International Scholarly Research Network
ISRN Communications and Networking
Volume 2012, Article ID 932456, 19 pages
doi:10.5402/2012/932456

Review Article

An Overview of Algorithms for Network Survivability

F. A. Kuipers

Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands

Correspondence should be addressed to F. A. Kuipers, F.A.Kuipers@tudelft.nl

Received 4 September 2012; Accepted 25 September 2012

Academic Editors: H. Kubota and M. Listanti

Network survivability—the ability to maintain operation when one or a few network components fail—is indispensable for present-day networks. In this paper, we characterize three main components in establishing network survivability for an existing network, namely, (1) determining network connectivity, (2) augmenting the network, and (3) finding disjoint paths. We present a concise overview of network survivability algorithms, where we focus on presenting a few polynomial-time algorithms that could be implemented by practitioners and give references to more involved algorithms.

1. Introduction

Given the present-day importance of communications systems and infrastructures in general, networks should be designed and operated in such a way that failures can be mitigated. Network nodes and/or links might for instance fail due to malicious attacks, natural disasters, unintentional cable cuts, planned maintenance, equipment malfunctioning, and so forth. Resilient, fault tolerant, survivable, reliable, robust, and dependable, are different terms that have been used by the networking community to capture the ability of a communications system to maintain operation when confronted with network failures. Unfortunately, the terminology has overlapping meanings or contains ambiguities, as pointed out by Al-Kuwaiti et al. [1]. In this paper, we will use the term *survivable networks* to refer to networks that, when a component fails, may "survive" by finding alternative paths that circumvent the failed component. Three ingredients are needed to reach survivability.

(1) *Network connectivity*, that is, the network should be well connected (connectivity properties are discussed in Section 1.1).

(2) *Network augmentation*, that is, new links may need to be added to increase the connectivity of a network.

(3) *Path protection*, that is, a procedure to find alternative paths in case of failures.

These three ingredients will be explained in the following sections.

1.1. Network Connectivity. A network is often represented as a graph $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of N nodes (which for instance represent routers) and $\mathcal{L}$ is the set of L links (which for instance represent optical fiber lines or radio channels). Links may be characterized by weights representing for instance their capacity, delay, length, cost, and/or failure probability. A graph is said to be connected if there exists a path between each pair of nodes in the graph, else the graph is said to be disconnected. In the context of survivability, the notion of connectivity may be further specified as k-connectivity, where at least k disjoint paths exist between each pair of nodes. Depending on whether these paths are node or link disjoint, we may discriminate between node and link connectivity. The link connectivity $\lambda(G)$ of a graph G is the smallest number of links whose removal disconnects G. Correspondingly, the node connectivity $\kappa(G)$ of a graph is the smallest number of nodes whose removal disconnects G.

In 1927, Menger provided a theorem [2]—in German—that could be interpreted as follows.

Theorem 1 (Menger's theorem). *The maximum number of link/node-disjoint paths between A and B is equal to the minimum number of links/nodes that would separate A and B.*

Menger's theorem clearly relates to the k link/node connectivity of a graph, in the sense that a k link/node-connected graph has at least k link/node-disjoint paths between any pair of nodes in the graph. The minimum number of links/nodes separating two nodes or sets of nodes is referred to as a minimum cut. In order to assess the link/node connectivity of a network, we therefore need to find its minimum cut.

A somewhat less intuitive notion of connectivity stems from the spectrum of the Laplacian matrix of a graph and is denoted as algebraic connectivity. The algebraic connectivity was introduced by Fiedler in 1973 [3] and is defined as follows.

Definition 2 (algebraic connectivity). The algebraic connectivity equals the value of the second smallest eigenvalue of Q, where Q is the Laplacian matrix $Q = \Delta - A$, with A an $N \times N$ adjacency matrix with elements $a_{ij} = 1$ if there is a link between nodes i and j, else $a_{ij} = 0$, and $\Delta = \text{diag}(d_1, \ldots, d_N)$ an $N \times N$ diagonal matrix with d_j the degree of node j.

The algebraic connectivity has many interesting properties that characterize how strongly a graph G is connected (e.g., see [4, 5]). Moreover, the multiplicity of the smallest eigenvalue (of value 0) of the Laplacian Q is equal to the number of components in the graph G. Hence, if the algebraic connectivity $\alpha(G)$ is larger than 0, the network is connected, else the algebraic connectivity is 0, and the network is disconnected. We have that

$$\alpha(G) \leq \kappa(G) \leq \lambda(G) \leq \delta(G), \tag{1}$$

where $\delta(G)$ is the minimum degree in the network. For ease of notation, when G is not specified, we use α, κ, λ, and δ.

Connectivity properties, may be less obvious when applied to multilayered networks [6, 7], like IP over WDM networks, where a 2-link-connected IP network operated on top of an optical WDM network, with multiple IP links sharing (e.g., groomed on) the same WDM link, could still be disconnected by a single-link failure at the optical layer.

In probabilistic networks, links and/or nodes x are available with a certain probability p_x, which is often computed as $p_x = \text{MTTF}(x)/(\text{MTTF}(x) + \text{MTTR}(x))$, with $\text{MTTF}(x)$ the mean time to failure of x and $\text{MTTR}(x)$ the mean repair time of x. Often the term network availability is used to denote the probability that the network is connected (e.g., see [8]). When the node probabilities are all one and all the link probabilities are independent and of equal value p, then a reliability polynomial (a special case of the Tutte polynomial, e.g., see [9]) is a polynomial function in p that gives the probability that the network remains connected after its links fail with probability p.

1.2. Network Augmentation. The outcome of testing for network connectivity could be that the network is not sufficiently robust (connected). Possibly, rewiring the (overlay) network could improve its robustness properties [10]. However, this is more involved when applied to the physical network, and improving network performance or network robustness is therefore often established by adding new links and possibly also nodes to the network. Adding links or nodes can be costly (which could be reflected by link/node weights), and the new links/nodes should therefore be placed wisely, such that the desired network property is obtained with the fewest amount of links/nodes or such that the addition of a fixed amount of links/nodes maximizes the desired network property. This class of problems is referred to as (network) augmentation problems, and within this class the problems only differ in their objectives. For instance, k-connectivity is an important property in the context of network robustness, and reaching it through link additions might be one such objective. The alternative objective of algebraic connectivity augmentation leads to an NP-hard problem [11]. Similarly, adding a minimum amount of links to make a graph chordal is also NP-hard [12] (a graph is chordal if each of its cycles of four or more nodes has a link connecting two nonadjacent nodes in the cycle).

1.3. Path Protection. Network protocols like OSPF are deployed in the internet to obtain a correct view of the topology and in case of changes (like the failure of a link) to converge the routing towards the new (perturbed) situation. Unfortunately, this process is not fast, and applications may still face unacceptable disruptions in performance. In conjunction with MPLS, an MPLS fast reroute mechanism can be used that, as the name suggests, provides the ability to switch over in subsecond time from a failed primary path to an alternate (backup) path. This fast reroute mechanism is specified in RFC 4090 [13], May 2005, and has already been implemented by several vendors. The concept has also been extended to pure IP networks and is referred to as IP fast reroute [14]. RFC 4090 defines RSVP-TE extensions to establish backup label-switched path (LSP) tunnels for local repair of LSP tunnels. The backup path can either be configured to protect against a link or a node failure. Since the backup paths are precomputed, no time is lost in computing backup paths or performing signalling in the event of a failure. The fast reroute mechanism as described in RFC 4090 assumes that MPLS (primary and backup) paths are computed and explicitly routed by the network operator. Hence, there is a strong need for efficient algorithms to compute disjoint paths.

Depending on whether backup paths are computed before or after a failure of the primary path, survivability techniques can be broadly classified into restoration or protection techniques.

(i) *Protection scheme*: protection is a proactive scheme, where backup paths are precomputed and reserved in advance. In 1 : 1 protection, traffic is rerouted along the backup path upon the failure of the primary path. In 1+1 protection, the data is duplicated and sent concurrently over the primary and backup paths.

(ii) *Restoration scheme*: restoration is a reactive mechanism that handles a failure after it occurs. Thus, the backup path is not known *a priori*. Instead, a backup path is computed only after the failure in the primary path is sensed.

In general, protection has a shorter recovery time since the backup path is precomputed, but it is less efficient in terms of capacity utilization and less flexible. Restoration, on the other hand, provides increased flexibility and efficient resource utilization, but it may take a longer time for recovery, and there is no guarantee that a backup path will be found. As a compromise between the two schemes, Banner and Orda [15] considered designing a low-capacity backup network (using spare capacity or by installing new resources) that is fully provisioned to reroute traffic on the primary network in case of a failure. The backup network itself is not used to transport "primary" traffic. Backup networks with specific topological features have also been addressed in the literature, for instance protection [16] and preconfigured [17] cycles or redundant trees [18].

Depending on how rerouting is done after a failure in the primary path, there are three categories of survivability techniques.

(i) *Path-based protection/restoration*: in path-based protection, a link- or node-disjoint backup path is precomputed and takes over when the primary path fails. In path-based restoration, a new path is computed between the source and destination nodes of the failed path. If such a backup path cannot be found, the request is blocked.

(ii) *Link-based protection/restoration*: in link-based protection, each link is preassigned a local route that is used when the link fails, and in link-based restoration, the objective is to compute a detour between the two ends of the failed link for all paths that are using the link. Since link-based protection/restoration requires signaling only between the two ends of the failed link, it has a smaller recovery time than path-based protection/restoration, which requires end-to-end signaling between the source and destination nodes.

(iii) *Segment-based protection/restoration*: the segment-based scheme (e.g., see [19]) is a compromise between path-based and link-based schemes. Thus, in segment-based protection, backup routes are precomputed for segments of the primary path. In segment-based restoration, a detour of the segment containing the failed link is computed following a failure.

Depending on whether sharing of resources is allowed among backup paths, protection schemes can be of two types:

(i) *Dedicated protection*: in this scheme, resources (e.g., links, wavelength channels, etc.) are not shared among backup paths and are exclusively reserved for a given path request.

(ii) *Shared protection*: in this scheme, backup paths may share resources as long as their primary paths do not share links. In $M:N$ protection, M backup paths are used to protect N primary paths. The shared scheme provides a better resource utilization; however, it is more complicated and requires more information, such as the shareability of each link.

In general, path protection requires less capacity than link protection, while shared protection requires less capacity than dedicated protection. However, path protection is more vulnerable to multiple link failures than link protection, and so is shared protection compared to dedicated protection.

1.4. Paper Outline and Objective. The remainder of this paper is structured as follows. In Section 2, we give an overview of several methods for determining the connectivity properties of a network. In case a network is found to be insufficiently connected from a survivability perspective, links may have to be added to increase the connectivity. In Section 3, we list key results in network connectivity augmentation. Once a network is designed to withstand some failures, proper path protection/restoration schemes should be in place that can quickly defer traffic to alternate routes in case of a failure. In Section 4, we survey work on finding disjoint paths in a network. We conclude in Section 5.

Throughout the paper, the objective is not to list and explain all the relevant algorithms. Rather, we aim to briefly explain some fundamental concepts and some polynomial-time algorithms that could easily be deployed by practitioners or which can be (and have been) used as building blocks for more advanced algorithms, and to provide pointers to further reading.

2. Determining Network Connectivity

In Section 1.1, we indicated that Menger's theorem implies that finding a minimum cut corresponds to finding the connectivity of a network. In this section, we will look further at finding cuts in a network.

Definition 3 (link (edge) cut). A link cut refers to a set of links whose removal separates the graph into two disjoint subgraphs, and where all links in the removed cut-set have an end-point in both subgraphs.

The two subgraphs need not be connected themselves.

Definition 4 (node (vertex) cut). A node cut refers to a set of nodes whose removal separates the graph into two disjoint subgraphs, and where all nodes in the removed cut-set have at least one adjacent link to both subgraphs.

Definition 5 (minimum link/node cut). A minimum cut is a cut whose cardinality is not larger than that of any other cut in the network.

Definitions for a cut also have a variant in which a source node s and a terminating node t need to be separated.

Definition 6 (s-t cut). An s-t cut refers to a cut that separates two nodes s and t in the graph such that both belong to different subgraphs.

