Network Interdiction via a Critical Disruption Path: Branch-and-Price Algorithms

Donatella Granata

Institute for Application of Calculation, "Mauro Picone," National Research Council, Naples, Italy

Gregory Steeger

Colorado School of Mines, Division of Economics and Business, Golden, Colorado

Steffen Rebennack*

Colorado School of Mines, Division of Economics and Business, Golden, Colorado

Abstract

This paper presents an innovative approach to maximally disconnect a given network. More specifically, this work introduces the concept of a *critical disruption path*, a path between a source and a destination vertex whose deletion minimizes the cardinality of the largest remaining connected component. Network interdiction models seek to optimally disrupt network operations. Existing interdiction models disrupt network operations by removing vertices or edges. We introduce the first problem and formulation that optimally fragments a network via interdicting a path. Areas of study in which this work can be applied include transportation and evacuation networks, surveillance and reconnaissance operations, anti-terrorism activities, drug interdiction, and counter human-trafficking operations. In this paper, we first address the complexity associated with the critical disruption path problem, and then provide a mixed-integer linear programming formulation for finding its optimal solution. Further, we develop a tailored Branch-and-Price algorithm that efficiently solves the critical disruption path problem. We demonstrate

*Corresponding author

Email addresses: d.granata@na.iac.cnr.it (Donatella Granata), gsteeger@mines.edu (Gregory Steeger), srebenna@mines.edu (Steffen Rebennack)

the superiority of the developed Branch-and-Price algorithm by comparing the results found via our algorithm with the results found via the monolith formulation. In more than half of the test instances that can be solved by both the monolith and our Branch-and-Price algorithm, we outperform the monolith by two orders of magnitude.

Keywords:
Network interdiction, mixed-integer linear programming, NP-completeness, Branch-and-Price, cuts

---

1. Introduction

Many real-world systems can be represented as networks and recent advances in computational power enable modelers to study larger, more realistic applications using networks [1]. Network interdiction models explore techniques to inhibit network operations. In the most general sense, network interdiction refers to an attack that leaves a network fragmented or disconnected. An interdiction event may either cause the network to become non-operational, in which case entities can no longer flow through the network from a source vertex to a sink vertex, or to some extent degraded so that either fewer entities can flow through the network, or so that it takes the entities longer to flow through the network. For instances in which the network remains operational, after an interdiction event, we may desire to evaluate the network's communication capability or level of degradation. We may also desire to provide an assessment on the network's vulnerability to failure.

Given a network $\mathcal{G} = (V, E, s, t)$, where V is the set of vertices and E is the set or edges, we desire the best way possible to destroy the network's connectivity. We destroy network $\mathcal{G}$'s connectivity by finding and removing a simple path from $s \in V$ to $t \in V$ such that the number of vertices in the largest remaining connected component is minimized. We term such a path a *Critical Disruption Path (CDP)*. Note that this path may not be unique.

CDPs are most relevant for applications in which exploiting (or assessing) a network's vulnerability is best achieved through finding a path in the network whose removal maximally impedes network operations. For instance, CDPs can be used to determine flight patterns for military surveillance and reconnaissance missions using remotely piloted, or traditional aircraft. The aircraft could fly close enough along a CDP of the network to gather the best

possible intelligence. Better intelligence produces more effective military operations. Additionally, operations focused on CDPs, in networks, may prove more effective than existing procedures. For example, to disrupt terrorist networks, current operations typically focus on eliminating key leaders (vertices), whereas larger disruptions may result from operations that fragment the terrorist network so that it is no longer connected and the size of the largest remaining terrorist cell (connected component) is minimized and this is best achieved via the removal of a CDP.

The CDP detection problem is the first interdiction model that fragments a network by removing an $s - t$ path, rather than vertices or edges. As such, the CDP detection problem fills a gap in the rich body of literature on interdiction models (see Section 2). This work also marks the first time that a Branch-and-Price (B&P) algorithm has been applied to network interdiction models.

