DISJOINT PATHS IN NETWORKS

Farabi Iqbal and Fernando A. Kuipers
Network Architectures and Services,
Delft University of Technology,
Mekelweg 4, 2628 CD Delft,
The Netherlands
Email: {M.A.F.Iqbal,F.A.Kuipers}@tudelft.nl

INTRODUCTION

Networks are prevalent in our daily lives. Our body consists of networks of neurons interconnected by synapses. Our transportation networks enable us to commute between places easily. The Internet, our vast gateway of information, is made up of networks of worldwide computers. The power grid networks provide electricity, without which society (nowadays) might cease to function. Our social networks keep us up to date with friends and families. Due to the importance of networks, varieties of network properties have been studied extensively, particularly in the field of graph theory.

In graph theory, a network is viewed as an interconnection of *nodes* by *links*. Nodes represent the points-of-interest in a network, e.g., routers in a communication network, ports in a maritime network, or cities in a transportation network. Links represent the connectors that bind points-of-interest together, e.g., cables in a communication network, trade routes in a maritime network or highways in a transportation network.

One of the most studied topics in graph theory is the shortest path problem, which is the problem of finding a path between two nodes in a network such that the sum of the weights of its constituent links is minimized. Examples of conventional shortest path algorithms are the Dijkstra algorithm [1] and the Bellman-Ford algorithm [2, 3]. Using the shortest path, signals can be exchanged with minimal delay between two routers in a communication network, freights can be sent with minimal fuel cost between two ports in a maritime network, and we can commute between cities faster.

The disjoint paths (DP) problem could be seen as an extension of the shortest path problem. Instead of having just a single shortest path, several paths that do not share any common links (or nodes) are computed. Providing disjoint paths to network traffic would increase the reliability of network connections, and correspondingly the network survivability. Network survivability is defined as the network's capability to provide continuous service in the presence of network component (e.g., nodes and/or links) failures [4]. Another variant of the disjoint paths problem is the disjoint path pairs problem, where instead of finding multiple disjoint paths for a single pair of source and destination nodes, a single path is computed for each pair of source and destination nodes, such that these paths are disjoint.

Disjoint paths have a wide range of applications. For example, having multiple disjoint paths for traffic in a communication network would improve its transmission reliability. By sending the traffic concurrently on multiple disjoint paths, the failure of a path would not affect the performance of other paths, and the traffic would still reach its destination. In transportation networks, having a pre-calculated number of disjoint paths would enable a truck driver to follow a different path for a change of scenery instead of always sticking to the shortest path.

The remainder of this paper is organized as follows. In the **Disjoint Paths** section, we give a formal definition of the disjoint paths problem, discuss additional conditions for the disjoint paths problem and their corresponding complexity, and explain some representative disjoint paths algorithms. In the **Availability-Based Disjoint Paths** section, we introduce the notion of path availability and its relation to the disjoint paths problem. The **Maximally Disjoint Paths** section elaborates on the scenario where the disjoint paths can partially overlap, instead of being fully disjoint. We continue with finding disjoint paths in multi-domain network context in the **Domain-Disjoint Paths** section. Since multiple links (or nodes) may fail simultaneously under a similar shared risk, the **Shared Risk Link Group (SRLG)-Disjoint Paths** section introduces the shared risk link group concept, and explains approaches for ensuring that the disjoint paths will not fail simultaneously due to a single link (or node) failure. Risks may also affect networks on a region basis, thus the **Region-Disjoint Paths** section discusses several region-based risk models and corresponding approaches for finding region-disjoint paths. A counterpart to the disjoint paths problem, the disjoint path pairs problem is covered in the **Disjoint Path Pairs** section, where the complexity of several additional conditions to the disjoint path pairs problem is also discussed. Finally, we give a brief summary of the paper in the last section.

DISJOINT PATHS

The disjoint paths (DP) problem is formally defined as follows:

Problem 1 *Disjoint paths (DP) problem:*
Given a directed network $G = (\mathcal{N}, \mathcal{L})$ of a set $\mathcal{N}$ of N nodes and a set $\mathcal{L}$ of L weighted links, two special nodes $s,t \in \mathcal{N}$, and an integer $k > 0$. Find k paths $P_1, P_2, ..., P_k$ from s to t, such that the paths share no common links (or nodes).

We consider only directed networks since an undirected network can be transformed into a representative directed network such as by the polynomial-time approach of [5] as illustrated in Figure 2. Each node n is split into two nodes n_{in} and n_{out} that are connected by a directed link with zero weight, and each undirected link (u, v) is replaced with directed links (u_{out}, v_{in}) and (v_{out}, u_{in}) of weight ℓ_{uv}. However, the transformation is not reciprocal since a directed network cannot have a representative undirected network.

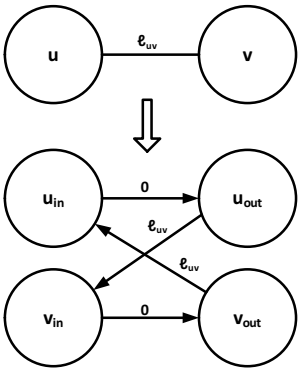


Figure 1: Network directivity transformation

We also consider only networks with link weights in the problem definition, instead of also considering networks with node weights. A weighted node n can be split into two unweighted nodes n_{in} and n_{out} that are connected by a directed link with the weight of n as illustrated in Figure 2, where ℓ_n represents the weight of node n. All the incoming links of node n will be connected to n_{in}, while all the outgoing links of node n will be connected to node n_{out}.

There are various disjointness criteria for computing the disjoint paths. In this section, we focus on the two most common disjointness criteria, namely node-disjointness and link-disjointness. Other types of disjointness criteria will be covered in later sections. Node-disjoint paths share no common nodes, except at the source and destination nodes, ensuring that at least one path remains available in case of a node or a link failure. Link-disjoint paths share no common links, ensuring that at least one path remains available in case of a link failure.

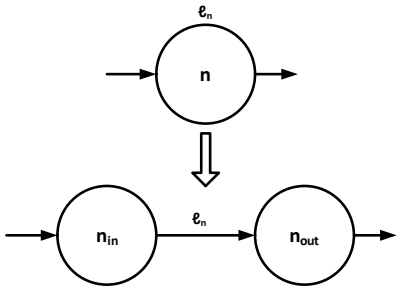


Figure 2: Node weight transformation

Finding node-disjoint paths is more restrictive than finding link-disjoint paths, since if two paths are node-disjoint, they are also link-disjoint. Link-disjoint paths algorithms can be used to find node-disjoint paths, using the node-splitting technique of [3], which is illustrated in Figure 3. Each node n of G is split into two nodes n_{in} and n_{out}, and connected by a directed link (n_{in}, n_{out}) of weight zero. All the incoming links of node n will be connected to n_{in}, while all the outgoing links of node n will be connected to node n_{out}.

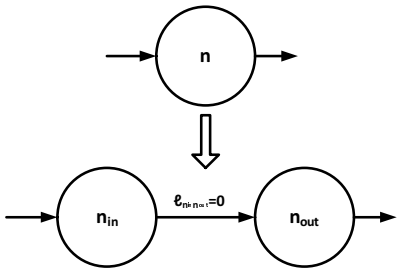


Figure 3: Node splitting transformation

Additional conditions could also be imposed on the disjoint paths, such as:

Min-max disjoint paths problem - the sum of the weights of all the constituent links of the path with the largest path weight is minimized.

Min-min disjoint paths problem - the sum of the weights of all the constituent links of the path with the smallest path weight is minimized.

Bounded disjoint paths problem - the sum of the weights of all the constituent links of each path should each be less than Δ.

Min-sum disjoint paths problem - the sum of the weights of all the constituent links of the k paths is minimized.