(1) $f(u,v) \leftarrow 0\ \forall (u,v) \in \mathcal{L}$ and $f^* \leftarrow 0$ /*Initialize to zero flow*/
(2) While /*loop until the algorithm terminates in line 9*/
(3) For all nodes $u \in \mathcal{N}$, compute in G_f the hopcount $h(u)$ to t /*by Breadth-First-Search [28] from t*/
(4) Compute a blocking flow f in G_f thereby skipping links (u,v) for which $h(v) \neq h(u)+1$
(5) If f exists
(6) $f^* \leftarrow f^* + f$
(7) Update G_f
(8) else
(9) return f^*

Algorithm 1: Dinitz-Max-Flow (G, c, s, t).

Often, when referring to a cut, a link cut is meant. In the remainder of this paper, we will use the same convention and only specify the type of cut for node cuts.

Definition 7 (maximum cut). A maximum cut is a cut whose cardinality is not exceeded by that of any other cut in the network.

Definition 8 (sparsest cut). The sparsest cut (sometimes also referred to as the (Cheeger) isoperimetric constant) is a cut for which the ratio of the number of links in the cut-set divided by the number of nodes in the smaller subgraph is not larger than that of any other cut in the network.

Finding a maximum or sparsest cut is a hard problem (the maximum-cut problem is APX-hard [20] and the sparsest-cut problem is NP-hard [21, 22]), but fortunately a minimum cut, and consequently the network's connectivity, can be computed in polynomial time as will be indicated below. The algebraic connectivity α could be used to approximate the sparsest cut γ as $\alpha/2 \leq \gamma \leq \sqrt{\alpha(2\delta - \alpha)}$ [4, 21]. Dinh et al. [23] investigated the notion of pairwise connectivity (the number of connected pairs, which bears similarities to the sparsest-cut problem), and proved that finding the smallest set of nodes/links whose removal degrades the pairwise connectivity to certain degree is NP-complete.

2.1. Determining Link Connectivity. In the celebrated paper from Ford and Fulkerson [24] (and independently by Elias et al. [25]) a maximum flow from a source s to a terminal t in a network, where the links have a given capacity, is shown to be equal to the minimum-weight s-t link cut in that network, where the weight of the cut is the sum of the capacities of the links in the cut-set; the so-called *max-flow min-cut theorem*. By using a max-flow algorithm and setting the capacity of all links to 1, one can therefore compute the minimum s-t link cut, or the minimum link cut when repeated over all possible s-t pairs. It is not our goal to overview all maximum-flow algorithms (an excellent discourse of the subject is presented in the book by Ahuja et al. [26]), but we will present Dinitz's algorithm, which can be used to determine the minimum s-t link cut in $O(L \cdot (\min\{N^{2/3}, \sqrt{L}\}))$ time. We will subsequently present the algorithm of Matula for determining the minimum link cut in $O(NL)$ time.

2.1.1. Dinitz' Algorithm. Dinitz' algorithm, published in 1970 by Yefim Dinitz, was the first maximum-flow algorithm to run in polynomial time (contrary to the pseudopolynomial running time of the Ford-Fulkerson algorithm [24]). The algorithm is sometimes referred to as Dinic's or Dinits' algorithm, and also different variants are known. A historical perspective of the different variants is presented by Dinitz himself in [27]. In order to describe Dinitz' algorithm, as presented in Algorithm 1, some definitions are given.

Definition 9. The residual capacity $c_f(u,v)$ of a link (u,v) is interpreted in two directions as follows:

$$c_f(u,v) = c(u,v) - f(u,v), \qquad c_f(v,u) = f(u,v), \tag{2}$$

where the flow $f(u,v)$ over a link (u,v) cannot exceed the capacity $c(u,v)$ of that link.

Definition 10. The residual graph G_f of G is the graph in which a directed link (u,v) exists if $c_f(u,v) > 0$.

Definition 11. A blocking flow f_b is an s-t flow such that any other s-t flow f would have to traverse a link already saturated by f_b.

A blocking flow could be obtained by repeatedly finding (via Depth-First-Search [28]) an augmenting flow along an s-t path (or pruning the path from the graph in unit-capacity networks). In unit-capacity networks, the algorithm runs in $O(L \cdot (\min\{N^{2/3}, \sqrt{L}\}))$, which therefore also is the time complexity to determine a minimum s-t link cut with Dinitz' algorithm (for unit node capacities, a complexity of $O(L\sqrt{N})$ can be obtained [29]).

For further reference, in Table 1, we present some key achievements in computing minimum s-t link cuts.

2.1.2. Matula's Algorithm. In this section, we describe the algorithm from Matula [43] for determining the link connectivity of an undirected network. Matula's algorithm is based on the following lemma.

Table 1: Related work on computing minimum s-t link cuts.

Year[1]	Reference	Complexity	Description
1951	Dantzig [30]	$O(N^2LU)$	Linear programming, where U is the largest link capacity.
1956	Ford and Fulkerson [31]	$O(NLU)$	Augmenting paths.
1970	Dinitz [27]	$O(N^2L)$, $O\left(L \cdot \left(\min\left\{N^{2/3}, \sqrt{L}\right\}\right)\right)$	Resp. capacitated and unit-capacity graphs. Shortest augmenting paths.
1974	Karzanov [32]	$O(N^3)$	Preflow-push (A simplification of Karzanov's algorithm has been presented by Tarjan [33]).
1980	Galil and Naamad [34]	$O(NL\log^2 N)$	Extension of Dinitz' algorithm.
1982	Shiloach and Vishkin [35]	$O((N^3 \log N)/p)$	Parallel algorithm for $p \leq N$ processors.
1983	Sleator and Tarjan [36]	$O(NL\log N)$	Dynamic tree data structure.
1986	Goldberg and Tarjan [37]	$O(NL\log(N^2/L))$	Highest-label preflow-push.
1987	Ahuja et al. [38]	$O\left(NL\log\left((N/L)\sqrt{\log U}+2\right)\right)$	Excess scaling.
1989	Cheriyan and Hagerup [39]	$O(NL + N^2\log^3 N)$	Randomized algorithm.
1990	Alon [40]	$O(NL)$	Deterministic version of Cheriyan and Hagerup's randomized algorithm.
1998	Goldberg and Rao [41]	$O\left(\min\left\{N^{2/3}, \sqrt{L}\right\} L\log(N^2/L)\log U\right)$	Length function.
2011	Christiano et al. [42]	$\widetilde{O}(LN^{1/3}\epsilon^{-11/3})$, $\widetilde{O}(L + N^{4/3}\epsilon^{-8/3})$	Resp. $(1-\epsilon)$ and $(1+\epsilon)$ approximation, where $\widetilde{O}(f(x))$ denotes $O(f(x)\log^c f(x))$ for some constant c.

[1] Throughout the paper, we take the convention of listing the year of the first (conference) publication, while referring to the extended (journal) version there where applicable.

Lemma 12. *Let G be a graph with a minimum cut of size $\lambda(G) \leq \delta(G)-1$ that partitions the graph G into two subgraphs $G_1(\mathcal{N}_1, \mathcal{L}_1)$ and $G_2(\mathcal{N}_2, \mathcal{L}_2)$, then any dominating set S of G contains nodes of both G_1 and G_2 (a dominating set $S \in \mathcal{N}$ is a subset of the nodes in G, such that every node in $\mathcal{N}$ is either in S or adjacent to a node in S).*

Proof. For subgraph G_i, $i = 1, 2$, holds that the sum of the nodal degrees in G_i is bounded by

$$N_i(N_i - 1) + \lambda(G) \geq \sum_{u \in G_i} d(u) \geq N_i\delta(G). \tag{3}$$

The upper bound occurs if all nodes in G_i are connected to each other and some of the nodes have a link that is part of the cut-set. The lower bound stems from each node having a degree larger or equal than the minimum degree $\delta(G)$. From the bounds in (3), we can derive that

$$(N_i - \delta(G))(N_i - 1) \geq \delta(G) - \lambda(G). \tag{4}$$

Since $\lambda(G) \leq \delta(G) - 1$ is assumed, $(N_i - \delta(G))(N_i - 1) \geq 1$ and consequently both terms on the left-hand side cannot be smaller than 1. Hence, $N_i - \delta(G) \geq 1$, which means that, under the assumption that $\lambda(G) \leq \delta(G) - 1$, there is at least one node in G_1 that does not have a neighbor in G_2 (and vice versa). In other words, any dominating set S of G should contain nodes of both G_1 and G_2. □

The algorithm of Matula (see Algorithm 2) starts with a node of minimum degree (e.g., node s in G_1) and gradually builds a dominating set S by adding nodes not yet part of or adjacent to the growing set. Since at one point a node, for example u^*, from G_2 needs to be added, keeping track of the minimum cut between newly added dominating nodes, and S will result in finding the overall minimum cut. The algorithm is presented below.

In the algorithm of Matula, an augmenting path is a path in the residual network, where a residual network is the network that remains after pruning the links of a previous augmenting path. There are no 1-hop paths from u^* to S, because then $u^* \in T$. If u^* has n_T neighbors that belong to T, then there exist n_T 2-hop paths from u^* to S, for which either the first hop from u^* to T or the second hop from T to S is part of the minimum cut. These n_T paths form the first augmenting paths, after which $\lambda(G) - n_T$ remains. These remaining augmenting paths can be found in $O(L)$ time each and since there are at most $d(u^*) - n_T$ such paths, the complexity of the algorithm is bounded by $O(NL)$. Finally, if $\lambda(G) = \delta(G)$, then the initialization guarantees that that value would be found.

For directed multigraphs, Shiloach [44] provided a theorem that is stronger than Menger's theorem, namely.

Theorem 13. *Let G be a directed k-link-connected multigraph, then for all $s_1, \ldots, s_k$, $t_1, \ldots, t_k \in \mathcal{N}$ (not necessarily distinct) there exist link-disjoint paths P_i from s_i to t_i for $i = 1, \ldots, k$.*

We refer to Mansour and Schieber [45] for an $O(NL)$-time algorithm for determining the link connectivity in directed networks.

For further reference, in Table 2 we present some key achievements in computing minimum link cuts.

(1) For a node s of minimum degree set $S \leftarrow \{s\}$, $T \leftarrow \{t \mid t \in \text{adj}(s)\}$, $U \leftarrow \mathcal{N} - S - T$, and $\lambda \leftarrow d(s)$.
(2) While $U \neq \emptyset$
(3) Choose $u \in U$
(4) $n \leftarrow$ The number of shortest augmenting paths from u to S
(5) If $n < \lambda$ then $\lambda \leftarrow n$
(6) Set $S \leftarrow S \cup u$, $T \leftarrow T \cup \{t \mid t \in \text{adj}(u)\}$, followed by $U \leftarrow \mathcal{N} - S - T$

Algorithm 2: Matula-Min-Cut (G).

Table 2: Related work on computing minimum link cuts.

Year	Reference	Complexity	Description
1971	Podderyugin [70]	$O(NL)$	Undirected graphs. Variation of Ford-Fulkerson max-flow algorithm in how augmenting paths of one and two hops are handled.
1971	Tarjan [67]	$O(N + L)$	Testing for 2-link connectivity in undirected graphs via DFS.
1975	Even and Tarjan [29]	$O(N^{5/3}L)$	Application of Dinitz' algorithm.
1986	Karzanov and Timofeev [59]	$O(\lambda N^2)$	Undirected graphs.
1987	Matula [43]	$O(NL)$, $O(\lambda N^2)$	Undirected graphs. It is also shown that the maximum subgraph link connectivity can be determined in $O(N^2L)$.
1989	Mansour and Schieber [45]	$O(NL)$, $O(\lambda^2 N^2)$	Directed graphs. Relation between minimum cut and dominating set.
1990	Nagamochi and Ibaraki [71]	$O(L + \lambda N^2)$	Undirected graphs. Algorithm does not use a max-flow algorithm.
1991	Galil and Italiano [72]	$O(N + L)$	Testing for 3-link-connectivity in undirected graphs.
1991	Gabow [73]	$O(\lambda L \log(N^2/L))$, $O(L + \lambda^2 N \log(N/\lambda))$	Directed, resp. undirected graphs. Matroid approach.
1996	Karger [74]	$O\left(L \log^2 N\right)$	Randomized algorithm.