The contributions of this work are threefold:

(1) introduction of the Critical Disruption Path detection problem

(2) proof of NP-completeness for the CDP detection problem

(3) development of a tailored Branch-and-Price algorithm for solving the CDP detection problem

Section 2 provides a literature review on network interdiction methods and measures. In Section 3, we state some basic definitions, present our notation, and prove the problem's computational complexity. Section 4 introduces the monolith for the CDP detection problem: a Mixed-Integer Linear Programming (MILP) formulation. Next, in Section 5, we discuss the basics of the B&P algorithm. We analyze the intricacies of our tailored B&P algorithm in greater detail in Section 6. Section 7 states our computational results. Finally, Section 8 delivers our conclusions.

2. Literature Review

We now present a review of the literature on network interdiction (see Table 1). In the interest of brevity, we only discuss the portion of this literature that is most closely related to our work. To quantify how well a given network is fragmented, we must be able to measure the connectivity or disconnectivity of the resulting network after removing vertices, edges, or a

path. Network connectivity is measured in many different ways, depending on the application.

Earliest efforts in network interdiction seek to reduce the flow through the network by removing edges. Rather than disrupting the network by minimizing the flow, Israeli and Wood [2] seek the set of edges to interdict in order to maximize the length of the shortest path. Royset and Wood [3] solve a slightly different problem in which the interdictor seeks to destroy a set of edges in a capacitated network, while minimizing both total interdiction costs and maximum flow.

In an effort to destroy cohesion between players (vertices) in a network, Borgatti [4] presents two models that fragment a network by removing vertices. Arulselvan et al. [5, 6] present another instance in which vertices are removed to disrupt a given network. Arulselvan et al. study what they term the Critical Node Problem (CNP), where the objective is to find a set of k vertices whose deletion maximizes the resulting number of disconnected components. Borgatti [4] minimizes two measures of graph fragmentation: the number of components, and the number of pairs of vertices that are disconnected from each other. In a slightly different context, Arulselvan et al. [5] study how to manage risk in networks. In this context, critical vertices, vertices whose removal cause the most network disruption, are identified. Networks are considered less vulnerable when more vertices must be deleted to adequately fragment the network. Dinh et al. [7] study network vulnerability based on what they term overall-pairwise connectivity. The measure Addis et al. employ counts the number of pairs of vertices that are still connected by at least one path in the graph that remains (the residual graph) following vertex deletion.

Last, Shen et al. [9] examine three connectivity metrics: the number of connected components, the size of the largest component, and the minimum cost required to reconnect the graph after vertices are deleted. To best disrupt a given network, Shen et al. either maximize the number of connected components, minimize the largest component's size, or maximize the minimum cost required to reconnect the graph. By removing a CDP, we seek to minimize the largest remaining component's size. Shen et al. [9] present the work that is the most closely related to our work. Though both models minimize the size of largest connected component, they achieve this minimization via different methodologies. Shen et al. [9] remove a predetermined number of nodes to destroy a given network's connectivity, whereas we optimally destroy a network's connectivity by removing a path that minimizes the size of

the largest connected component. To determine which model or formulation to use, one must consider the specific application being studied.