Li et al. [6] prove that the min-max condition is strongly NP-complete, except in directed acyclic networks, where it is (weakly) NP-complete. The min-max condition is useful when all the k paths are used simultaneously to send the traffic (e.g., in communication networks).

The min-min condition is NP-hard to solve [7], and to be approximated within a factor of ε for any constant $\varepsilon > 1$ [8]. The min-min condition is useful when only one path is active and used by traffic, while the other $k-1$ paths remain idle as backups. Only when an active path fails will one of the backup paths be activated to substitute the active path. Hence, the active path should have as minimal weight as possible, since the active path is used more frequently than the backup paths.

The bounded condition is NP-hard [9] and APX-hard [10]. The bounded condition is helpful when each of the path weights needs to be constrained. The k paths may or may not have similar path weight constraint.

The min-sum condition, the simplest of all the four conditions, can be solved efficiently. However, in the presence of secondary conditions (e.g., min-max, min-min or bounded conditions) for resolving a tie of solutions, this variant of the min-sum condition becomes NP-hard [11]. For the remainder of this section, we focus on discussing several representative polynomial-time algorithms that are usable in tackling the min-sum disjoint paths problem.

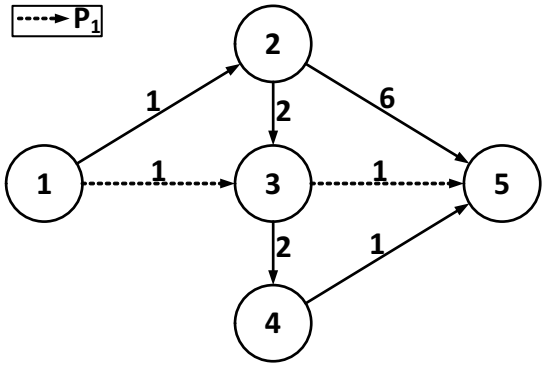
A simple heuristic for solving the disjoint paths problem is shown in Algorithm 1, which we will refer to as the Iterative DP algorithm. The Iterative DP algorithm is based on the use of k consecutive shortest path computations.

Algorithm 1 Iterative DP(G,s,t,k)

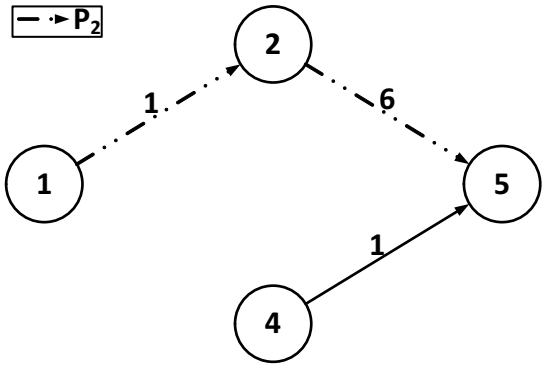
1: **for** $i = 1,...,k$
2: Find the shortest path P_i from node s to node t
3: Remove (intermediate nodes) links of P_i from G

Consider the problem of finding $k = 2$ min-sum disjoint paths from node 1 to node 5 in the network G of five nodes and seven links, illustrated in Figure 4a. The shortest path from 1 to 5 is $P_1 = 1-3-5$ of weight 2. For node-disjoint paths, node 3 is removed from G and the new shortest path in the modified network G is $P_2 = 1-2-5$ of weight 7, as illustrated in Figure 4b. For link-disjoint paths, links $(1,3)$ and $(3,5)$ are removed from G, and the new shortest path in the modified network G is $P_2 = 1-2-3-4-5$ of weight 6, as illustrated in Figure 4c.

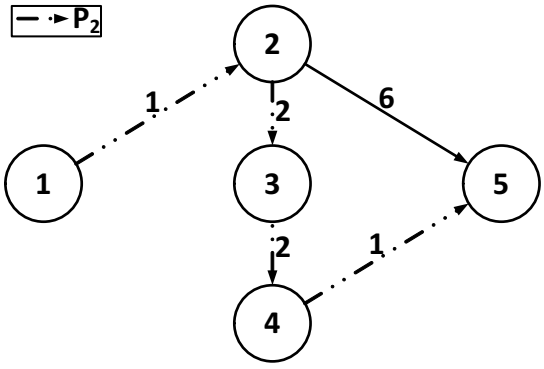
The Iterative DP algorithm may fail to return disjoint paths even though they exist, for instance in *trap topologies* [12]. An example of a trap topology is illustrated in Figure 5a. Consider a problem of finding $k = 3$ node-disjoint paths from node 1 to node 5 in the network. The Iterative DP algorithm computes $P_1 = 1-2-3-4-5$ of weight 4 as the first path. After



(a) P_1



(b) P_2 for the node-disjoint paths

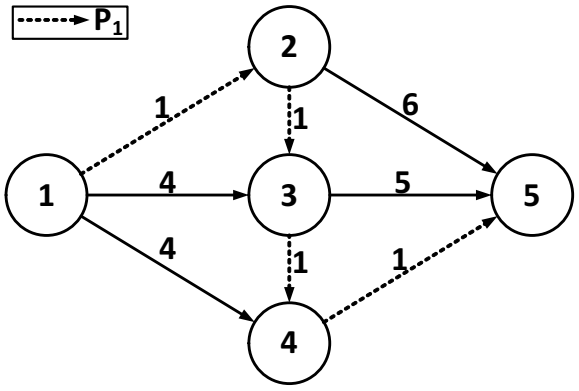


(c) P_2 for the link-disjoint paths

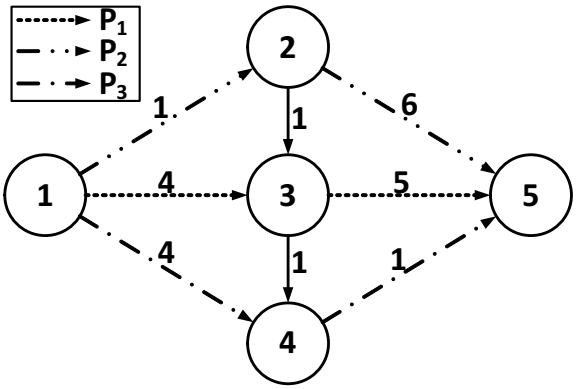
Figure 4: Illustrative example of the Iterative DP algorithm.

nodes 2, 3 and 4 are removed from G, node 5 become disconnected and there are no paths possible from node 1 to node 5. Choosing P_1 as the first path has disconnected the network. If a different path is chosen as the first path, there can actually be two more paths in the network, as illustrated by Figure 5b. Although the trap topology may be relatively rare [12], even in its absence the Iterative DP algorithm provides no guarantee that the total path weight of the disjoint paths satisfies the

min-sum condition.



(a) P_1 of the Iterative DP algorithm



(b) The actual P_1, P_2 and P_3

Figure 5: Example of a network with trap topology.

Bhandari [13] proposes a min-sum disjoint paths algorithm that does not succumb to the trap topology. Although there exists an earlier algorithm known as the Suurballe algorithm [14] that can also circumvent the trap topology problem, we focus on the Bhandari algorithm, because it is simpler to present.

The pseudo code of the Bhandari algorithm is presented in Algorithm 2. We illustrate the use of the Bhandari algorithm to find $k = 3$ link-disjoint paths from node 1 to node 5 for the network shown in Figure 6a. The first preliminary shortest path from 1 to 5 is $P_1 = 1-2-3-4-5$ of weight 4. The constituent links of P_1 are reversed and their weight is inverted as shown in Figure 6b. Then, another preliminary shortest path $P_2 = 1-4-3-5$ of weight 8 is computed, as shown in Figure 6c. The overlapping links are excluded to get the two link-disjoint paths P_1 and P_2, and lines 3 - 5 are repeated again before another preliminary path $P_3 = 1-3-2-6$ of weight 9 is computed, as shown in Figure 6d. Once again, the overlapping links are excluded to get the three final link-disjoint paths as shown in Figure 6e.