2.2. Determining Node Connectivity. Maximum-flow algorithms can also be used to determine the node connectivity, as demonstrated by Dantzig and Fulkerson [46] (and also discussed in [47]), by transforming the undirected graph $G(\mathcal{N}, \mathcal{L})$ to a directed graph $G'(\mathcal{N}', \mathcal{L}')$ as follows.

For every node $n \in \mathcal{N}$ place two nodes n' and n'' in $\mathcal{N}'$ and connect them via a directed link (n', n''), using the convention that the link starts at n' and ends at n''. For every undirected link $(i, j) \in \mathcal{L}$ place directed links (i'', j') and (j'', i') in G'. All links are assigned unit capacity.

The s-t node connectivity in G can be computed by finding a maximum flow from s'' to t' in G'. This can be seen as follows. Assume that there are κ node-disjoint paths between s and t, then there are also κ corresponding node-disjoint paths from s'' to t' in G'. Since each link has unit capacity, there thus exists a flow of at least κ. Since each link entering a node $n' \in G'$ has to traverse a single unit-capacity link (n', n'') at most one unit of flow can pass through a node, which corresponds to a node-disjoint path. Since there are only κ node-disjoint paths, the maximum flow in G' is equal to κ.

By using Dinitz' algorithm, one may compute the s-t node connectivity in $O(L \cdot (\min\{N^{2/3}, \sqrt{L}\}))$ time, and by using the algorithm of Mansour and Schieber [45], the node connectivity can be determined in $O(NL)$ time. We refer to Henzinger et al. [48] and Gabow [49] for more advanced algorithms to compute the node connectivity in directed and undirected graphs and to Yoshida and Ito [50] for a κ-node-connectivity property testing algorithm (in property testing the objective is to decide, with high probability, whether the input graph is close to having a certain property. These algorithms typically run in sub-linear time).

3. Network Connectivity Augmentation

In the previous section, we have provided an overview of several algorithms to determine the connectivity of a network. In this section, we will overview several network augmentation algorithms that can be deployed to increase the connectivity (or some other metric) of a network by adding links. Network augmentation problems seem closely related to network deletion problems (e.g., see [51]), where the objective is to remove links in order to reach a certain property. However, there may be significant differences in terms of complexity. For instance, finding a minimum-weight set of links to cut a λ-link-connected graph such that its connectivity is reduced to $\lambda = 0$ is solvable in polynomial time (as discussed in Section 2.1), while adding a minimum-weight set of links to increase a disconnected graph to λ-link-connectivity is NP-complete as shown in Section 3.1. When

both link deletions and link additions are permitted, we speak of link modification problems, for example, see [52].

3.1. Link Connectivity Augmentation. In this section, we consider the following link augmentation problem.

Problem 1 (the link connectivity augmentation (LCA) problem). Given a graph $G(\mathcal{N}, \mathcal{L})$ consisting of N nodes and L links, link connectivity λ and an integer β, the link connectivity augmentation problem is to add a minimum-weight set of links, such that the link connectivity of the graph G is increased from λ to $\lambda + \beta$.

We can discriminate several variants based on the graph (directed, simple, planar, etc.) or if link weights are used or not (i.e., in the unweighted case all links have weight 1). Let us start with the weighted link connectivity augmentation problem.

Theorem 14. *The weighted LCA problem is NP-hard.*

We will use the proof due to Frederickson and JáJá [53] to show that the 3-dimensional matching (3DM) problem is reducible to the weighted LCA problem (an earlier proof has been provided by Eswaran and Tarjan [54], but since it aims to augment a network without any links to one that is 2 connected and has N links (a cycle), it has the characteristics of a design rather than an augmentation problem).

Problem 2 (3-dimensional matching (3DM)). Given a set $M \subseteq X \times Y \times Z$ of triplets, where X, Y, and Z are disjoint sets of q elements each, is there a matching subset $M' \subseteq M$ that contains all $3q$ elements, such that $|M'| = q$, and thus no two elements of M' agree in any coordinate?

Proof. For a 3DM instance $M \subseteq X \times Y \times Z$, with $|M| = p$, $X = \{x_i \mid i = 1, \ldots, q\}$, $Y = \{y_i \mid i = 1, \ldots, q\}$, and $Z = \{z_i \mid i = 1, \ldots, q\}$, we create the graph $G(\mathcal{N}, \mathcal{L})$ of the corresponding instance of the weighted LCA problem as follows:

$$\begin{aligned} \mathcal{N} &= \{r\} \cup \{x_i, y_i, z_i \mid i = 1, \ldots, q\} \\ &\quad \cup \left\{a_{ijk}, \overline{a}_{ijk} \mid \left(x_i, y_j, z_k\right) \in M\right\}, \\ \mathcal{L} &= \{(r, x_i), (r, y_i), (r, z_i) \mid i = 1, \ldots, q\} \\ &\quad \cup \left\{\left(x_i, a_{ijk}\right), \left(x_i, \overline{a}_{ijk}\right) \mid \left(x_i, y_j, z_k\right) \in M\right\}. \end{aligned} \tag{5}$$

The graph G as constructed above forms a tree and therefore is 1 connected. Links from the complement G^c of G can be used to augment the graph to 2-link connectivity. The weights of the links in $G^c(\mathcal{N}, \mathcal{L}^c)$ are $w(a_{ijk}, \overline{a}_{ijk}) = w(y_j, a_{ijk}) = w(z_k, \overline{a}_{ijk}) = 1$ for $(x_i, y_j, z_k) \in M$, and for the remaining links in G^c, the weight is 2.

M contains a matching M' if and only if there is a set $\mathcal{L}' \subseteq \mathcal{L}^c$ of weight $w(\mathcal{L}') = p + q$ such that $G'(\mathcal{N}, \mathcal{L} \cup \mathcal{L}')$ is 2-link connected. Assuming M' exists, then adding links (y_j, a_{ijk}) and $(z_k, \overline{a}_{ijk})$ for each triple $(x_i, y_j, z_k) \in M'$ will establish the (2-connected) cycle $r - y_j - a_{ijk} - x_i - \overline{a}_{ijk} - z_k - r$. Since $|M'| = q$, the weight of these added links is $2q$. The remaining nodes that are not yet on a cycle are the nodes a_{ijk} and $\overline{a}_{ijk}$ belonging to $(x_i, y_j, z_k) \in \{M - M'\}$. These nodes will be directly connected, thereby creating the cycle $x_i - a_{ijk} - \overline{a}_{ijk} - x_i$. In total, $|M - M'| = p - q$ additional links will be added, leading to a total weight of links that have been added of $p+q$. Since the graph G is a tree with $2(p+q)$ leaves and the minimum link weight is 1, a network augmentation solution of weight $p + q$ is indeed the lowest possible. It remains to demonstrate that an augmentation of weight $p+q$ will lead to a valid matching M'. Since, in a solution of weight $p+q$, each leaf will be connected by precisely one link from $\mathcal{L}^c$, a link (y_j, a_{ijk}) will prevent adding a link $(a_{ijk}, \overline{a}_{ijk})$, and therefore also link $(z_k, \overline{a}_{ijk})$ must be added. The corresponding triple (x_i, y_j, z_k) was not augmented before and is, therefore, part of a valid matching. The remaining $p - q$ links $(a_{ijk}, \overline{a}_{ijk})$ do not contribute to the matching. □

Frederickson and JáJá also used the construction of this proof to prove that the node-connectivity and strong-connectivity variants of the weighted LCA problem are NP-hard (in a directed graph strong connectivity is used, which means that there is a directed path from each node to every other node in the graph). We remark that the unweighted simple graph preserving LCA problem was claimed to be NP-hard by Jordán (reproduced in [55]) by using a reduction to one of the problems treated by Frederickson and JáJá. However, Jordán appears to be using an unweighted problem of which only (in the paper [53] referred to) the weighted version is proved to be NP-hard, and it is therefore not clear whether the unweighted problem is indeed NP-hard. For fixed β, the unweighted simple graph preserving problem can be solved in polynomial time [55].

Eswaran and Tarjan [54] were the first to report on augmentation problems. They considered augmenting a network towards either 2-link connectivity, 2-node connectivity or strong connectivity, and provided for each unweighted problem variant an algorithm of complexity $O(N + L)$ (Raghavan [56] pointed out an error in the strong connectivity algorithm and provided a fix for it). Since most protection schemes only focus on protecting against one single failure at a time (by finding two disjoint paths as discussed in Section 4), we will first present the 2-link-connectivity augmentation algorithm of Eswaran and Tarjan [54].

3.1.1. Eswaran and Tarjan Algorithm. The algorithm of Eswaran and Tarjan as presented in Algorithm 6 makes use of preorder (Algorithm 4) and postorder (Algorithm 3) numbering of nodes in a tree T (the label $l(u)$ of node u denotes its number as a result of the ordering) and a procedure (Algorithm 5) to find 2-link-connected components.

We have assumed that the initial graph was connected. Eswaran and Tarjan's algorithm also allows to start with disconnected graphs, by augmenting the forest of condensed 2-link-connected components to a tree.

3.1.2. Cactus Representation of All Minimum Cuts. The algorithm of Eswaran and Tarjan uses a tree representation of all the 2-link-connected components in G, which is

(1) For each $v \leftarrow \text{adj}(u)$ do PostOrder (v, i)
(2) $i \leftarrow i + 1$
(3) $l(u) \leftarrow i$

Algorithm 3: PostOrder (T, u, i).

(1) $i \leftarrow i + 1$
(2) $l(u) \leftarrow i$
(3) For each $v \leftarrow \text{adj}(u)$ do PreOrder (v, i)

Algorithm 4: PreOrder (T, u, i).

subsequently used to find a proper augmentation. By using a so-called cactus representation of all minimum cuts in a network, a similar strategy could be deployed to augment a network to a connectivity >2. A graph G is defined to be a cactus graph if any two distinct simple cycles in G have at most one node in common (or equivalently, any link of G belongs to at most one cycle). In this section, we will present the cactus representation.

We will use the notation (X, Y) to represent a set of links that connect nodes in X to nodes in Y. The link-set $(X, \overline{X})$, with $\overline{X} = \mathcal{N} \setminus X$, refers to a cut-set of links whose removal separates the graph into two subgraphs of nodes X and nodes $\overline{X}$. Dinitz et al. [58] have proposed a cactus structure $\mathcal{H}(G)$ to represent all the minimum cuts of a graph G (possibly with parallel links) and have shown that there can be at most $\binom{N}{2}$ such minimum cuts. The structure $\mathcal{H}(G)$ possesses the following properties.

(1) $\mathcal{H}(G)$ is a cactus graph, that is, any two distinct simple cycles of $\mathcal{H}(G)$ have at most one node in common.

(2) Each proper cut in $\mathcal{H}(G)$ is a minimum cut (a cut is called proper if the removal of the links in that cut partitions the graph in precisely two subgraphs. A minimum cut is always proper).

(3) For any link (u, v) that is part of a cycle in $\mathcal{H}(G)$ the weight $w(u, v) = \lambda/2$, else $w(u, v) = \lambda$.

(4) $w(\lambda(\mathcal{H}(G))) = \lambda(G)$, where $w(\lambda(\mathcal{H}(G)))$ represents the minimum-weight link cut of $\mathcal{H}(G)$.

A cactus graph without cycles is a tree, and if $\lambda(G)$ is odd, then $\mathcal{H}(G)$ is a tree. Cycles in the cactus graph $\mathcal{H}(G)$ reflect so-called crossing cuts in G.

Definition 15. Two cuts $(X, \overline{X})$ and $(Y, \overline{Y})$, with $X, Y \in \mathcal{N}$, are crossing cuts, if all four sets $X \cap Y$, $X \cap \overline{Y}$, $\overline{X} \cap Y$, and $\overline{X} \cap \overline{Y}$ are non-empty.

Karzanov and Timofeev [59] have outlined an algorithm to compute $\mathcal{H}(G)$ that consists of two parts: (1) computing all minimum cuts and (2) constructing the corresponding cactus representation. However, Nagamochi and Kameda [60] reported that their cactus representation may not be unique. We assume that all minimum cuts are already known (e.g., by computing minimum s-t cuts between all possible source-destination pairs, by the Gomory-Hu tree algorithm [61], or with Matula's algorithm as explained in [62]) and focus on explaining—by following the description of Fleischer [63]—how to build a unique cactus graph $\mathcal{H}(G)$ for the graph G.

