies), than we get back the SRDC solution. Note that the results of DARC-QoS do not vary significantly more on traffic than the results of SRDC.



(a) Dense networks (average nodal degree between 3.2 and 4)

(b) Sparse networks (average nodal degree between 2.4 and 2.8)

Figure 7: Bandwidth cost of the DARC-DD problem with delay bound of 9 ms in a single random topology with each size and density.

6.2. Performance Evaluation of DARC-DD

For the performance evaluation of DARC-DD 200 *connection requests* were generated randomly in real-world topologies, while three different sets of 200 requests were used in random networks. In Figure 6 the total bandwidth cost of the DARC-DD ILP is shown depending on the delay bound in real-world topologies. It can be observed that as the delay bound decreases the total cost of DARC-DD increases dramatically. It is foreseeable because DARC-DD takes all possible failure scenarios into account and none of the requests were blocked. This means that if any single link failure event occurs, DARC-DD provided a solution within the given differential delay bound. Therefore, it sacrifices the cost efficiency in order to find three routing DAGs approximately with the same delay. Note that, in Fig. 6b with DD bound of 9 ms DARC still has the same solution as SRDC, while in the larger network in Fig. 6a it already requires longer paths for 19 ms.

Fig. 7 shows the total bandwidth cost in networks with different sizes and densities for a given delay bound (9 ms). The tendencies are the same as in real world topologies, i.e., DARC-DD needs much more resources than SRDC to satisfy all of the constraints related to all single link failures, independently from network size. As mentioned before the networks are generated on a unit square. This means that the average delay parameters (on arcs) for the smaller networks are higher than by larger ones, because the average distance is greater between two adjacent nodes. Furthermore in denser networks (i.e., networks with more arcs) there are more candidate paths than in sparser networks for finding the DARC-DD solution. This

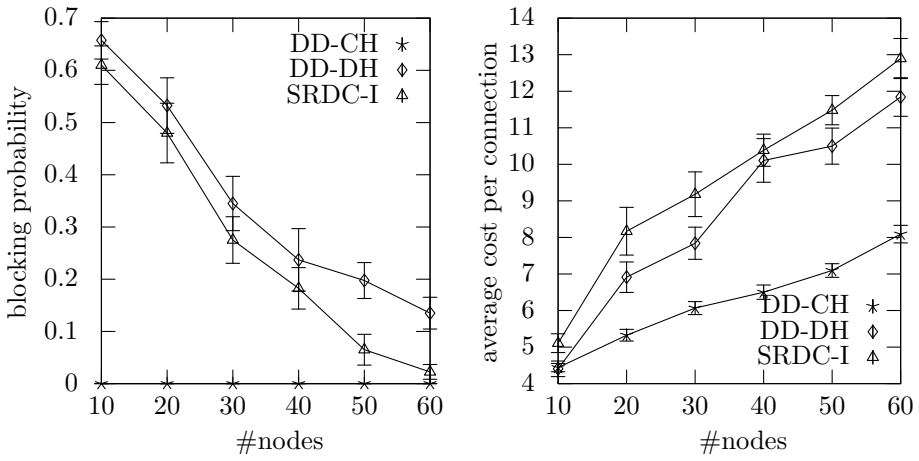


Figure 8: Blocking probability and average cost per connection of the DARC-DD heuristics in dense networks, with delay bound of 9 ms.

leads to the fact that in small sparse networks blocking occurs (resulting also in lower average cost in Fig. 7 (b)). Some bandwidth cost fluctuation can be observed owing to the randomly generated traffic and networks, too, as in larger networks not all $s - t$ pairs are considered as requests.

In Figure 8 the blocking probability and the average cost of the two DARC-DD heuristics are presented in dense networks. One can observe, that the blocking probability is above 50% for smaller networks, thanks to the limited number of link-disjoint k-shortest paths. However, the heuristics get effective when the number of alternative paths increases. Further note that, by design, there is a trade off between the two DARC-DD heuristic versions. On one hand, if our objective is to minimize the blocking probability, then DD-DH is the better choice for the price of higher average cost per connection. On the other hand, DD-CH provides lower average cost per connection, but in this case the blocking probability increases.

Although block several connections, the heuristic approaches decrease the running time of the DARC-DD ILP significantly. For example, in the 30 node sparse network the average running time of the heuristics is less than a second per demand, while on the other hand, the ILP needs an average time of about 40 minutes for one single demand. In other words, the heuristic is about 2400 times faster than the ILP.

7. Conclusion

In this paper we introduced Delay Aware Routing with Coding (DARC) which captures several QoS routing and differential delay aware bounds of diversity coding-based survivable routing. Our work is built on the fact that the optimal solution of this routing problem has a well-defined structure, i.e.,

can be decomposed into three routing DAGs between the source and target node. In order to reach the benefits provided by this redundancy, we defined the delay of routing DAGs capturing all possible single-link failure scenarios. Although DARC inherits several results from previous works on finding three link-disjoint paths between the communication end-points obeying some additional delay bounds, the two delay parameters (i.e., before- and after-failure end-to-end delay) of the routing DAGs leads to a more complex routing problem (even if the problems were already hard for paths both for QoS and differential delay bounds). Thus, we gave NP-completeness proofs for these problems, and demonstrated that although minimizing the total bandwidth cost of routing DAGs is polynomial time solvable, it leads to a hard problem when an additional delay bound introduced on the routing DAGs. We gave an Integer Linear Program for DARC-QoS and DARC-DD to find a minimal cost solution while obeying additional QoS and differential delay bounds, respectively. We discussed the approximability of both problems, while a heuristic approach was proposed for DARC-DD to select paths with appropriate delay differences. Through simulations on small- and medium-scale network topologies we demonstrated the effect of the different delay bounds on the total bandwidth cost of the optimal and heuristic solution.