Since the Bhandari algorithm works for networks with single weighted links, Guo et al. [15] propose a heuristic algorithm for finding link-disjoint multi-constrained paths between a pair of source and destination nodes, where the network links are characterized by multiple link weights.

Instead of finding disjoint paths between two nodes, Suurballe and Tarjan [16] propose an adaptation of the Suurballe algorithm [14] to find a pair (i.e., $k = 2$) of link-disjoint paths from a source node s to all reachable destination nodes, in $O(N \log N + L)$ time, similar to the Dijkstra algorithm.

Though the Suurballe-Tarjan and the Bhandari algorithms are often mentioned as simply the Suurballe algorithm, these algorithms should be differentiated. Though this confusion does not affect the results of the papers, we believe that proper citation is necessary such that the confusion does not spread.

AVAILABILITY-BASED DISJOINT PATHS

The availability-based disjoint paths problem is formally defined as follows:

Problem 2 *Availability-based disjoint paths problem:*
Given a network $G = (\mathcal{N}, \mathcal{L})$ of a set $\mathcal{N}$ of N nodes and a set $\mathcal{L}$ of L weighted links, two special nodes $s,t \in \mathcal{N}$, a decimal number δ, and an integer $k > 0$. Each link weight represents the availability of the link. Find k paths $P_1, P_2, ..., P_k$ from s to t, such that the paths share no common links (or nodes), and that the total availability A_t of a connection that is assigned with the k paths is at least δ.

Path availability is the probability that the path will be found in operating state at a random time in the future [17]. The availability (A) of a path P can be computed by multiplying the availability of all of its constituent links:

$$A = \prod_{y \in P} a_y \tag{1}$$

where a_y is the availability of the link y that depends on the Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR) of the link.

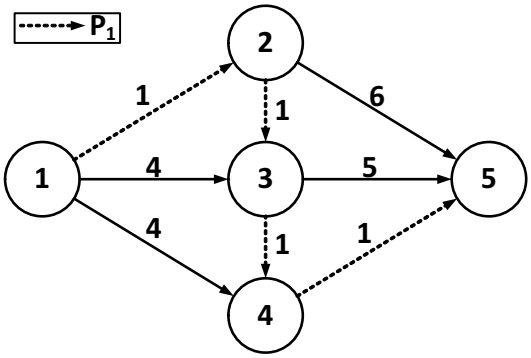
$$a = \frac{MTBF}{MTBF + MTTR} \tag{2}$$

$$MTBF = \frac{\text{total operating time}}{\text{number of failures}} = \frac{1}{\text{failure rate}} \tag{3}$$

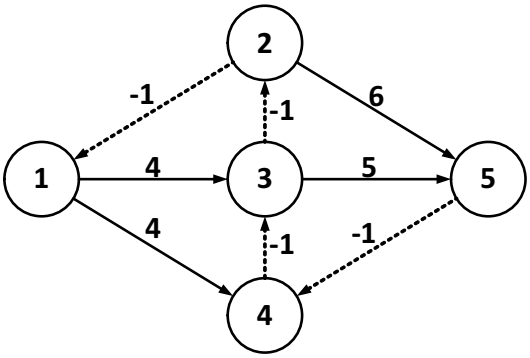
$$MTTR = \text{failure localization time} + \text{failure repair time} \tag{4}$$

Algorithm 2 Bhandari (G, s, t, k)

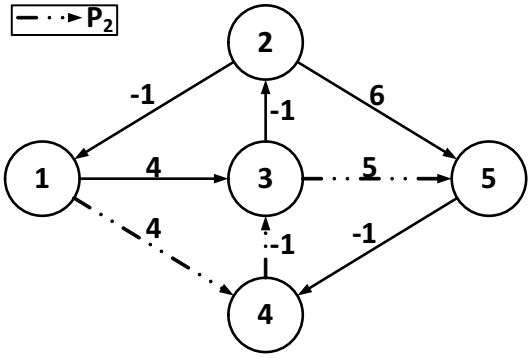
Find the shortest path P_1 from node s to node t
for $i = 2, ..., k$
    For node-disjoint paths, split the intermediate nodes of all P_x where $x < i$ (refer to Figure 3)
    Replace each link of all P_x where $x < i$ with a reverse link of inverted link weight in the original graph
    Find the shortest path P_i from node s to node t
    Remove all overlapping links to get i disjoint paths P_x where $x \leq i$.



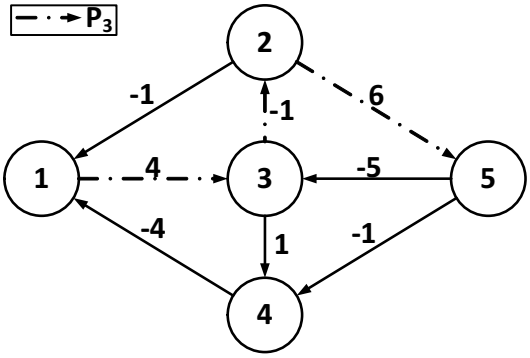
(a) P_1



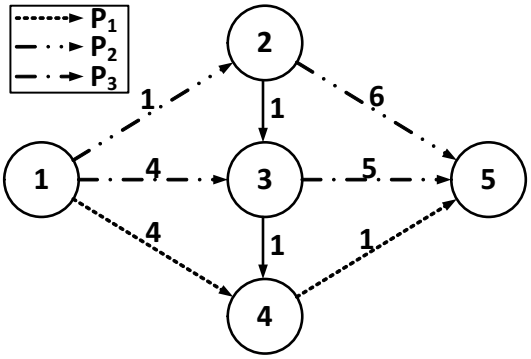
(b) link reversal



(c) P_2



(d) P_3



(e) non-overlapping P_1, P_2 and P_3

Figure 6: Illustrative example of Bhandari algorithm

MTBF is the mean time that the link is operational and MTTR is the mean time that the link is non-operational (while it is being repaired). The time required to localize the link failure is considered as a part of the MTTR and the failed link is put to operation as soon as it is fixed.

When traffic is duplicated and sent simultaneously on all of its assigned disjoint paths from a source node to a destination node, the connection reliability of the traffic is determined by all of the disjoint paths. Connection reliability in this context means the probability that the connection will operate without any service-affecting failure for a period of time [17]. This scenario is quite common in telecommunication networks where data can be duplicated. Providing multiple disjoint paths for traffic would increase its connection reliability. The total availability A_t of a connection that is assigned with k disjoint paths can be calculated as:

$$A_t = 1 - \prod_k (1 - A_k) \tag{5}$$

where A_k is the availability of path k. For example, the total availability A_t of a connection that is assigned with $k = 2$ disjoint paths P_1 and P_2 can be calculated as:

$$\begin{aligned} A_t &= 1 - (U_1)(U_2) \\ &= 1 - (1 - A_1)(1 - A_2) \\ &= A_1 + A_2(1 - A_1) \end{aligned} \tag{6}$$

where U_x is the unavailability of path x [18]. Instead of considering fully disjoint paths, Yang et al. [19] also compute the connection availability of partially disjoint paths, and they provide algorithms for finding availability-based (partially) disjoint paths. The benefits of using partially disjoint paths are discussed in the following section.