Karzanov and Timofeev [59] observe that for a link $(i, j) \in \mathcal{L}$, any two minimum cuts $(X, \overline{X})$ and $(Y, \overline{Y})$ that separate i and j are nested, which means that $X \subset Y$ (or vice versa). If we assign the nodes of G a preorder labelling $\{n_1, \ldots, n_N\}$, such that node n_{i+1} is adjacent to a node in the set $\mathcal{N}_i := \{n_1, \ldots, n_i\}$, and define $\mathcal{M}_i$ to be the set of minimum cuts that contain $\mathcal{N}_{i-1}$ but not n_i, then it follows that all cuts in $\mathcal{M}_i$ are noncrossing for each $i \in \{2, \ldots, N\}$. For instance, consider a 4-node ring $\{(a, b), (b, c), (c, d), (d, a)\}$, where three minimum cuts separate nodes a and d, namely, $(\{a\}, \{b, c, d\})$, $(\{a, b\}, \{c, d\})$, and $(\{a, b, c\}, \{d\})$. Clearly $\{a\} \subset \{a, b\} \subset \{a, b, c\}$, which allows us to represent them as a path graph $a - b - c - d$. The three possibilities to cut this chain correspond to the three minimum cuts that separate a and d in the ring graph. For each $\mathcal{M}_i$ there is a corresponding path graph P_i. These $N - 1$ path graphs are used to create a single cactus graph. We proceed to present the algorithm as described by Fleischer [63] (for an alternative description we refer to [64]), see Algorithm 7. We define η to be the function that maps nodes of G to nodes in $\mathcal{H}(G_{i+1})$, and we define G_i to be the graph G with nodes $\mathcal{N}_i$ contracted to a single node (and any resulting self-loops removed). Let G_r be the smallest graph that has a minimum cut of value λ, where r corresponds to the largest index of such a graph. $\mathcal{H}(G_r)$ is a path graph. The algorithm builds $\mathcal{H}(G_i)$ from $\mathcal{H}(G_{i+1})$ until $\mathcal{H}(G_1) = \mathcal{H}(G)$ is obtained.

Figure 1 gives an example of the execution of the algorithm on a 4-node ring.

3.1.3. Naor-Gusfield-Martel Algorithm. Naor et al. [65] have proposed a polynomial-time algorithm to augment the link connectivity of a graph G from λ to $\lambda + \beta$, by adding the smallest number of (possibly parallel) links. The authors first demonstrate how to augment the link connectivity by one in $O(NL)$ time, after which it is explained how executing this algorithm β times could optimally augment the graph towards $\lambda + \beta$ link connectivity (Cheng and Jordán [66] further discuss link connectivity augmentation by adding one link at a time). In practice, as a result of the costs in network augmentation, a network's connectivity is likely not augmented with $\beta > 1$. We will therefore only present the algorithm to augment the link connectivity by one, see Algorithm 9, and refer to [65] for the extended algorithm. The algorithm uses the cactus structure $\mathcal{H}(G)$ that was presented in the previous section to represent all the minimum cuts of a graph G. The algorithm is similar in approach to the Eswaran-Tarjan algorithm, since a cactus representation of a 1-connected network is the tree representation used by Eswaran and Tarjan, and the algorithm connects "leafs" as Eswaran and Tarjan have done. Naor et al., however, use a different definition of leafs for cactus graphs.

(1) Find a directed spanning tree T in G rooted at a node s
(2) PostOrder $(T, s, 0)$
(3) For $i = 1, \ldots, N$
(4) For $j \in \text{adj}(i)$ /*Only nodes "downstream"*/
(5) if $\min(\{i - N_D(i) + 1\} \cup \{N_L(j)\} \cup \{j \mid (i, j) \notin T\}) > j - N_D(j)$ and $\max(\{i\} \cup \{N_H(j)\} \cup \{j \mid (i, j) \notin T\}) \leq j$ /*$N_D(i)$ is the number of descendants in the tree including i*/
(6) then (i, j) is 1-link-connected. /*Its removal cuts the graph*/

Algorithm 5: Tarjan-2-link-components (G), developed by Tarjan [57].

(1) Find the 2-link-connected components of G
(2) Condense G into a tree T for which each node represents one of the 2-link-connected components of G
(3) Number the nodes in T in preorder, starting from an arbitrary non-leaf node s /*PreOrder $(T, s, 0)$*/
(4) For $i = 1, \ldots, \lceil r/2 \rceil$ choose links $(u(i), u(i + \lfloor r/2 \rfloor))$, where $u(i), \ldots, u(r)$ are the r leaves of T ordered in increasing node number
(5) Map the ends of each chosen link to an arbitrary node in the corresponding 2-link-connected component

Algorithm 6: Eswaran-Tarjan-2-link-augmentation (G).

Definition 16 (cactus leaf). A node in a cactus representation $\mathcal{H}(G)$ is a cactus leaf if it has degree 1 or is a cycle node of degree 2.

Similarly to a tree, if the cactus $\mathcal{H}(G)$ has k leafs, then $\lceil k/2 \rceil$ links need to be added to increase the connectivity by 1.

The algorithm uses a Depth-First-Search-like procedure, see Algorithm 8, to label the nodes of the cactus graph.

For further reference, in Table 3, we present some key achievements in augmenting link connectivity in unweighted graphs.

Splitting off a pair of links (u, v) and (v, w) refers to deleting those links and adding a new link (u, w). A pair of links is said to be splittable if the s-t min-cut values remain unaffected after splitting off the pair of links and is considered in the context of Mader's theorem.

Theorem 17 (Mader [68, 69]). *Let G be a connected undirected graph where for some node s the degree $d(s) \neq 3$, and the removal of one of the adjacent links of s does not disconnect the graph, then s has a pair of splittable links.*

Mader's theorem has been used by for instance Cai and Sun [75] and Frank [77] in developing network augmentation algorithms. The algorithms (as already outlined in 1976 by Plesník [84]) attach a new node s to the graph with $(\lambda + \beta)$ parallel links between s and all other nodes in the graph and subsequently proceed to split off splittable links.

As indicated by Theorem 14, the weighted LCA problem is NP-complete for both undirected graphs and directed graphs. Frederickson and JáJá [53] provided an $O(N^2)$ algorithm to make a weighted graph 2 connected. The algorithm is a 2-approximation algorithm if the starting graph is connected, else it is a 3-approximation algorithm. Khuller and Thurimella [85] proposed a 2-approximation algorithm for increasing the connectivity of a weighted undirected graph to $(\lambda+\beta)$ that has a complexity of $O(N(\lambda+\beta)\log N(L+N\log N))$. Taoka et al. [86] compare via simulations several approximation and heuristic algorithms, including their own maximum-weight-matching-based algorithm.

Under specific conditions, the weighted LCA problem may be polynomially solvable, as shown by Frank [77] for the case that link weights are derived from node weights.

3.2. Node Connectivity Augmentation. In this section, we consider the following node augmentation problem.

Problem 3. The Node Connectivity Augmentation (NCA) problem. Given a graph $G(\mathcal{N}, \mathcal{L})$ consisting of N nodes and L links, node connectivity κ and an integer γ, the node connectivity augmentation problem is to add a minimum-weight set of links, such that the node connectivity of the graph G is increased from κ to $\kappa + \gamma$.

Like for the LCA problem.

Theorem 18. *The weighted NCA problem is NP-hard.*

Proof. The proof of Theorem 14 also applies here. □

The unweighted undirected NCA problem has received most attention. The specific case of making a graph 2-node connected was treated by Eswaran and Tarjan [54], Rosenthal and Goldner [87] (a correction to this algorithm has been made by Hsu and Ramachandran [88]). Watanabe

(1) Compute P_i for $i = 2, \ldots, N$
(2) For $i = r - 1, \ldots, 1$
(3) Replace the node q in $\mathcal{H}(G_{i+1})$ that contains nodes $\{n_1, \ldots, n_{i+1}\}$ with the Path P_{i+1}. If $P_{i+1} = X_1, \ldots, X_k$, then remove q and introduce k new nodes $q_1, \ldots, q_k$ with links (q_j, q_{j+1}) for $1 \leq j < k$.
(4) Connect path P_{i+1} to $\mathcal{H}(G_{i+1})$. For any tree or cycle link (q, w) in $\mathcal{H}(G_{i+1})$, let $W \neq \emptyset$ be the set of nodes in w, or if w is an empty node, the nodes in any nonempty node w' reachable from w by some path of links disjoint from a cycle containing (q, w). Find the subset X_j such that $w \subset X_j$ and connect W to q_j.
(5) Label the nodes of P_{i+1}. Let Q be the set of nodes mapped to q in $\mathcal{H}(G_{i+1})$. Update η by $\eta^{-1}(q_j) := X_j \cap Q$ for all $1 \leq j < k$. All other mappings remain unchanged.
(6) Remove all empty nodes of degree ≤ 2 and all empty 2-way cut nodes by contracting an adjacent tree link (a node is an x-way cut node if its removal separates the graph into x connected parts). Replace all empty 3-way cut nodes with 3 cycles.

Algorithm 7: Build-Cactus (G).

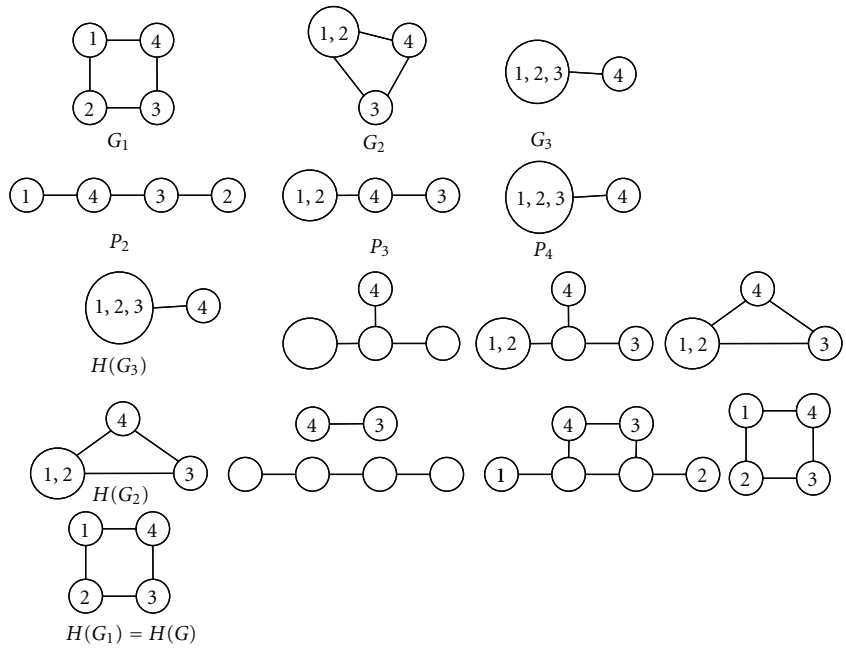

Figure 1: Example of a cactus construction for a 4-node ring topology. The top "row" gives the graphs G_i, for $i = 1, \ldots, r$. The second row gives the paths P_i, for $i = 2, \ldots, N$. The third row presents the first iteration of the for loop in algorithm Build-Cactus, while the fourth row presents the last iteration. The last row presents the cactus representation of the ring, which is a ring itself.

(1) Assign different colors to the different simple cycles /*, for example, by finding the articulation points [67]*/
(2) DFS traversal that starts at an arbitrary node and obeys the following rule: if a node u is visited for the first time via a cycle with some color, then traverse all other differently colored links adjacent to u before traversing the adjacent link of the same color. Enumerate the cactus leafs $u_1, \ldots, u_k$ in the order in which they are first encountered in the DFS traversal.

Algorithm 8: Cactus-DFS ($\mathcal{H}(G)$).