Acknowledgments

The authors were partially supported by the Hungarian Scientific Research Fund (OTKA grant K108947). P. Babarczi was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences (MTA). This work has been partially supported by Ericsson.

References

[1] J. Rak, *Resilient Routing in Communication Networks.* Springer, 2015.

[2] A. Fumagalli and L. Valcarenghi, "IP restoration vs. WDM protection: is there an optimal choice?" *Network, IEEE*, vol. 14, no. 6, pp. 34–41, 2000.

[3] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *Proc. IEEE Infocom*, vol. 4. Citeseer, 2004, pp. 2307–2317.

[4] P. Babarczi, A. Pasic, J. Tapolcai, F. Németh, and B. Ladóczki, "Instantaneous recovery of unicast connections in transport networks: Routing versus coding," *Elsevier Computer Networks*, vol. 82, pp. 68–80, 2015.

[5] S. Rouayheb, A. Sprintson, and C. Georghiades, "Robust network codes for unicast connections: A case study," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 644–656, 2011.

[6] A. E. Kamal and M. Mohandespour, "Network coding-based protection," *Optical Switching and Networking*, vol. 11, Part B, pp. 189 – 201, 2014.

[7] A. Pasic, J. Tapolcai, P. Babarczi, E. Bérczi-Kovács, Z. Király, and L. Rónyai, "Survivable routing meets diversity coding," in *IFIP Networking*, 2015, pp. 1–9.

[8] P. Babarczi, J. Tapolcai, L. Rónyai, and M. Médard, "Resilient flow decomposition of unicast connections with network coding," in *Proc. IEEE Intl. Symp. on Information Theory (ISIT)*, 2014, pp. 116–120.

[9] A. E. Kamal, A. Ramamoorthy, L. Long, and S. Li, "Overlay protection against link failures using network coding," *IEEE/ACM Transactions on Networking*, vol. 19, no. 4, pp. 1071–1084, 2011.

[10] B. Ladóczki, C. Fernandez, O. Moya, P. Babarczi, J. Tapolcai, and D. Guija, "Robust network coding in transport networks," in *34th IEEE INFOCOM Demo session*, 2015, pp. 27–28.

[11] A. Pasic and P. Babarczi, "Delay aware survivable routing with network coding in software defined networks," in *7th International Workshop on Reliable Networks Design and Modeling (RNDM)*, 2015, pp. 41–47.

[12] A. Orda and A. Sprintson, "Efficient algorithms for computing disjoint QoS paths," in *23th IEEE INFOCOM*, vol. 1, 2004.

[13] S. Huang, C. U. Martel, and B. Mukherjee, "Survivable multipath provisioning with differential delay constraint in telecom mesh networks," *IEEE/ACM Trans. on Networking*, vol. 19, no. 3, pp. 657–669, 2011.

[14] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[15] M. R. Garey and D. S. Johnson, "Computers and intractability: a guide to NP-completeness," 1979.

[16] H. C. Joksch, "The shortest route problem with constraints," *Journal of Math. analysis and applications*, vol. 14, no. 2, pp. 191–197, 1966.

[17] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations research*, vol. 17, no. 1, pp. 36–42, 1992.

[18] D. H. Lorenz and D. Raz, "A simple efficient approximation scheme for the restricted shortest path problem," *Operations Research Letters*, vol. 28, no. 5, pp. 213–219, 2001.

[19] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984.

[20] C.-L. Li, S. T. McCormick, and D. Simchi-Levi, "The complexity of finding two disjoint paths with min-max objective function," *Discrete Applied Mathematics*, vol. 26, no. 1, pp. 105–115, 1990.

[21] Y. Xiao, K. Thulasiraman, and G. Xue, "Constrained shortest link-disjoint paths selection: a network programming based approach," *IEEE Trans. on Circuits and Systems I*, vol. 53, no. 5, pp. 1174–1187, 2006.

[22] L. Guo and H. Shen, "Efficient approximation algorithms for computing k-disjoint minimum cost paths with delay constraint," in *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2012 13th International Conference on*, 2012, pp. 627–631.

[23] S. Ahuja, M. Krunz, and T. Korkmaz, "Optimal path selection for minimizing the differential delay in Ethernet-over-SONET," *Computer Networks*, vol. 50, no. 13, pp. 2349–2363, 2006.

[24] A. Srivastava, S. Acharya, M. Alicherry, B. Gupta, and P. Risbood, "Differential delay aware routing for Ethernet over SONET/SDH," in *24th IEEE INFOCOM*, vol. 2, 2005, pp. 1117–1127.

[25] A. Srivastava, "Flow aware differential delay routing for next-generation ethernet over SONET/SDH," in *IEEE ICC*, vol. 1, 2006, pp. 140–145.

[26] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, pp. 125–145, 1974.

[27] R. E. Korf, E. L. Schreiber, and M. D. Moffitt, "Optimal sequential multi-way number partitioning," in *International Symposium on Artificial Intelligence and Mathematics (ISAIM-2014)*, 2013.

[28] B. Wu, P.-H. Ho, J. Tapolcai, and P. Babarczi, "Optimal allocation of monitoring trails for fast SRLG failure localization in all-optical networks," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010, pp. 1–5.

[29] D. Karger, R. Motwani, and G. D. S. Ramkumar, "On approximating the longest path in a graph," *Algorithmica*, vol. 18, no. 1, pp. 82–98, 1997.

[30] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proc. INOC*, 2007.

[31] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.