MAXIMALLY DISJOINT PATHS

The maximally disjoint paths problem is formally defined as follows:

Problem 3 *Maximally disjoint paths problem:*
Given a directed network $G = (\mathcal{N}, \mathcal{L})$ of a set $\mathcal{N}$ of N nodes and a set $\mathcal{L}$ of L weighted links, two special nodes $s,t \in \mathcal{N}$, and an integer $k > 0$. Find k paths $P_1, P_2, ..., P_k$ from s to t, such that the paths share minimal common links (or nodes).

Fully disjoint paths may or may not exist in a sparse network. For a network to have k disjoint paths between nodes s and t, both nodes s and t must have at least k node degree (i.e., k neighbours). There must also exist enough nodes and links such that the k disjoint paths are possible. Maximally disjoint paths are useful if fully disjoint paths do not exist. A pair of paths is maximally disjoint if the number of (nodes) links common to both paths is minimum. For example, paths P_1 and P_2 both use link $(3,4)$ to go from node 1 to 4 as shown in Figure 7 since a completely disjoint path pair does not exist. However, maximally disjoint paths are still prone to simultaneous path failures, if the shared nodes or links fail. For example, if link $(3,4)$ fails, both P_1 and P_2 will fail.

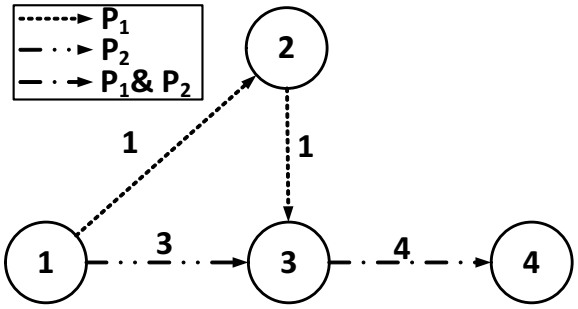


Figure 7: Example of two maximally disjoint paths

Omran et al. [20] show that the maximally disjoint paths problem is NP-hard, and hard to approximate within a factor of $2^{\log^{1-\varepsilon} N}$ for any constant $\varepsilon > 0$. They give a security-related routing scenario as an application to the maximally disjoint paths problem. A very important person needs to be transferred safely between two network nodes. To do this, a random path is selected from k pre-calculated maximally disjoint paths just before the travel commences. Any overlapping links of the k pre-calculated paths would need to have additional security measures, such as by placing security guards on the links. Since this would imply higher risk and cost, link overlapping needs to be minimized.

Castanon [21] proposes an efficient algorithm based on minimum cost flow algorithms [22] for finding the k best disjoint paths in a trellis graph. Compared to maximally disjoint paths that minimize the number of shared links (or nodes), best paths are paths that are as diverse as possible, i.e, minimize the total number of times links are shared [20]. The transformation of an arbitrary network to a trellis graph based on graph partitioning techniques is provided in [23], where a formal definition of a trellis graph is also provided. Lee and Wu [24] observe that the network transformation of [23] is but a heuristic, and propose a polynomial-time algorithm for finding k best paths in arbitrary networks based on network transformations and minimum cost flow algorithms.

Taft-Plotkin et al. [25] extend the Suurballe-Tarjan algorithm to efficiently return a pair (i.e., $k = 2$) of maximally disjoint paths. Their MADSWIP algorithm runs in $O(N \log N + L)$ time, similar to the Dijkstra algorithm and the Suurballe-

Tarjan algorithm. Instead of considering links with additive link weights, they also assume that each link is characterized by another restrictive link weight (e.g., bandwidth in a communication network). Restrictive link weights have flow-like characteristics where instead of being accumulated along a path, the minimum link weight of a link will decide whether the link is usable to the traffic. For example, path P_1 in Figure 8 requires a capacity c of two, so P_1 cannot use the link $(1,2)$ which would lead to a shorter path since link $(1,2)$ can only provide a capacity of one. The MADSWIP algorithm computes maximum-bandwidth maximally disjoint paths and minimizes the total path weights (i.e., min sum condition) as a secondary objective. If all links have the same restrictive link weight, the MADSWIP algorithm returns the min-sum maximally disjoint paths. The MADSWIP algorithm is usable in both directed and undirected networks.

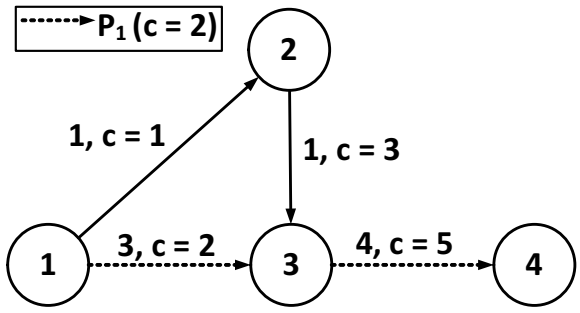


Figure 8: Restrictive link weight

For the general case when $k \geq 2$, Omran et al. propose a $k-1$ approximation algorithm based on network transformation techniques and conventional maximum flow algorithms [22] for the maximally link-disjoint paths problem and heuristics to improve the algorithm. Their network transformation technique would enable the use of conventional maximum flow algorithms to find maximally link-disjoint paths in the transformed network.

DOMAIN-DISJOINT PATHS

The domain-disjoint paths problem is formally defined as follows:

Problem 4 *Domain-disjoint paths problem:*

Given a directed network $G = (\mathcal{D}, \mathcal{T}, \mathcal{N}, \mathcal{L})$ of a set $\mathcal{D}$ of D domains, a set $\mathcal{T}$ of T weighted inter-domain links, a set $\mathcal{N}$ of N nodes, a set $\mathcal{L}$ of L weighted links, two special nodes $s,t \in \mathcal{N}$, and an integer $k > 0$. Each domain $d \in \mathcal{D}$ consists of a set of nodes $\mathcal{N}_d \subseteq \mathcal{N}$ and a set of links $\mathcal{L}_d \subseteq \mathcal{L}$ that interconnect the nodes in $\mathcal{N}_d$. Find k paths $P_1, P_2, ..., P_k$ from s to t, such that the paths share no common domains.

A domain is a set of nodes and links. Figure 9 illustrates a network with four domains that are interconnected by inter-domain links. A domain can represent an administrative region in optical networks, a province of cities in transportation networks, etc.

The domain-disjoint paths problem is NP-hard [26]. Optimal approaches of finding inter-domain shortest paths have been studied in recent years, particularly in the field of optical networking. When domains do not prefer to disclose their internal network topology to other domains, the topology of a domain can be abstracted [27], before paths are computed on a per-domain level. For example, domain 3 in Figure 9 can be abstracted using one of several aggregation models, as illustrated in Figure 10. The link weight of each intra-domain link will also be abstracted to the virtual link aggregation of the full mesh and star aggregation.

Full mesh aggregation tends to be the aggregation of choice due to the easiness in modeling the virtual connectivity between all border nodes within a domain. Each border node will have a virtual node degree equal to the number of border nodes in the domain. However, several virtual links interconnecting the border nodes may actually share similar physical links, giving a false impression of disjoint paths within the domain. The advantages, disadvantages and differences between the topology aggregation models are discussed in [28].

Topology aggregation of domains can be beneficial for the scalability of the information exchanged between domains, and for ensuring the confidentiality and security of domains. A complete physical view of domains could ease the act of sabotage or attack on the domains. Domains will then maintain and process smaller information size, enabling paths to be computed with less voluminous input size. However, using topology aggregation may also lead to sub-optimal path computation due to insufficient internal domain information.

Using the aggregated network topology, an inter-domain path will be computed from the source domain to the destination domain. Each domain will then need to refine the path segment that travels through its domain, e.g., finding the exact shortest path between the border nodes that are part of the inter-domain path. Finding inter-domain paths usually comes under the assumption that finding subsequent intra-domain paths is assured.