(1) $\mathcal{H}(G) \leftarrow$Build-Cactus (G)
(2) Cactus-DFS $(\mathcal{H}(G))$
(3) Form the pairs $\{(U_i, U_{i+\lceil k/2 \rceil}) : 1 \le i \le \lfloor k/2 \rfloor\}$, where U_i is the set of nodes from G that map to the leaf u_i of $\mathcal{H}(G)$.
(4) For each pair $(U_i, U_{i+\lceil k/2 \rceil}) : 1 \le i \le \lfloor k/2 \rfloor$, add a link between a node from G in U_i and a node from G in $U_{i+\lceil k/2 \rceil}$. If k is odd, then connect a node in $U_{\lceil k/2 \rceil}$ to a node in a different leaf U_j.

Algorithm 9: Naor-Gusfield-Martel-Aug-1 (G).

and Nakamura [89] and Jordán [90] solved the case for achieving 3-node-connectivity, while Hsu [91] developed an algorithm to upgrade a 3-node connected graph to a 4-node-connected one. Increasing the connectivity of a κ-node-connected graph (where κ can be any integer) by 1 was studied by many researchers [90, 92–97], since it was long unknown whether the problem was polynomially solvable. In 2010, Végh [98] provided a polynomial-time algorithm to increase the connectivity of any κ-node-connected unweighted undirected graph by $\gamma = 1$.

Augmenting the node connectivity of directed graphs has been treated by Frank and Jordán [99]. They found a min-max formula that finds the minimum number of required new links to make an unweighted digraph $(\kappa + \gamma)$-node connected. Frank and Végh [100] developed a polynomial-time algorithm to make a κ-node-connected directed graph $(\kappa + 1)$-node connected.

As the weighted NCA problem is NP-complete, special cases have been considered [88–91, 98, 101]. Most of these articles discuss specific connectivity targets (γ and/or κ have specific values) or specific topologies, like trees. Also heuristic and approximation algorithms have been proposed [85, 102–107].

4. Disjoint Paths

When a network is (made to be) robust, algorithms should be in place that can find link- or node-disjoint paths to protect against a link or node failure. There can be several objectives associated with finding link- or node-disjoint paths.

Problem 4. Given a graph $G(\mathcal{N}, \mathcal{L})$, where $|\mathcal{N}| = N$ and $|\mathcal{L}| = L$, a weight $w(u, v)$ and a capacity $c(u, v)$ associated with each link $(u, v) \in \mathcal{L}$, a source node s and a terminal node t, and two bounds $\Delta_1 \ge 0$ and $\Delta_2 \ge 0$ find a pair of disjoint paths from s to t such as the following.

Min-Sum Disjoint Paths Problem. The total weight of the pair of disjoint paths is minimized.

Min-Max Disjoint Paths Problem. The maximum path weight of the two disjoint paths is minimized.

Min-Min Disjoint Paths Problem. The smallest path weight of the two disjoint paths is minimized.

Bounded Disjoint Paths Problem. The weight of the primary path should be less than or equal to Δ_1, and the weight of the backup path should be less than or equal to Δ_2.

Widest Disjoint Paths Problem. The smallest capacity over all links in the two paths is maximized.

The most common and simpler one is the *min-sum* disjoint paths problem. If the two paths are used simultaneously for load-balancing purposes (or 1 + 1 protection), then the min-max objective is desirable. Unfortunately, the min-max disjoint paths problem is NP-hard [108]. If failures are expected to occur only sporadically (and in case of 1 : 1 protection), then it may be desirable to minimize the weight of the primary (shorter) path (min-min objective), which also leads to an NP-hard problem [109]. The min-max and min-min disjoint paths problems could be considered as extreme cases of the bounded disjoint paths problem, which was shown to be NP-hard [110] and later proven to be APX-hard by Bley [111] (the graph structure referred to as lobe that was used by Itai et al. [110] to prove NP-completeness has since often been used to prove that other disjoint paths problems are NP-complete, e.g., [112–114]). Finding widest disjoint paths can easily be done by pruning "low-capacity" links from the graph and finding disjoint paths. When the capacity requirements for the primary and backup paths are different, disjoint paths problems usually become NP-complete [115].

Beshir and Kuipers [116] investigated the min-sum disjoint paths problem with min-max, min-min, bounded, and widest, as secondary objectives in case multiple min-sum paths exist between s and t. From these variants, only the widest min-sum link-disjoint paths problem is not NP-hard.

Li et al. [112] studied the min-sum disjoint paths problem, where the link-weight functions are different for the primary and backup paths and showed that this problem is hard to approximate. Bhatia [117] demonstrated that the problem remains hard to approximate in the case that the weights for the links of the backup path are a fraction $0 < \rho < 1$ of the normal link weights (for the primary path).

Sherali et al. [118] investigated the time-dependent min-sum disjoint paths problem, where the link weights are time-dependent. They proved that the problem is NP-hard, even if only one link is time-dependent and all other links are static.

4.1. Min-Sum Disjoint Paths. Finding min-sum disjoint paths is equivalent to finding a minimum-cost flow in unit-capacity networks [26]: a minimum-cost flow of k

Table 3: Related work on augmenting link connectivity in unweighted graphs.

Year	Reference	Complexity	Description
1976	Eswaran and Tarjan [54]	$O(N+L)$	Augmenting to 2-connectivity.
1986	Cai and Sun [75]	NA	Splitting off links.
1987	Watanabe and Nakamura [76]	$O\left((\lambda+\beta)^2N^4((\lambda+\beta)N+L)\right)$	Based on a derived formula for the minimum number of links to $(\lambda+\beta)$-link-connect G.
1990	Frank [77]	$O(N^6)$	Different s-t connectivities may be specified, instead of one $(\lambda+\beta)$-connectivity for all pairs[1].
1990	Naor et al. [65]	$O(\beta^2NL+\beta^3N^2+NC_{\text{flow}})$	C_{flow} is the complexity of computing a maximum flow. If $\beta=1$, then the complexity is $O(NL)$.
1991	Gabow [78]	$O\left(L+(\lambda+\beta)^2N\log N\right)$	Poset representation of cuts applied to the Naor-Gusfield-Martel algorithm.
1994	Benczúr [79]	$\widetilde{O}(\min\{N^3,\beta N^2\}), O(N^4)$	Resp. randomized and deterministic algorithms.
1996	Nagamochi and Ibaraki [80]	$O(N(L+N\log N)\log N)$	Splitting off links.
1998	Benczúr and Karger [81]	$O(N^2\log^8 N)$	Randomized algorithm.
2004	Nagamochi and Ibaraki [82]	$O(NL+N^2\log N)$	Maximum adjacency ordering[2].

[1] NP-hard variations of this problem and corresponding approximation results are provided by Nutov [83].
[2] Maximum adjacency ordering rule: add a new node n_{i+1} to previously selected nodes $\{n_1,\ldots,n_i\}$ that has the largest number of links to the set $\{n_1,\ldots,n_i\}$. Start with an abitrary node n_1.

will traverse k disjoint paths. In fact, Suurballe's algorithm, which is most often cited as an algorithm to compute two disjoint paths, is an algorithm that uses augmenting paths, like in several max-flow algorithms. The original Suurballe algorithm as presented in [119] allows to compute k node (or link) disjoint paths between a single source-destination pair, by using k shortest path computations. Later, this approach was used by Suurballe and Tarjan [120] to find two link (or node) disjoint paths from a source s to all other nodes in the network (i.e., $N-1$ source-destination pairs), by using only two shortest-paths computations, that is, in $O(N\log N+L)$ time. Both papers focus on directed networks, but can also be applied to undirected networks.

In directed networks, a link-disjoint paths algorithm can be used to compute node-disjoint paths, if we split each node u into two nodes u_1 and u_2, with a directed link (u_1,u_2), and the incoming links of u connected to u_1 and the outgoing links of u departing from u_2.

In undirected networks, a link-disjoint paths algorithm can be used to compute node-disjoint paths by the transformation described in Section 2.2.

We will present the Suurballe-Tarjan algorithm, see Algorithm 10, for computing two link-disjoint paths between s and every other node in the network.

Instead of finding an augmenting path for each source-destination pair, Suurballe and Tarjan have found a way to combine these augmenting flow computations into two Dijkstra-like shortest-paths computations. First a shortest paths tree T is computed in line 1, and based on the computed shortest path lengths, the link weights are modified in line 2. This link weight modification was also used by Suurballe and is to assure that $w(u,v)\geq 0$ for all links, with equality if (u,v) is in T. In Suurballe's original algorithm the direction of the links on the shortest path from s to d was reversed, after which a shortest (augmenting) path in the newly modified graph was computed. In Suurballe-Tarjan's algorithm the links maintain their direction, but an additional parameter q is used instead. The algorithm proceeds in a Dijkstra-like fashion. Lines 3–6 correspond to the initialization of the smallest length $l(i)$ from s to i found so far, its corresponding predecessor list $\pi(i)$ and $q(i)$. The algorithm repeatedly extracts a node of minimum length (in line 8) and removes that node from the tree T (in line 9). A slightly different relaxation procedure is used (lines 10–12). Upon termination of the algorithm, the disjoint paths between the source s and a destination t can be retrieved via the lists $\pi()$ and $q()$ with Algorithm 11.

Taft-Plotkin et al. [121] extended the approach of Suurballe and Tarjan in two ways: (1) they return maximally disjoint paths, and (2) they also take bandwidth into account. Their algorithm, called MADSWIP, computes maximum-bandwidth maximally disjoint paths and minimizes the total weight as a secondary objective. Consequently, by assigning all links equal bandwidth, the MADSWIP algorithm returns the min-sum maximally disjoint paths.

For a distributed disjoint paths algorithm, we refer to the work of Ogier et al. [122] and Sidhu et al. [123].

Roskind and Tarjan [124] presented an $O(L\log L+k^2N^2)$ algorithm for finding k link-disjoint spanning trees of minimum total weight. Xue et al. [125, 126] have considered quality of service and quality of protection issues in computing two disjoint trees (quality of Protection (QoP) as used by Xue et al. refers to the amount of link failures that can be survived. QoP sometimes is used to refer to probabilistic survivability, as discussed in the following section or protection differentiation as overviewed by Cholda et al. [127]). Ramasubramanian et al. [128] proposed a distributed algorithm for computing two disjoint trees. Guo et al. [129] considered finding two link-disjoint paths subject to multiple Quality-of-Service constraints.

```
(1) Compute the shortest paths tree T rooted at s
(2) Modify the weights of each link (u, v) ∈ ℒ to w′(u, v) = w(u, v) − l(s, v) + l(s, u)
    /* l(s, i) is the length of the shortest path in G from s to i* /
(3) For i = 1, . . . , N
(4)         l(i) ← ∞, q(i) ← NIL, π(i) ← NIL
(5) l(s) ← 0
(6) Q ← 𝒩
(7) While Q ≠ ∅
(8)         EXTRACT-MIN (Q) → u
(9)         DELETE (T, u) /* T becomes a forest of subtrees* /
(10)          For each non-tree link (i, j) in T such that i = u or i and j are in different subtrees
(11)              If l(u) + w(i, j) < l(j)
(12)                  l(j) ← l(u) + w(i, j), π(j) ← i, q(j) ← u
```

Algorithm 10: Suurballe-Tarjan-2-link-disjoint-paths (G, s).

```
(1) x ← t
(2) While x ≠ s
(3)         mark x
(4)         x ← q(x)
(5) For i = 1, 2
(6) x ← t
(7) While x ≠ s
(8)         If x is marked
(9)             unmark x
(10)            P_i ← P_i + (p(x), x)
(11)            x ← p(x)
(10)        Else
(11)            P_i ← P_i + (y, x) /* y is parent of x in T* /
(12)            x ← y
```

Algorithm 11: Return-Suurballe-Tarjan-2-link-disjoint-paths (G, s, t).

4.2. Probabilistic Survivability. When two disjoint primary and backup paths are reserved for a connection, any failure on the primary path can be survived by using the backup path. The backup path therefore provides 100% survivability guarantee against a single failure. When no backup paths are available, that is, unprotected paths are used, then the communication along a path will fail if there is a failure on that path. Banner and Orda [130] have introduced the term p-survivable connection to denote a connection for which there is a probability $\geq p$ that all its links are operational (the related notion of Quality of Protection (QoP), as defined by Gerstel and Sasaki [131], was argued to be difficult to apply to general networks). The previous two cases correspond to $p = 1$ and $p = 0$, respectively. Banner and Orda proved that, under the single-link failure model, at most two paths are needed to establish a p-survivable connection, if it exists. Based on this observation, they studied and proposed algorithms for several problem variants, namely establishing p-survivable-B-bandwidth, most survivable, and widest p-survivable connections for 1 : 1 and 1 + 1 protection architectures (the MADSWIP algorithm [121] can also be used to find the most survivable connection). The p-survivable-B-bandwidth problem asks for a connection with survivability $\geq p$ and bandwidth $\geq B$ and solving it provides a foundation for solving the other problems. We will therefore discuss the solution proposed by Banner and Orda for the p-survivable-B-bandwidth problem.