Reference	Approach		Contribution(s) & Methodology
	Vertices	Edges	
Harris and Ross [10], 1955	–	–	seminal work, flow capacity estimation
Ford and Fulkerson [11], 1956	–	✓	first application of max-flow min-cut theorem
Ford and Fulkerson [12], 1957	–	✓	algorithm for max-flow problem
Wollmer [13], 1964	–	✓	minimize max-flow in capacitated network (LP)
McMasters and Mustin [14], 1970	–	✓	capacitated supply networks (LP)
Ball et al. [15], 1989	–	✓	interdiction with resource constraints
Wood [16], 1993	–	✓	models solving [15] (IP)
Whiteman [17], 1996	–	✓	tool to meet desired flow restrictions (IP)
Israeli and Wood [2], 2002	–	✓	max length of shortest path (MILP & Benders)
Myung and Kim [18], 2004	–	✓	minimize number of connections (MILP)
Borgatti [4], 2006	✓	–	minimize player cohesion (combinatorial)
Lim and Smith [19], 2007	–	✓	discrete or continuous interdiction (MILP)
Royset and Wood [3], 2007	–	✓	bi-objective max-flow (MILP & Lagrangian)
Arulselvan et al. [5], 2007	✓	–	CNP: maximize # of disconnected components (IP)
Arulselvan et al. [6], 2009	✓	–	minimize pair-wise connectivity (IP)
Dinh et al. [7], 2010	✓	✓	degrade connectivity to a given extent (MILP)
Addis et al. [8], 2011	✓	–	minimize # of connected vertices (DP)
Di Summa et al. [20], 2012	✓	–	branch-and-cut approach to CNP
Shen et al. [9], 2012	✓	–	MILP models for CNP, connectivity measures

Table 1: *Network Interdiction Literature. This table shows the literature related to network interdiction modeling. Columns 2 and 3 show the approach used to fragment the network (*i.e.*, node removal or arc removal). The last column lists the major contribution(s) of the work and methods used. LP = Linear Programming, IP = Integer Programming, MILP = Mixed-Integer Linear Programming, CNP = Critical Node Problem, DP = Dynamic Programming*

3. Notation and Computational Complexity

Given an undirected network $\mathcal{G} = (V, E, s, t)$, we denote its vertex and edge sets by V and E, respectively. We distinguish two special vertices in the network $\mathcal{G}$: a *source* $s \in V$ and a *destination* $t \in V$. We assume $\mathcal{G}$ to be a digraph: $(i, j) \in E$ implies that $(j, i) \in E$.

A *simple path* p in $\mathcal{G}$ is a sequence of distinct vertices $v_1, v_2, v_3, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for each i, $1 \leq i \leq k-1$ without repetition of vertices. For brevity, throughout this paper, the term *path* refers to a simple path. We denote the set of vertices in the path p by $V(p)$ and the set of edges in the path p by $E(p)$. Define the length of path p to be the number of vertices in p, *i.e.*, $|p| = |V(p)|$. For a path $p = (v_1, v_2, v_3, \ldots, v_k)$, we call the *left endpoint* the first vertex v_1, and the *right endpoint* the last vertex v_k. Throughout this paper, all the paths considered have s and t as the left and right endpoints, respectively. A small letter p denotes a single path, while $\mathcal{P}$ denotes a collection of paths.

A network $\mathcal{G}' = (V', E')$ is a *subgraph* of $\mathcal{G} = (V, E)$, if $V' \subseteq V$ and $E' \subseteq E$. $\mathcal{G}' = (V', E')$ is the subgraph of $\mathcal{G}$ *induced* by V' if E' contains each edge of E, with both endpoints in V'. The largest connected subgraphs of a disconnected network are its *connected components*.

With this notation in place, we are ready to formally define a Critical Disruption Path, the Critical Disruption Path Detection (CDPD) problem, and its decision version: the k-Critical Disruption Path Detection (k-CDPD) problem.

Definition 1 (*Critical Disruption Path (CDP)*).
A *Critical Disruption Path (CDP)*, p, in $\mathcal{G} = (V, E, s, t)$ is a simple path from s to t such that the largest connected component of the induced subgraph $\mathcal{G}^p := (V^p, E^p)$ where $V^p := V \setminus V(p)$, $E^p := E \cap (V^p \times V^p)$, obtained via deletion of p, contains the smallest number of vertices among all $s - t$ paths in $\mathcal{G}$.

Note that there may be multiple CDPs in a network. We term the problem in which we seek a CDP of a network the *Critical Disruption Path Detection* (CDPD) problem. Following the style of [21], the decision version of CDPD is stated as follows:

Definition 2 (k-*Critical Disruption Path Detection (k-CDPD)*).