For finding link-disjoint paths or node-disjoint paths in an aggregated multi-domain topology, existing disjoint paths algorithms can be extended, as in [29] that uses an algorithm based on the Bhandari algorithm. However, the solution of [29] does not guarantee domain-disjoint paths, since its disjoint paths may or may not travel the same set of domains through different inter-domain links.

For domain-disjoint paths, it must be ensured that the k paths do not share any domain. One simple way of doing this

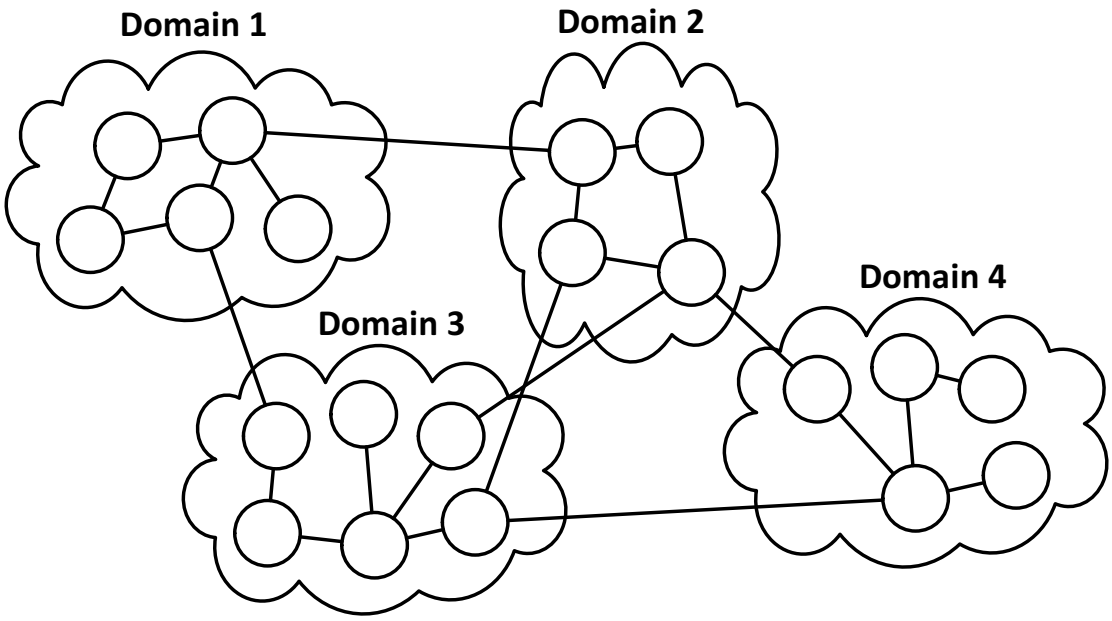


Figure 9: Example of a multi-domain network

is by aggregating every domain using the node aggregation model, compute the domain-disjoint paths using any existing node-disjoint paths algorithm, and later refine the inter-domain path segments of each domain. However, this approach cannot guarantee that the returned domain-disjoint paths are optimal since node aggregation is the aggregation model that shows the least information on the internal domain connectivity. An inter-domain path may then appear to be optimal at the domain level, but may actually use up a very long intra-domain path.

When other aggregation models are used, Gao et al. [26] propose the addition of cyclic structures to the domain that would enable the Bhandari algorithm to be applied directly in finding the domain-disjoint paths. Though intra-domain paths refining is still needed, the link weight abstractions included in the full mesh or star aggregation will lead to a better intra-domain path than when using the node aggregation. The cyclic structure ensures that each domain can only be traveled at most once by the k paths. The cyclic structure is made up of a directed cycle of nodes, connected by links with weight $-H$, where H is a very large cost. The number of cycle nodes of a domain equals the number of the border nodes of the domain. Each cyclic node is connected to two additional virtual nodes, an additional node with a directed link of weight 0 towards the cycle and an additional node with a directed link of weight $(B-1)\times H$ away from the cycle, where B is the number of border nodes of the domain. For example, the cyclic structure of domain 2 is shown in Figure 11. The virtual nodes are then connected to the aggregated domain topology.

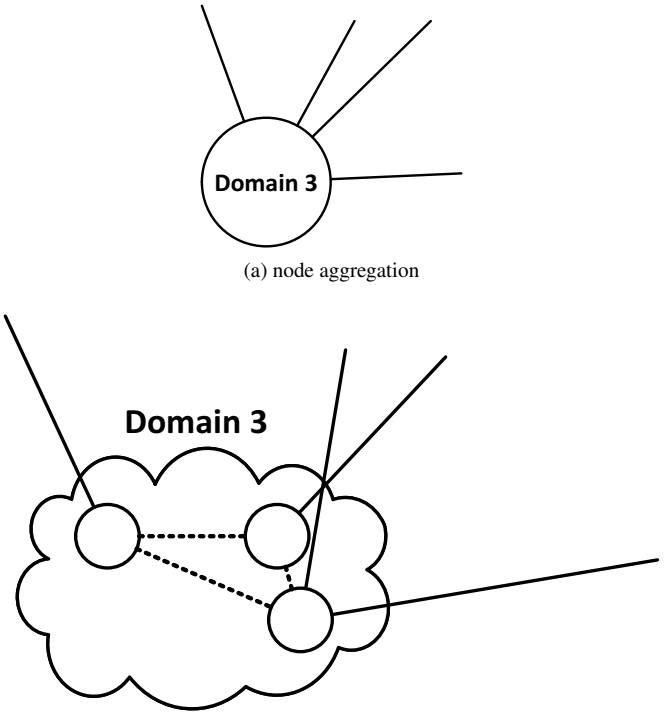
SHARED RISK LINK GROUP (SRLG)-DISJOINT PATHS

The shared risk link group (SRLG)-disjoint paths problem is formally defined as follows:

Problem 5 *Shared risk link group (SRLG)-disjoint paths problem:*

Given a directed network $G=(\mathcal{N},\mathcal{L})$ of a set $\mathcal{N}$ of N nodes, a set $\mathcal{L}$ of L weighted links, a set $\mathcal{R}$ of R risk groups, two special nodes $s,t\in\mathcal{N}$, and an integer $k>0$. Each link $(u,v)\in\mathcal{L}$ can belong to one or more risk groups. Find k paths $P_1,P_2,...,P_k$ from s to t, such that the paths share no common risk groups.

A shared risk link group (SRLG) is a group of links that share a component whose failure causes the failure of all links in the group [30]. A link can belong to multiple SRLGs. An example of such a component is the fiber conduit [31] in optical networks, where several optical links may be placed alongside in one single conduit, as illustrated in Figure 12a. Links $(1,2),(3,2)$ and $(3,4)$ are placed inside a similar conduit, while links $(3,2)$ and $(3,4)$ also share another similar conduit. Each conduit can be regarded as a SRLG. SRLG-disjoint paths share no common SRLG among themselves. The failure of a path due to a risk would not affect other paths as well. Other example applications of SRLGs are the correlated congestion of transportation networks and cascading failures of power grid networks [32].