The approach by Banner and Orda to solve the p-survivable-B-bandwidth problem is twofold. First, the graph is transformed, after which a minimum-cost flow is found on the resulting graph. The graph transformation is depicted in Figure 2 and slightly differs for the 1 : 1 and 1 + 1 cases. Clearly, if a link does not have sufficient spare capacity to accommodate the requested bandwidth B, then it does not need to be considered further (Figure 2(a)). If, for 1 : 1 protection, $b_e \geq B$, then there is sufficient bandwidth for both disjoint paths, since the backup path is only used after failure of the primary path. To allow for both paths to share that link, it is transformed to two links (Figure 2(b)). If the original link is only used by one path, then that link is protected, and hence the weight 0 is assigned to the top link. If both paths have to use the original link, then the connection's survivability is affected by the failure probability of that link, which is why the weight $-\ln(1 - p_e)$ is assigned to the lower link (the logarithm is used to transform a multiplicative metric to an additive metric).

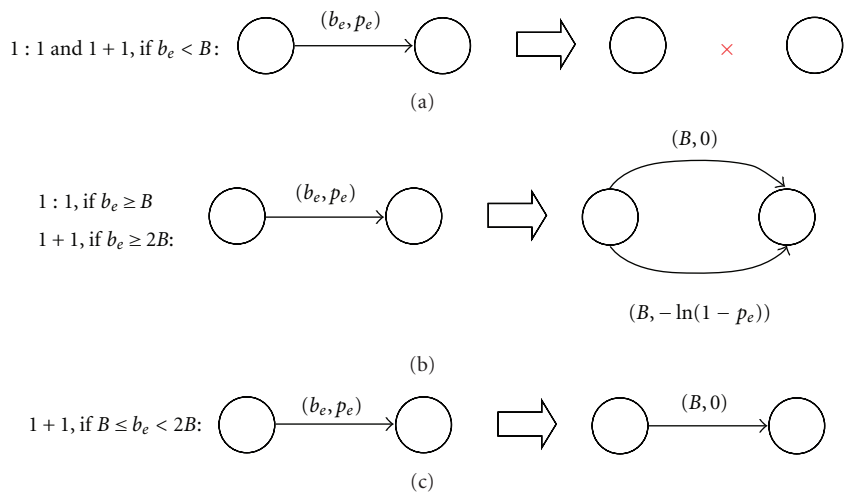

Figure 2: Graph transformation for the p-survivable-B-bandwidth problem. For each link e with capacity b_e and failure probability p_e, the new links consist of a bandwidth-weight pair.

The same applies to the 1 + 1 case, with the exception that the concurrent transmission of data over both paths requires twice the requested bandwidth. For the remaining range $B \leq b_e < 2B$ for 1 + 1 protection (Figure 2(c)), it holds that only one of the paths can use that link, which is why there is no weight penalty.

In the transformed graph, a minimum-cost flow of $2B$ units corresponds to two maximally disjoint paths of each B bandwidth. The minimum-cost flow could, for instance, be found with the cycle-canceling algorithm of Goldberg and Tarjan [132], while the corresponding maximally disjoint paths could be returned via a flow-decomposition algorithm [26].

Luo et al. [133] studied the min-sum p-survivable connection problem, where each link is characterized by a weight and a failure probability, and the problem is to find a connection of least weight and survivability $\geq p$. Contrary to the min-sum maximally-disjoint paths problem, this problem is NP-hard, since it contains the NP-hard restricted shortest paths problem (e.g., see [134]). Luo et al. proposed an ILP and two approximation algorithms for this problem. Chakrabarti and Manimaran [135] studied the min-sum p-survivable-B-bandwidth problem, for which they considered a segment-based protection strategy.

She et al. [114] have considered the problem of finding two link-disjoint paths, for which the probability that at least one of these paths will not fail is maximized. They refer to this problem as the maximum-reliability (max-rel) link-disjoint paths problem. The rationale behind this problem is to establish two disjoint paths that give 100% protection against a single-link failure, while reducing the failure probability of the connection as much as possible when multiple failures may occur. Assuming that the link-failure probabilities p_i are independent, then the reliability of a connection (consisting of two link-disjoint paths P_1 and P_2) is defined as $\prod_{i \in P_1} q_i + \prod_{j \in P_2} q_j + \prod_{i \in P_1} q_i \cdot \prod_{j \in P_2} q_j$, with $q_i = 1 - p_i$, for $i \in \mathcal{L}$. The max-rel link-disjoint paths problem is proven to be NP-complete. She et al. [114] evaluated two simple heuristic algorithms that both transform the link probabilities p_i to link weights $\log_q q_i$, with $q_i = 1 - p_i$, for $i \in \mathcal{L}$, and $0 < q < 1$. Based on these weights, one heuristic finds a shortest path, prunes its links from the graphs, and finds a shortest path in the pruned graph. This is often referred to as an active-path-first (APF) approach. The second heuristic uses Suurballe's algorithm to find two link-disjoint paths. Contrary to the first heuristic, the second always returns link-disjoint paths if they exist.

4.3. Multiple Failures. The single-link failure model has been most often considered in the literature, but multiple failures may occur as follows.

(i) Due to lengthy repair times of network equipment, there is a fairly long time span in which new failures could occur.

(ii) In case of terrorist attacks, several targeted parts of the network could be damaged. With Suurballe's algorithm, k link/node-disjoint paths can be found to establish full protection against $k - 1$ link/node failures.

(iii) In layered networks, for instance IP-over-WDM, one failure on the lowest-layer network, may cause multiple failures on higher-layer networks. Similarly, the links of a (single-layered) network may share the same duct, in which case a damaging of the duct may damage all the links inside. These links are often said to belong to the same shared risk link group (SRLG) (the node variant SRNG also exists. When both nodes and links can belong to a shared risk group, the term Shared Risk Resource Group (SRRG) is used, e.g., see [136]). Finding two SRLG-disjoint paths—paths of which the links in one path may not share a risk link group with links from the other path—is an NP-complete problem [137]. In specific cases, the SRLG-disjoint paths problem is polynomially

solvable, as discussed by Bhandari [138], Datta and Somani [139], and Luo and Wang [140]. In those cases, for instance, when the links in a SRLG share the same endpoint, a graph transformation can be made that reflects the shared risk groups, and on which a simple link-disjoint paths algorithm can be run. Lee et al. [141] have generalized the SRLG problem to include failure probabilities. In the deterministic SRLG scenario, when a SRLG r fails (e.g., a cable in the physical network is cut) all higher-layer links that belong to that group fail. In the probabilistic SRLG (PSRLG) scenario, the links (i, j) in that PSRLG r fail with probability $p_{ij}^r > 0$. If $p_{ij}^r \in \{0, 1\}$, for $(i, j) \in \mathcal{L}$, then the problem of finding PSRLG-disjoint paths reduces to the NP-complete problem of finding SRLG-disjoint paths. For SRLG types of problems, often an integer programming formulation is provided (e.g., [137, 142–144]) or an active-path-first (APF) approach is used as a heuristic. Hu [137] provided a basic ILP formulation to return a min-sum pair of SRLG-disjoint paths. Xu et al. [143] gave an ILP and an APF heuristic for the case of shared (backup paths) protection.

(iv) Natural disasters may affect all nodes and links within a certain geographical area. Work on multilink geographic failures has mostly focused on determining the geographic max-flow and min-cut values of a network under geographic failures of circular shape (e.g., Sen et al. [145], Agarwal et al. [146], and Neumayer et al. [147]). Trajanovski et al. [148] proved that, the problem of finding two region-disjoint paths is NP-hard, and they proposed a heuristic for it.

5. Conclusion

We have provided an overview of algorithms for network survivability. We have considered how to verify that a network has certain connectivity properties, how to augment an existing network to reach a given connectivity, and, lastly, how to find alternative paths in case network failures occur. Our focus has been on algorithms for general networks, although much work has also been done for specific networks, such as optical networks, where additional constraints like wavelength continuity and signal impairments induce an increased complexity, for example, see our work [149–151]. We have not discussed how to design a survivable network from scratch. Typically network design problems are hard to solve and involve many constraints, but since they only need to be solved sporadically, longer computation times are permitted. Predominantly, integer programming is used to design a network, as we have done in [152].

Acknowledgment

The author would like to thank Professor Piet Van Mieghem for his constructive comments on an earlier version of this paper.

References

[1] M. Al-Kuwaiti, N. Kyriakopoulos, and S. Hussein, "A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 2, pp. 106–124, 2009.

[2] K. Menger, "Zur allgemeinen kurventheorie," *Fundamenta Mathematicae*, vol. 10, pp. 96–115, 1927.

[3] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, pp. 298–305, 1973.

[4] P. Van Mieghem, *Graph Spectra for Complex Networks*, Cambridge University Press, 2011.

[5] N. M. M. de Abreu, "Old and new results on algebraic connectivity of graphs," *Linear Algebra and Its Applications*, vol. 423, no. 1, pp. 53–73, 2007.

[6] K. Lee, E. Modiano, and H. W. Lee, "Cross-layer survivability in WDM-based networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 4, pp. 1000–1013, 2011.

[7] K. Lee, H. W. Lee, and E. Modiano, "Reliability in layered networks with random link failures," *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1835–1848, 2011.

[8] W. Zou, M. Janic, R. Kooij, and F. A. Kuipers, "On the availability of networks," in *Proceedings of the BroadBand Europe*, Antwerp, Belgium, December 2007.

[9] C. J. Colbourn, *The Combinatorics of Network Reliability*, Oxford University Press, New York, NY, USA, 1987.

[10] P. Van Mieghem, H. Wang, X. Ge, S. Tang, and F. A. Kuipers, "Influence of assortativity and degree-preserving rewiring on the spectra of networks," *European Physical Journal B*, vol. 76, no. 4, pp. 643–652, 2010.

[11] D. Mosk-Aoyama, "Maximum algebraic connectivity augmentation is NP-hard," *Operations Research Letters*, vol. 36, no. 6, pp. 677–679, 2008.

[12] M. Yanakakis, "Computing the minimum fill-in is NP-complete," *SIAM Journal on Algebraic and Discrete Methods*, vol. 2, no. 1, pp. 77–79, 1981.

[13] P. Pan, G. Swallow, and A. Atlas, "Fast reroute extensions to RSVP-TE for LSP tunnels," IETF Request for Comments RFC 4090, 2005.

[14] M. Shand and S. Bryant, "IP fast reroute framework," IETF Request for Comments RFC 5714, 2010.

[15] R. Banner and A. Orda, "Designing low-capacity backup networks for fast restoration," in *Proceedings of the IEEE INFOCOM*, San Diego, Calif, USA, March 2010.

[16] G. Ellinas, A. G. Hailemariam, and T. E. Stern, "Protection cycles in mesh WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1924–1937, 2000.

[17] R. Asthana, Y. N. Singh, and W. D. Grover, "p-cycles: an overview," *IEEE Communications Surveys and Tutorials*, vol. 12, no. 1, pp. 97–111, 2010.

[18] M. Médard, S. G. Finn, R. A. Barry, and R. G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 641–652, 1999.

[19] K. P. Gummadi, M. J. Pradeep, and C. S. R. Murthy, "An efficient primary-segmented backup scheme for dependable real-time communication in multihop networks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 81–94, 2003.

[20] C. H. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *Journal of Computer and System Sciences*, vol. 43, no. 3, pp. 425–440, 1991.

[21] B. Mohar, "Isoperimetric numbers of graphs," *Journal of Combinatorial Theory, Series B*, vol. 47, no. 3, pp. 274–291, 1989.

[22] D. W. Matula and F. Shahrokhi, "Sparsest cuts and bottlenecks in graphs," *Discrete Applied Mathematics*, vol. 27, no. 1-2, pp. 113–123, 1990.