INSTANCE: Undirected network $\mathcal{G} = (V, E, s, t)$ and non-negative integer k.

QUESTION: Is there a CDP, p, in $\mathcal{G}$ such that the cardinality of each connected component in the induced subgraph $\mathcal{G}^p$ is less than or equal to k?

To illustrate the CDPD problem, we consider the network given in Figure 1. Initially, one may think a CDP of a network is simply its longest path, however this is not necessarily the case. The network given in Figure 1 has twelve vertices

$$V = \{s, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, t\},$$

and thirteen edges

$$E = \{(s, 1), (s, 6), (1, 2), (1, 4), (2, 3), (2, 4), (2, 5), (4, 10), (6, 7), (7, 8), (8, 9), (9, 10), (10, t)\}.$$

In this example, if we remove the longest path $p = \{s, 6, 7, 8, 9, 10, t\}$ (*case b*) the largest remaining connected component has size 5. Whereas, after removing the CDP $p' = \{s, 1, 2, 4, 10, t\}$ (*case c*), the largest component in the residual graph has size 4.

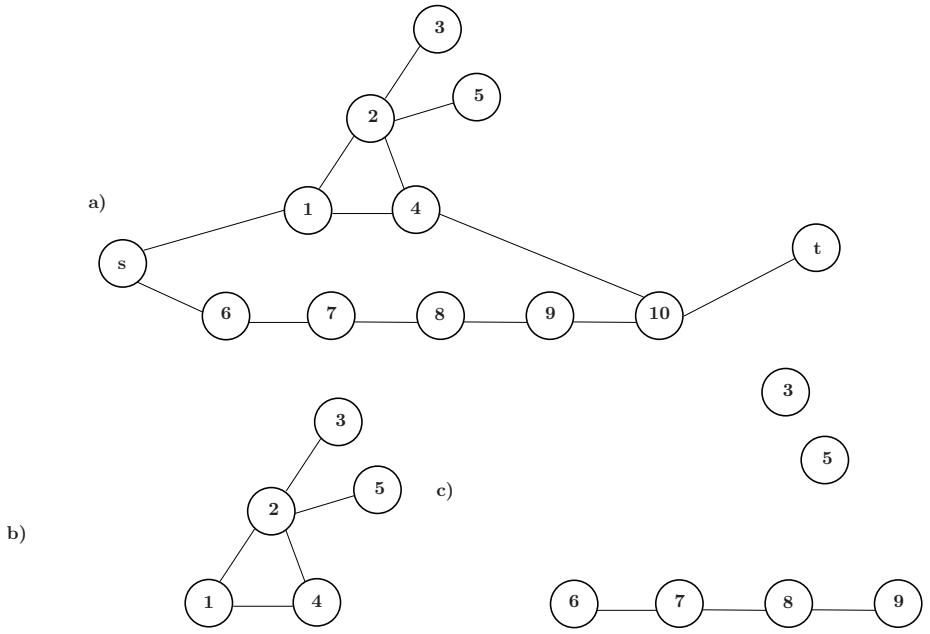


Figure 1: *a) Graph $\mathcal{G} = (V, E, s, t)$. b) Residual graph after elimination of the longest $s-t$ path. In this case, the largest connected component has size 5. c) Residual graph after removal of CDP $p' = \{s, 1, 2, 4, 10, t\}$. In this case two connected components remain: one of size 4, and two of size 1.*

Next, we prove that the CDPD problem is NP-hard. In doing so, we prove the NP-completeness of its corresponding decision version, k-CDPD, by reduction from the Hamiltonian Path problem, which is known to be (strongly) NP-complete [21].

Definition 3 (*Hamiltonian Path*).

INSTANCE: Undirected network $\mathcal{G} = (V, E, s, t)$.

QUESTION: Is there an $s - t$ path which visits every vertex in $\mathcal{G}$ exactly once?