(a) node aggregation

(b) full mesh aggregation

(c) star aggregation

Figure 10: Example of topology aggregation models

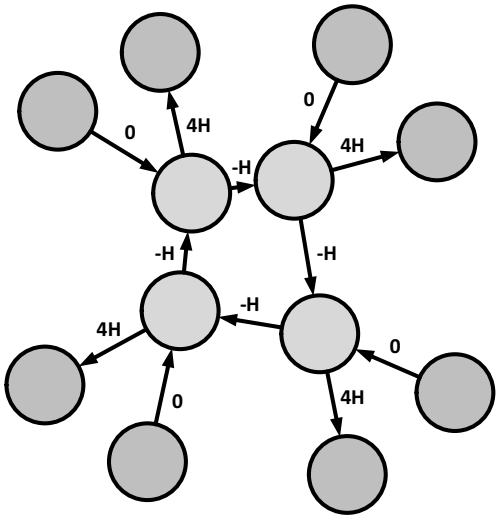
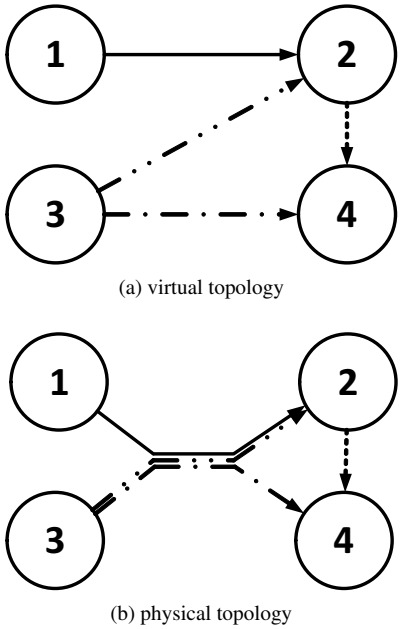


Figure 11: Example of a cyclic structure



(a) virtual topology

(b) physical topology

Figure 12: Example of shared risk link groups (SRLGs)

The SRLG-disjoint paths problem is NP-hard [33] and hard to approximate [32] in general, except in some specific instances such when SRLGs follow the star property (a link belongs to at most two SRLGs and two risks affecting the same link form stars at different nodes) and the number of SRLGs is constant [34], or when SRLGs follow the star property and the maximum node degree is at most four [34], or when when SRLGs follow the star property and the network is a directed acyclic graph [34], or in special span-sharing topologies [31]. [35] solves the problem when SRLG follows the star property by an algorithm based on network transformations, and [31] proposes an efficient algorithm based on the Bhandari algorithm to solve the problem in special span-sharing topologies.

Hu et al. [33] propose an Integer Linear Programming (ILP) formulation to solve the SRLG-disjoint paths problem. However, the running time of an exact algorithm can be too long and impractical when the network size is large or when the number of risk groups is large. The SRLG-disjoint paths problem can also be heuristically solved by a variant of the Iterative

DP algorithm that ensures the k paths do not share any SRLG. However, the trap topology problem is more likely to manifest (up to 30% more chance [36]) in the SRLG-disjoint paths problem when using sequential algorithms than in the disjoint paths problem. Xu et al. [36] propose a heuristic based on the Iterative DP algorithm that can avoid the trap topology in solving the SRLG-disjoint paths problem.

Instead of fully SRLG-disjoint paths, Rostami et al. [37] propose an algorithm based on ant colony optimization that finds a pair of maximally SRLG-disjoint paths. They also provide a good overview of research on SRLG-disjoint paths.

The concept of SRLG can also be extended to encompass the shared risk node group (SRNG) where instead of links, nodes belong to certain risk groups, as shown in Figure 13. Though we only illustrate the case when a node belongs to a risk group, there is no limit to the number of risk groups that a node could belong to. When nodes represent routers in a communication network, all routers of the same manufacturer would belong to a risk group since a bug affecting the router model would affect them all.

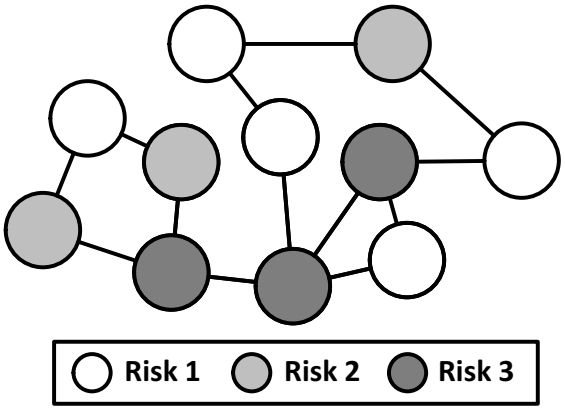


Figure 13: Example of shared risk node groups (SRNGs)

REGION-DISJOINT PATHS

The region-disjoint paths problem is formally defined as follows:

Problem 6 *Region-disjoint paths problem:*

Given a network $G = (\mathcal{N}, \mathcal{L})$ of a set $\mathcal{N}$ of N nodes and a set $\mathcal{L}$ of L weighted links that is embedded in a two-dimensional plane, and two special nodes $s,t \in \mathcal{N}$, a diameter $D > 0$, and an integer $k > 0$. Find k paths $P_1, P_2, ..., P_k$ from s to t, such that the paths cannot be affected by a single regional failure of diameter D (unless the failure includes s or t).

Each intermediate node of any of the region-disjoint paths must be on a distance greater than D from every intermediate node of all the other region-disjoint paths. For example, the weight of the links of the network shown in Figure 14 represents the distance in kilometer between its adjacent nodes. Consider the problem of finding a pair of region-disjoint paths from node 1 to node 6 with $D = 15$ km. Since the distance between links $(2,4)$ and $(3,5)$ is less than D, paths P_1 and P_2 are not a viable solution to the problem.

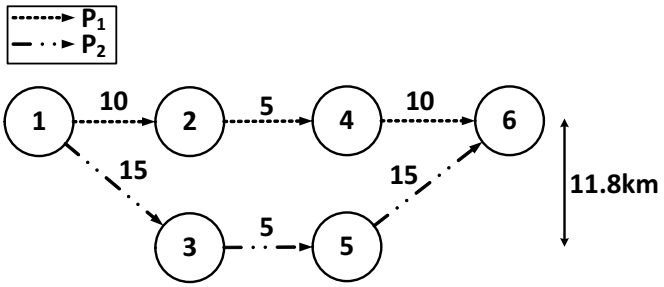


Figure 14: Example of a pair of paths that are not region-disjoint when $D = 15$ km

The region-disjoint paths problem is shown to be NP-hard (even for $k = 2$) and hard to approximate in [38] for most instances of the problem, but might be polynomially solvable in certain instances, e.g., if all network nodes are pairwise on a distance greater than D. The disaster area shape and characteristics also play an important role in shaping the algorithms for solving the region-disjoint paths problem.

Agarwal et al. [39] study the problem of finding the most vulnerable region in a network by assuming that a disaster would have an epicenter, and the probability of a node or link to be affected depends on its distance from the epicenter. They propose an approximation algorithm to find the most vulnerable region.

Neumayer et al. [40] assume that all the nodes and links within a disaster area (they model a disaster area also with a circular plane with an epicenter) fail. They present an ILP formulation with a polynomial number of constraints, and greedy exact and heuristic algorithms that consider only a subset of possible paths to solve the region-disjoint paths problem in [41].

Instead of using an epicenter, the work of Trajanovski et al. [38] is unique in such a way that the shape of a region is not confined to circular shapes as in [40], but they also consider ellipses and general polygons. Another difference is that Trajanovski et al. assume that a regional failure would affect only all the nodes in the region and all the links that are connected to those nodes. Links that pass the region but are not connected to any nodes in the region will not be affected by the region failure. Trajanovski et al. then proceed with a heuristic

algorithm based on the Suurballe-Tarjan algorithm to solve the region-disjoint paths problem.

DISJOINT PATH PAIRS

The disjoint path pairs problem is formally defined as follows:

Problem 7 *Disjoint Path Pairs problem:*
Given a directed network $G = (\mathcal{N}, \mathcal{L})$ of a set $\mathcal{N}$ of N nodes and a set $\mathcal{L}$ of L weighted links, an integer $k > 0$, and k pairs of nodes (s_k, t_k) where $s_k, t_k \in \mathcal{N}$. Find a path from each s_k to its corresponding node pair t_k, such that the paths share no common links (or nodes).