[23] T. N. Dinh, Y. Xuan, M. T. Thai, E. K. Park, and T. Znati, "On approximation of new optimization methods for assessing network vulnerability," in *Proceedings of the IEEE INFOCOM*, San Diego, Calif, USA, March 2010.

[24] D. R. Fulkerson and G. B. Dantzig, "Computation of maximal flows in networks," *Naval Research Logistics*, vol. 2, no. 4, pp. 277–283, 1955.

[25] P. Elias, A. Feinstein, and C. E. Shannon, "A note on the maximum flow through a network," *IRE Transactions on Information Theory*, vol. 2, pp. 117–119, 1956.

[26] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, NJ, USA, 1st edition, 1993.

[27] Y. Dinitz, "Dinitz' algorithm: the original version and Even's version," in *Essays in Memory of Shimon Even*, O. Goldreich, A. L. Rosenberg, and A. L. Selman, Eds., vol. 3895 of *Lecture Notes in Computer Science*, pp. 218–240, Springer, Berlin, Germany, 2006.

[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Mass, USA, 3rd edition, 2009.

[29] S. Even and R. E. Tarjan, "Network flow and testing graph connectivity," *SIAM Journal on Computing*, vol. 4, pp. 507–518, 1975.

[30] G. B. Dantzig, "Application of the simplex method to a transportation problem," in *Activity Analysis of Production and Allocation*, pp. 359–373, John Wiley & Sons, New York, NY, USA, 1951.

[31] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[32] A. V. Karzanov, "Determining the maximal flow in a network by the method of preflows," *Soviet Mathematics-Doklady*, no. 15, pp. 434–437, 1974.

[33] R. E. Tarjan, "A simple version of Karzanov's blocking flow algorithm," *Operations Research Letters*, vol. 2, no. 6, pp. 265–268, 1984.

[34] Z. Galil and A. Naamad, "An $O(EV \log^2 V)$ algorithm for the maximal flow problem," *Journal of Computer and System Sciences*, vol. 21, pp. 203–217, 1980.

[35] Y. Shiloach and U. Vishkin, "An $O(n^2 \log n)$ parallel max-flow algorithm," *Journal of Algorithms*, vol. 3, no. 2, pp. 128–146, 1982.

[36] D. D. Sleator and R. E. Tarjan, "A data structure for dynamic trees," *Journal of Computer and System Sciences*, vol. 26, no. 3, 1983.

[37] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM*, vol. 35, no. 4, pp. 921–940, 1988.

[38] R. K. Ahuja, J. B. Orlin, and R. E. Tarjan, "Improved time bounds for the maximum flow problem," *SIAM Journal on Computing*, vol. 18, no. 5, pp. 939–954, 1989.

[39] J. Cheriyan and T. Hagerup, "Randomized maximum-flow algorithm," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 203–226, 1995.

[40] N. Alon, "Generating pseudo-random permutations and maximum flow algorithms," *Information Processing Letters*, vol. 35, no. 4, pp. 201–204, 1990.

[41] A. V. Goldberg and S. Rao, "Beyond the flow decomposition barrier," *Journal of the ACM*, vol. 45, no. 5, pp. 783–797, 1998.

[42] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S. H. Teng, "Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs," in *Proceedings of The 43rd ACM Symposium on Theory of Computing, STOC' 11*, pp. 273–281, San Jose, CA, USA, June 2011.

[43] D. W. Matula, "Determining edge connectivity in $O(nm)$," in *Proceedings of the 28th Symposium on Foundations of Computer Science (FOCS '87)*, pp. 249–251, Los Angeles, Calif, USA, October 1987.

[44] Y. Shiloach, "Edge-disjoint branching in directed multigraphs," *Information Processing Letters*, vol. 8, no. 1, pp. 24–27, 1979.

[45] Y. Mansour and B. Schieber, "Finding the edge connectivity of directed graphs," *Journal of Algorithms*, vol. 10, no. 1, pp. 76–85, 1989.

[46] G. B. Dantzig and D. R. Fulkerson, "On the max-flow min-cut theorem of networks," in *Linear Inequalities and Related Systems*, Annals of Mathematics Studies, Study 38, pp. 215–221, Princeton University Press, Princeton, NJ, USA, 1956.

[47] S. Even, *Graph Algorithms*, Computer Science Press, 1979.

[48] M. R. Henzinger, S. Rao, and H. N. Gabow, "Computing vertex connectivity: new bounds from old techniques," *Journal of Algorithms*, vol. 34, no. 2, pp. 222–250, 2000.

[49] H. N. Gabow, "Using expander graphs to find vertex connectivity," *Journal of the ACM*, vol. 53, no. 5, pp. 800–844, 2006.

[50] Y. Yoshida and H. Ito, "Property testing on k-vertex-connectivity of graphs," *Algorithmica*, vol. 62, no. 3-4, pp. 701–712, 2012.

[51] P. Van Mieghem, D. Stevanović, F. A. Kuipers et al., "Decreasing the spectral radius of a graph by link removals," *Physical Review E*, vol. 84, no. 1, 2011.

[52] A. Natanzon, R. Shamir, and R. Sharan, "Complexity classification of some edge modification problems," *Discrete Applied Mathematics*, vol. 113, no. 1, pp. 109–128, 2001.

[53] G. Frederickson and J. JáJá, "Approximation algorithms for several graph augmentation problems," *SIAM Journal on Computing*, vol. 10, no. 2, pp. 270–283, 1981.

[54] K. P. Eswaran and R. E. Tarjan, "Augmentation problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 653–665, 1976.

[55] J. Jensen and T. Jordán, "Edge-connectivity augmentation preserving simplicity," in *Proceedings of the 9th Annual ACM/SIAM Symposium On Discrete Algorithms (SODA '97)*, pp. 306–315, 1997.

[56] S. Raghavan, "A note on Eswaran and Tarjan's algorithm for the strong connectivity augmentation problem," in *The Next Wave in Computing, Optimization, and Decision Technologies*, vol. 29 of *Operations Research/Computer Science Interfaces Series*, pp. 19–26, Springer, 2005.

[57] R. E. Tarjan, "A note on finding the bridges of a graph," *Information Processing Letters*, vol. 2, no. 6, pp. 160–161, 1974.

[58] E. A. Dinitz, A. V. Karzanov, and M. V. Lomonosov, "On the structure of the system of minimum edge cuts of a graph," in *Issledovaniya po Diskretnoi Optimizatsii*, A. A. Fridman, Ed., pp. 290–306, Nauka, Moscow, Russia, 1976.

[59] A. V. Karzanov and E. A. Timofeev, "Efficient algorithm for finding all minimal edge cuts of a nonoriented graph," *Cybernetics and Systems Analysis*, vol. 22, no. 2, pp. 156–162, 1986.

[60] H. Nagamochi and T. Kameda, "Canonical cactus representation for minimum cuts," *Japan Journal of Industrial and Applied Mathematics*, vol. 11, no. 3, pp. 343–361, 1994.

[61] R. E. Gomory and T. C. Hu, "Multi-terminal network flows," *Journal of the Society for Industrial and Applied Mathematics*, vol. 9, no. 4, pp. 551–570, 1961.

[62] D. Gusfield and D. Naor, "Extracting maximal information about sets of minimum cuts," *Algorithmica*, vol. 10, no. 1, pp. 64–89, 1993.

[63] L. Fleischer, "Building chain and cactus representations of all minimum cuts from Hao-Orlin in the same asymptotic run time," *Journal of Algorithms*, vol. 33, no. 1, pp. 51–72, 1999.

[64] H. Nagamochi, "Graph algorithms for network connectivity problems," *Journal of the Operations Research Society of Japan*, vol. 4, no. 4, pp. 199–223, 2004.

[65] D. Naor, D. Gusfield, and C. Martel, "A fast algorithm for optimally increasing the edge connectivity," *SIAM Journal on Computing*, vol. 26, no. 4, pp. 1139–1165, 1997.

[66] E. Cheng, "Successive edge-connectivity augmentation problems," *Mathematical Programming B*, vol. 84, no. 3, pp. 577–593, 1999.

[67] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[68] W. Mader, "A reduction method for edge-connectivity in graphs," *Annals of Discrete Mathematics*, vol. 3, pp. 145–164, 1978.

[69] A. Frank, "On a theorem of Mader," *Discrete Mathematics*, vol. 101, no. 1–3, pp. 49–57, 1992.

[70] V. D. Podderyugin, "An algorithm for determining edge-connectivity of a graph," in *Proceedings of the Seminar on Combinatorial Mathematics*, Moscow, Russia, 1971, Doklady Akademii Nauk SSSR, Scientiffic Council on the Complex Problem "Cybernetics", pp. 136–141, 1973.

[71] H. Nagamochi and T. Ibaraki, "Computing edge-connectivity in multigraphs and capacitated graphs," *SIAM Journal on Discrete Mathematics*, vol. 5, no. 1, pp. 54–66, 1992.

[72] Z. Galil and G.F. Italiano, "Reducing edge connectivity to vertex connectivity," *ACM SIGACT News*, vol. 22, no. 1, pp. 57–61, 1991.

[73] H. N. Gabow, "A matroid approach to finding edge-connectivity and packing arborescences," *Journal of Computer and System Sciences*, vol. 50, no. 2, pp. 259–273, 1995.

[74] D. R. Karger, "Minimum cuts in near-linear time," *Journal of the ACM*, vol. 47, no. 1, pp. 46–76, 2000.

[75] G. Cai and Y. Sun, "The minimum augmentation of any graph to a K-edge-connected graph," *Networks*, vol. 19, no. 1, pp. 151–172, 1989.

[76] T. Watanabe and A. Nakamura, "Edge-connectivity augmentation problems," *Journal of Computer and System Sciences*, vol. 35, no. 1, pp. 96–144, 1987.

[77] A. Frank, "Augmenting graphs to meet edge-connectivity requirements," *SIAM Journal on Discrete Mathematics*, vol. 5, no. 1, pp. 25–53, 1992.

[78] H. N. Gabow, "Applications of a poset representation to edge connectivity and graph rigidity," in *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS '91)*, pp. 812–821, October 1991.

[79] A. A. Benczúr, "Augmenting undirected connectivity in RNC and in randomized $\tilde{O}(n^3)$ time," in *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC '94)*, pp. 658–667, New York, NY, USA, May 1994.

[80] H. Nagamochi and T. Ibaraki, "Deterministic $\tilde{O}(nm)$ time edge-splitting in undirected graphs," *Journal of Combinatorial Optimization*, vol. 1, no. 1, pp. 5–46, 1997.

[81] A. A. Benczúr and D. R. Karger, "Augmenting undirected edge connectivity in $\tilde{O}(n^2)$ time," *Journal of Algorithms*, vol. 37, no. 1, pp. 2–36, 2000.

[82] H. Nagamochi and T. Ibaraki, "Graph connectivity and its augmentation: applications of MA orderings," *Discrete Applied Mathematics*, vol. 123, pp. 447–472, 2002.

[83] Z. Nutov, "Approximating connectivity augmentation problems," *ACM Transactions on Algorithms*, vol. 6, no. 1, article 5, 2009.

[84] J. Plesník, "Minimum block containing a given graph," *Archiv der Mathematik*, vol. 27, no. 1, pp. 668–672, 1976.

[85] S. Khuller and R. Thurimella, "Approximation algorithms for graph augmentation," *Journal of Algorithms*, vol. 14, no. 2, pp. 214–225, 1993.

[86] S. Taoka, T. Watanabe, and T. Mashima, "Maximum weight matching-based algorithms for k-edge-connectivity augmentation of a graph," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, pp. 2231–2234, May 2005.

[87] A. Rosenthal and A. Goldner, "Smallest augmentation to biconnect a graph," *SIAM Journal on Computing*, vol. 6, no. 1, pp. 55–66, 1977.

[88] T. S. Hsu and V. Ramachandran, "Finding a smallest augmentation to biconnect a graph," *SIAM Journal on Computing*, vol. 22, no. 5, pp. 889–912, 1993.

[89] T. Watanabe and A. Nakamura, "A minimum 3-connectivity augmentation of a graph," *Journal of Computer and System Sciences*, vol. 46, no. 1, pp. 91–128, 1993.