The disjoint path pairs problem is reduced to the disjoint paths problem when $s_1 = ... = s_k = s$ and $t_1 = ... = t_k = t$, where efficient algorithms exist. The disjoint path pairs problem is NP-hard in general directed networks for $k \geq 2$ [42], but is solvable in polynomial time in undirected networks [43], directed planar networks [44], and directed acyclic networks [42].

Robertson and Seymour [43] give an $O(n^3)$ time algorithm to solve the disjoint path pairs problem in undirected networks, though their algorithm involves the use of enormous constants, which can be impractical. Kawarabayashi et al. [45] then presented a faster $O(n^2)$ time algorithm based on the Roberson-Seymour algorithm for the link-disjoint path pairs problem.

However, if k is to be maximized instead of fixed, the disjoint path pairs problem becomes NP-hard [46]. A sample application of the disjoint path pairs problem is given in [45], where certain prescribed channels on a chip need to be interconnected without any wires of different pins coming into contact with each other. For example, the disjoint path pairs P_1, P_2, P_3 and P_4 shown in Figure 15 are all link-disjoint.

Similar to the disjoint paths problem, several additional conditions have also been studied for the disjoint path pairs problem, namely the min-sum and the min-max conditions.

Min-sum disjoint path pairs problem - the sum of the weights of all the constituent links of the k paths is minimized.

Min-max disjoint path pairs problem - the sum of the weights of all the constituent links of the path with the largest path weight is minimized.

The NP-hardness of the min-sum condition when $k \geq 2$ has been open for more than three decades [47]. Zhang and Zhao [48] show that the min-sum condition is FP^{NP}-complete for directed and undirected networks with weighted links, and that both the min-sum and min-max conditions cannot be approximated within $\Omega(L^{1-\varepsilon})$ for any constant $\varepsilon > 0$ for directed and undirected networks with unweighted links. They give a bi-criteria approximation algorithm for the problem. Brandes et al. [49] prove that the min-sum and the min-max conditions are NP-hard in planar networks, even if the network fulfils the Eulerian condition and the maximum degree is four. However, the min-sum condition is polynomially solvable in some specific instances of the problem. Kammer and Tholey [50] show that the min-sum condition can be solved in polynomial time if G is a chordal network. Kobayashi and Sommer [51] show that the min-sum condition is polynomially solvable when $k = 2$ for node pairs adjacent to at most two faces in a planar network. They also show that the min-sum condition is also polynomially solvable when $k = 3$ for six node pairs adjacent to one face in any order in a planar network.

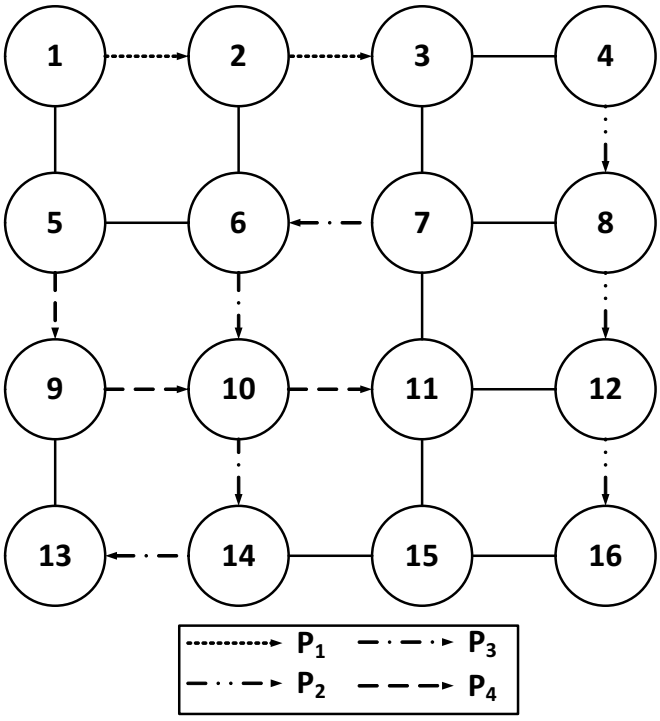


Figure 15: Example of disjoint path pairs

SUMMARY

This paper presents the basic concepts, complexity analysis, and discussions on the disjoint paths problem. The disjoint paths problem has a wide range of applications across various network types, particularly in providing a more reliable service to network traffic. Traffic that can make use of disjoint paths is more robust to the effect of network node or link failures. This paper also covers many different variants of the disjoint paths problem, namely the availability-based disjoint paths problem, the maximally disjoint paths problem, the domain-disjoint paths problem, the shared risk link group (SRLG)-

Table 1: Summary.

Problem	Purpose	Example application	Complexity
Disjoint paths	Paths share no common links or nodes.	Providing different candidate paths for truck drivers in transportation networks.	Polynomially solvable (e.g., by Bhandari algorithm [13]).
Availability-based disjoint paths	Paths share no common links or nodes, with an availability constraint.	Improving the connection availability of traffic in telecommunication networks.	Polynomially solvable (e.g., by Bhandari algorithm [13]).
Maximally disjoint paths	Paths share minimal common links or nodes.	Transferring a very important person safely between two locations.	NP-hard and hard to approximate within a factor of $2^{\log^{1-\varepsilon} N}$ for any constant $\varepsilon > 0$ [20]. However, when $k = 2$, the problem is polynomially solvable (e.g., by MADSWIP algorithm [25]).
Domain-disjoint paths	Paths share no common domains.	Improving the transmission reliability of traffic across multiple optical network domains.	NP-hard [26].
Shared risk link group (SRLG)-disjoint paths	Paths share no common risk groups.	Ensuring different disjoint paths do not use links belonging to similar conduits in optical networks.	NP-hard [33] and hard to approximate [32], except when SRLGs follow the star property and the number of SRLGs is constant [34], or when SRLGs follow the star property and the maximum node degree is at most four [34], or when when SRLGs follow the star property and the network is a directed acyclic graph [34], or in special span-sharing topologies [31].
Region-disjoint paths	Paths cannot be affected by a single regional failure of diameter D (unless the failure includes the source or the destination node).	Ensuring that a regional disaster would not affect different disjoint paths simultaneously.	NP-hard and hard to approximate [38], except if all network nodes are pairwise on a distance greater than D.
Disjoint path pairs	Paths share no common links or nodes (paths may have different source and destination nodes).	Interconnecting prescribed channels on a chip without any wires of different pins coming into contact with each other.	NP-hard in general directed networks for $k \geq 2$ [42], but is polynomially solvable in undirected networks [43], directed planar networks [44], and directed acyclic networks [42].

disjoint paths problem, the region-disjoint paths problem, and the disjoint path pairs problem.

A summary of the variants of the disjoint paths problems considered is presented in Table 1. Each problem is classified according to its purpose, an application example and complexity.

BIBLIOGRAPHY

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[2] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, pp. 87–90, 1958.

[3] L. Ford and D. R. Fulkerson, *Flows in networks*. Princeton Princeton University Press, 1962.

[4] D. Zhou and S. Subramaniam, "Survivability in optical networks," *IEEE Netw.*, vol. 14, no. 6, pp. 16–23, 2000.

[5] F. A. Kuipers, "An overview of algorithms for network survivability," *ISRN Commun. Netw.*, vol. 2012, p. 24, 2012.

[6] 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.

[7] L. Guo and H. Shen, "On finding min-min disjoint paths," *Algorithmica*, vol. 66, no. 3, pp. 641–653, 2013.

[8] D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He, "On the complexity of and algorithms for finding the shortest path with a disjoint counterpart," *IEEE/ACM Transactions on Netw.*, vol. 14, no. 1, pp. 147–158, 2006.