[90] T. Jordán, "On the optimal vertex-connectivity augmentation," *Journal of Combinatorial Theory B*, vol. 63, no. 1, pp. 8–20, 1995.

[91] T. Hsu, "On four-connecting a triconnected graph," in *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS '92)*, pp. 70–79, IEEE Computer Society, Washington, DC, USA, October 1992.

[92] T. Jordán, "A note on the vertex-connectivity augmentation problem," *Journal of Combinatorial Theory B*, vol. 71, no. 2, pp. 294–301, 1997.

[93] B. Jackson and T. Jordán, "A near optimal algorithm for vertex-connectivity augmentation," in *Proceedings of the 11th International Conference on Algorithms and Computation (ISAAC '00)*, pp. 312–325, Springer, London, UK, December 2000.

[94] B. Jackson and T. Jordán, "Independence free graphs and vertex connectivity augmentation," *Journal of Combinatorial Theory B*, vol. 94, no. 1, pp. 31–77, 2005.

[95] G. Kortsarz and Z. Nutov, "Approximating minimum-cost connectivity problems," in *Handbook of Approximation Algorithms and Metaheuristics*, Chapter 58, Chapman & Hall/CRC, 2007.

[96] G. Liberman and Z. Nutov, "On shredders and vertex connectivity augmentation," *Journal of Discrete Algorithms*, vol. 5, no. 1, pp. 91–101, 2007.

[97] J. Cheriyan and R. Thurimella, "Fast algorithms for k-shredders and k-node connectivity augmentation," *Journal of Algorithms*, vol. 33, no. 1, pp. 15–50, 1999.

[98] L. A. Végh, "Augmenting undirected node-connectivity by one," *SIAM Journal on Discrete Mathematics*, vol. 25, no. 2, pp. 695–718, 2011.

[99] A. Frank and T. Jordán, "Minimal edge-coverings of pairs of sets," *Journal of Combinatorial Theory, Series B*, vol. 65, no. 1, pp. 73–110, 1995.

[100] A. Frank and L. A. Végh, "An algorithm to increase the node-connectivity of a digraph by one," *Discrete Optimization*, vol. 5, no. 4, pp. 677–684, 2008.

[101] A. Frank, "Connectivity augmentation problems in network design," in *Mathematical Programming: State of the Art 1994*, J. R. Bridge and K. G. Murty, Eds., University of Michigan, Ann Arbor, Mich, USA, 1994.

[102] M. Grötschel, C. L. Monma, and M. Stoer, "Design of survivable networks," in *Handbooks in Operations Research and Management Science*, vol. 7, pp. 617–672, 1995.

[103] M. Penn and H. Shasha-Krupnik, "Improved approximation algorithms for weighted 2- and 3-vertex connectivity augmentation problems," *Journal of Algorithms*, vol. 22, no. 1, pp. 187–196, 1997.

[104] S. Khuller, "Approximation algorithms for finding highly connected subgraphs," in *Approximation Algorithms for NP-Hard Problems*, D. S. Hochbaum, Ed., pp. 236–265, PWS Publishing, Boston, Mass, USA, 1997.

[105] S. Khuller and B. Raghavachari, "Improved approximation algorithms for uniform connectivity problems," *Journal of Algorithms*, vol. 21, no. 2, pp. 434–450, 1996.

[106] R. Ravi and D. P. Williamson, "An approximation algorithm for minimum-cost vertex-connectivity problems," *Algorithmica*, vol. 18, no. 1, pp. 21–43, 1997.

[107] G. N. Frederickson and J. JáJá, "On the relationship between the biconnectivity augmentation and travelling salesman problems," *Theoretical Computer Science*, vol. 19, no. 2, pp. 189–201, 1982.

[108] 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.

[109] B. Yang, S. Q. Zheng, and S. Katukam, "Finding two disjoint paths in a network with min-min objective function," in *Proceedings of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems*, November 2003.

[110] A. Itai, Y. Perl, and Y. Shiloach, "The complexity of finding maximum disjoint paths with length constraints," *Networks*, vol. 12, no. 3, pp. 277–286, 1982.

[111] A. Bley, "On the complexity of vertex-disjoint length-restricted path problems," *Computational Complexity*, vol. 12, no. 3-4, pp. 131–149, 2003.

[112] C.-L. Li, S. T. Mccormick, and D. Simchi-Levi, "Finding disjoint paths with different path costs: complexity and algorithms," *Networks*, vol. 22, no. 7, pp. 653–667, 1992.

[113] D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He, "On finding disjoint paths in single and dual link cost networks," in *Proceedings of the IEEE INFOCOM*, March 2004.

[114] Q. She, X. Huang, and J. P. Jue, "How reliable can two-path protection be?" *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 922–933, 2010.

[115] B. H. Shen, B. Hao, and A. Sen, "On multipath routing using widest pair of disjoint paths," in *Proceedings of the High Perfomance Switching and Routing (HPSR '04)*, pp. 134–140, April 2004.

[116] A. A. Beshir and F. A. Kuipers, "Variants of the min-sum link-disjoint paths problem," in *Proceedings of the 16th Annual IEEE Symposium on Communications and Vehicular Technology (IEEE SCVT '09)*, IEEE/SCVT, Louvain-la-Neuve, Belgium, November 2009.

[117] R. Bhatia, M. Kodialam, and T. V. Lakshman, "Finding disjoint paths with related path costs," *Journal of Combinatorial Optimization*, vol. 12, no. 1-2, pp. 83–96, 2006.

[118] H. D. Sherali, K. Ozbay, and S. Subramanian, "The time-dependent shortest pair of disjoint paths problem: complexity, models, and algorithms," *Networks*, vol. 31, no. 4, pp. 259–272, 1998.

[119] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, no. 2, pp. 125–145, 1974.

[120] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325–336, 1984.

[121] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-service routing using maximally disjoint paths," in *Proceedings of the 7th International Workshop on Quality of Service (IWQoS)*, pp. 119–128, London, UK, May 1999.

[122] R. G. Ogier, V. Rutenburg, and N. Shacham, "Distributed algorithms for computing shortest pairs of disjoint paths," *IEEE Transactions on Information Theory*, vol. 39, no. 2, pp. 443–455, 1993.

[123] D. Sidhu, R. Nair, and S. Abdallah, "Finding disjoint paths in networks," *ACM SIGCOMM Computer Communication Review*, vol. 21, no. 4, pp. 43–51, 1991.

[124] J. Roskind and R. E. Tarjan, "Note on finding minimum-cost edge-disjoint spanning trees," *Mathematics of Operations Research*, vol. 10, no. 4, pp. 701–708, 1985.

[125] G. Xue, L. Chen, and K. Thulasiraman, "Quality-of-service and quality-of-protection issues in preplanned recovery schemes using redundant trees," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 8, pp. 1332–1345, 2003.

[126] W. Zhang, G. Xue, J. Tang, and K. Thulasiraman, "Faster algorithms for construction of recovery trees enhancing QoP and QoS," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 642–655, 2008.

[127] P. Cholda, A. Mykkeltveit, B. E. Helvik, O. J. Wittner, and A. Jajszczyk, "A survey of resilience differentiation frameworks in communication networks," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 4, pp. 32–55, 2007.

[128] S. Ramasubramanian, M. Harkara, and M. Krunz, "Linear time distributed construction of colored trees for disjoint multipath routing," *Computer Networks*, vol. 51, no. 10, pp. 2854–2866, 2007.

[129] Y. Guo, F. A. Kuipers, and P. Van Mieghem, "A link-disjoint paths algorithm for reliable QoS routing," *International Journal of Communication Systems*, vol. 16, no. 9, pp. 779–798, 2003.

[130] R. Banner and A. Orda, "The power of tuning: a novel approach for the efficient design of survivable networks," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 737–749, 2007.

[131] O. Gerstel and G. Sasaki, "Quality of protection (QoP): a quantitative unifying paradigm to protection service grades," *Optical Networks Magazine*, vol. 3, no. 3, pp. 40–49, 2002.

[132] A. V. Goldberg and R. E. Tarjan, "Finding minimum-cost circulations by canceling negative cycles," *Journal of the ACM*, vol. 36, no. 4, pp. 873–886, 1989.

[133] H. Luo, L. Li, and H. Yu, "Routing connections with differentiated reliability requirements in WDM mesh networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 253–266, 2009.

[134] F. A. Kuipers, A. Orda, D. Raz, and P. Van Mieghem, "A comparison of exact and ε-approximation algorithms for constrained routing," in *Proceedings of the 5th IFIP Networking Conference*, Coimbra, Portugal, May 2006.

[135] A. Chakrabarti and G. Manimaran, "Reliability constrained routing in QoS networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 662–675, 2005.

[136] D. Coudert, P. Datta, S. Perennes, H. Rivano, and M. E. Voge, "Shared risk resource group: complexity and approximability issues," *Parallel Processing Letters*, vol. 17, no. 2, pp. 169–184, 2007.

[137] J. Q. Hu, "Diverse routing in optical mesh networks," *IEEE Transactions on Communications*, vol. 51, no. 3, pp. 489–494, 2003.

[138] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*, Kluwer Academic Publishers, New York, NY, USA, 1999.

[139] P. Datta and A. K. Somani, "Graph transformation approaches for diverse routing in shared risk resource group (SRRG) failures," *Computer Networks*, vol. 52, no. 12, pp. 2381–2394, 2008.

[140] X. Luo and B. Wang, "Diverse routing in WDM optical networks with shared risk link group (SRLG) failures," in *Proceedings of the 5th IEE International Workshop on Design of Reliable Communication Networks (DRCN '05)*, Island of Ischia, Naples, Italy, October 2005.

[141] H. W. Lee, E. Modiano, and K. Lee, "Diverse routing in networks with probabilistic failures," *IEEE/ACM Transactions on Networking*, vol. 18, no. 6, pp. 1895–1907, 2010.

[142] W. D. Grover, *Mesh-Based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*, Prentice Hall PTR, London, UK, 2003.

[143] D. Xu, Y. Xiong, C. Qiao, and G. Li, "Trap avoidance and protection schemes in networks with shared risk link groups," *Journal of Lightwave Technology*, vol. 21, no. 11, pp. 2683–2693, 2003.

[144] H. Zang, C. S. Ou, and B. Mukherjee, "Path-protection routing and wavelength assignment (RWA) in WDM mesh networks under duct-layer constraints," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 248–258, 2003.

[145] A. Sen, S. Murthy, and S. Banerjee, "Region-based connectivity—a new paradigm for design of fault-tolerant networks," in *Proceedings of the 15st International Conference on High Performance Switching and Routing (HPSR '09)*, Paris, France, June 2009.

[146] P. K. Agarwal, A. Efrat, S. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman, "The resilience of WDM networks to probabilistic geographical failures," in *Proceedings of the IEEE INFOCOM*, pp. 1521–1529, Shanghai, China, April 2011.

[147] S. Neumayer, A. Efrat, and E. Modiano, "Geographic max-flow and mincut under a circular disk failure model," in *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM '12)*, March 2012.

[148] S. Trajanovski, F. A. Kuipers, P. Van Mieghem, A. Ilić, and J. Crowcroft, "Critical regions and region-disjoint paths in a network".

[149] A. A. Beshir, F. A. Kuipers, P. Van Mieghem, and A. Orda, "On-line survivable routing in WDM networks," in *Proceedings of the 21st International Teletraffic Congress (ITC '21)*, Paris, France, September 2009.

[150] A. A. Beshir, F. A. Kuipers, A. Orda, and P. Van Mieghem, "Survivable impairment-aware traffic grooming in WDM rings," in *Proceedings of the 23rd International Teletraffic Congress*, San Francisco, Calif, USA, September 2011.

[151] A. A. Beshir, F. A. Kuipers, A. Orda, and P. Van Mieghem, "Survivable routing and regenerator placement in optical networks," in *Proceedings of the 4th International Workshop on Reliable Networks Design and Modeling (RNDM '12)*, Petersburg, Russia, October 2012.

[152] A. A. Beshir, R. Nuijts, R. Malhotra, and F. A. Kuipers, "Survivable impairment-aware traffic grooming," in *Proceedings of the 16th European Conference on Networks and Optical Communications (NOC '11)*, Northumbria University, Newcastle upon Tyne, UK, July 2011.