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

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

[11] A. Beshir and F. A. Kuipers, "Variants of the min-sum link-disjoint paths problem," in *IEEE Proc. Annu. Symp. Commun. & Vehicular Tech. (SCVT'09)*, 2009.

[12] D. A. Dunn, W. D. Grover, and M. H. MacGregor, "Comparison of k-shortest paths and maximum flow routing for network facility restoration," *IEEE J. Selec. Areas in Commun.*, vol. 12, no. 1, pp. 88–99, 1994.

[13] R. Bhandari, "Optimal physical diversity algorithms and survivable networks," in *IEEE Proc. Symp. Comput. & Commun.*, 1997, pp. 433–441.

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

[15] Y. Guo, F. Kuipers, and P. Van Mieghem, "Link-disjoint paths for reliable QoS routing," *Int. J. Commun. Syst.*, vol. 16, no. 9, pp. 779–798, 2003.

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

[17] M. Clouqueur and W. D. Grover, "Availability analysis of span-restorable mesh networks," *IEEE J. Selec. Areas in Commun.*, vol. 20, no. 4, pp. 810–821, 2002.

[18] M. To and P. Neusy, "Unavailability analysis of long-haul networks," *IEEE J. Selec. Areas in Commun.*, vol. 12, no. 1, pp. 100–109, 1994.

[19] S. Yang, S. Trajanovski, and F. A. Kuipers, "Availability-based path selection," in *Proc. Int. Workshop on Reliable Netw. Design & Modeling (RNDM'14)*, 2014.

[20] M. T. Omran, J.-R. Sack, and H. Zarrabi-Zadeh, "Finding paths with minimum shared edges," *J. Combinatorial Opt.*, vol. 26, no. 4, pp. 709–722, 2013.

[21] D. A. Castanon, "Efficient algorithms for finding the k best paths through a trellis," *IEEE Trans. Aerospace & Electronic Syst.*, vol. 26, no. 2, pp. 405–410, 1990.

[22] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*. Prentice hall, 1993.

[23] S. D. Nikolopoulos, A. Pitsillides, and D. Tipper, "Addressing network survivability issues by finding the k-best paths through a trellis graph," in *IEEE Conf. Comput. & Commun. (INFOCOM'97)*, vol. 1, 1997, pp. 370–377.

[24] S.-W. Lee and C.-S. Wu, "A k-best paths algorithm for highly reliable communication networks," *IEICE Trans. Commun.*, vol. 82, no. 4, pp. 586–590, 1999.

[25] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-service routing using maximally disjoint paths," in *IEEE Int. Workshop on Quality of Service (IWQoS'99)*, 1999, pp. 119–128.

[26] C. Gao, M. M. Hasan, and J. P. Jue, "Domain-disjoint routing based on topology aggregation for survivable multidomain optical networks," *OSA J. Opt. Commun & Netw*, vol. 5, no. 12, pp. 1382–1390, 2013.

[27] W. C. Lee, "Topology aggregation for hierarchical routing in ATM networks," *Comp. Commun. Rev.*, vol. 25, no. 2, pp. 82–92, 1995.

[28] S. Uludag, K.-S. Lui, K. Nahrstedt, and G. Brewster, "Analysis of topology aggregation techniques for QoS routing," *ACM Comput. Surveys*, vol. 39, no. 3, p. 7, 2007.

[29] A. Sprintson, M. Yannuzzi, A. Orda, and X. Masip-Bruin, "Reliable routing with QoS guarantees for multi-domain IP/MPLS networks," in *IEEE Conf. Comput. & Commun. (INFOCOM'07)*, 2007, pp. 1820–1828.

[30] P. Sebos, J. Yates, G. Hjalmtysson, and A. Greenberg, "Auto-discovery of shared risk link groups," in *OSA Opt. Fiber Commun. Conf.*, vol. 4, 2001.

[31] R. Bhandari, "Optimal diverse routing in telecommunication fiber networks," in *IEEE Proc. Int. Conf. Comput. Commun. (INFOCOM'94)*, 1994, pp. 1498–1508.

[32] D. Coudert, P. Datta, S. Pérennes, H. Rivano, and M.-E. Voge, "Shared risk resource group complexity and approximability issues," *Parallel Processing Letters*, vol. 17, no. 02, pp. 169–184, 2007.

[33] J. Q. Hu, "Diverse routing in optical mesh networks," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 489–494, 2003.

[34] J. Bermond, D. Coudert, G. D'Angelo, and F. Z. Moataz, "SRLG-diverse routing with the star property," in *Int. Conf. Design of Reliable Commun. Netw. (DRCN'13)*, 2013, pp. 163–170.

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

[36] D. Xu, Y. Xiong, and C. Qiao, "A new PROMISE algorithm in networks with shared risk link groups," in *IEEE Global Telecommunications Conf. (GLOBECOM'03)*, vol. 5, 2003, pp. 2536–2540.

[37] M. J. Rostami, A. A. E. Zarandi, and S. M. Hoseininasab, "MSDP with ACO: A maximal SRLG disjoint routing algorithm based on ant colony optimization," *J. Netw. Comput. Applications*, vol. 35, no. 1, pp. 394–402, 2012.

[38] S. Trajanovski, F. A. Kuipers, A. Ilic, J. Crowcroft, and P. Van Mieghem, "Finding critical regions and region-disjoint paths in a network," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, 2015.

[39] P. K. Agarwal, A. Efrat, S. K. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman, "The resilience of WDM networks to probabilistic geographical failures," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, 2013.

[40] S. Neumayer, G. Zussman, R. Cohen, and E. Modiano, "Assessing the vulnerability of the fiber infrastructure to disasters," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1610–1623, 2011.

[41] S. Neumayer, A. Efrat, and E. Modiano, "Geographic max-flow and min-cut under a circular disk failure model," in *IEEE Conf. Comput. & Commun. (INFOCOM'12)*, 2012, pp. 2736–2740.

[42] S. Fortune, J. Hopcroft, and J. Wyllie, "The directed subgraph homeomorphism problem," *Theoretical Comput. Sci.*, vol. 10, no. 2, pp. 111–121, 1980.

[43] N. Robertson and P. D. Seymour, "Graph minors. XIII. the disjoint paths problem," *J. Combinatorial Theory, Series B*, vol. 63, no. 1, pp. 65–110, 1995.

[44] A. Schrijver, "Finding k disjoint paths in a directed planar graph," *SIAM J. Comput.*, vol. 23, no. 4, pp. 780–788, 1994.

[45] K. Kawarabayashi, Y. Kobayashi, and B. Reed, "The disjoint paths problem in quadratic time," *J. Combinatorial Theory, Series B*, vol. 102, no. 2, pp. 424–435, 2012.

[46] R. M. Karp, "On the complexity of combinatorial problems," *Netw.*, vol. 5, pp. 45–68, 1975.

[47] T. Fenner, O. Lachish, and A. Popa, "Min-sum 2-paths problems," *Lecture Notes in Comput. Sci.*, pp. 1–11, 2014.

[48] P. Zhang and W. Zhao, "On the complexity and approximation of the min-sum and min-max disjoint paths problems," in *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Springer, 2007, pp. 70–81.

[49] U. Brandes, G. Neyer, and D. Wagner, "Edge-disjoint paths in planar graphs with short total length," *Konstanzer Schriften in Mathematik und Informatik*, vol. 19, 1996.

[50] F. Kammer and T. Tholey, "The k-disjoint paths problem on chordal graphs," in *Graph-Theoretic Concepts in Comput. Sci.* Springer, 2010, pp. 190–201.

[51] Y. Kobayashi and C. Sommer, "On shortest disjoint paths in planar graphs," *Discrete Optimization*, vol. 7, no. 4, pp. 234–245, 2010.