Since a non-deterministic algorithm requires polynomial time to determine whether the deletion of a path p from s to t produces connected components with cardinality at most k, k-CDPD is in NP.

Theorem 1. *The k-CDP Detection problem is NP-complete.*

Consider network $\mathcal{G}' = (V', E', s, t)$ created from an arbitrary instance $\mathcal{G} = (V, E, v_s, v_t)$ of the Hamiltonian Path problem, with $k = |V|$. The network $\mathcal{G}'$ is constructed so that it contains a feasible CDP, from s to t, whose deletion produces connected components with cardinality equal to k, if and only if there is a Hamiltonian Path from v_s to v_t of length k in $\mathcal{G}$. To construct $\mathcal{G}'$ we

1. create two dummy vertices s and t
2. create edges (s,v_s) and (v_t,t)
3. create a clique of size $k + 1$ for each vertex $v \in V$

An example of this construction is shown in Figure 2. This construction is accomplished in polynomial time.

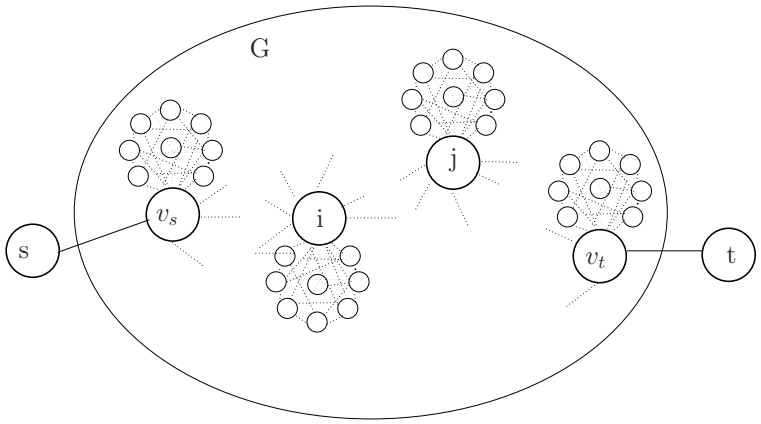


Figure 2: *Graph $\mathcal{G}'$ obtained from graph $\mathcal{G}$*

Given an instance $\mathcal{G} = (V, E, v_s, v_t)$ of the Hamiltonian Path problem, we seek to determine if there is a path p_{HP} from v_s to v_t in $\mathcal{G}'$, with $|p_{\text{HP}}| = k$,

which visits every vertex $v \in V$ exactly once. This is equivalent to determining if there exists a feasible solution (or CDP) p for the k-CDPD problem, whose deletion creates $|V|$ connected components, all of size k. We formalize this observation in the following proof of Theorem 1.

Proof $\Rightarrow$ Let $p_{\text{HP}} = \{v_s, \ldots, v_t\}$ be a Hamiltonian Path between v_s and v_t. We construct the path $p_{\text{CDP}} = \{s\} \cup p_{\text{HP}} \cup \{t\}$ from s to t by selecting only the edges that belong to p_{HP} and the edges (s, v_s) and (v_t, t). The deletion of p_{CDP} removes all the vertices $v \in V$ and reduces the size of each clique constructed on each vertex by one unit This, in turn, produces connected components with cardinality equal to k.

$\Leftarrow$ If the k-CDPD problem is a yes instance for $\mathcal{G}'$, then there is a CDP $p_{\text{CDP}} = \{s, v_s, \ldots, v_t, t\}$, from s to t, whose deletion produces connected components with cardinality k, and by construction the path from v_s to v_t is a Hamiltonian Path. □

The remainder of the paper focuses on exact solution methods for the CDPD problem.

4. Mixed-Integer Linear Programming Formulation: Monolith

The CDPD problem is closely related to both the longest path problem and the Hamiltonian cycle problem. This relationship is portrayed in the MILP formulation below, which is based on popular formulations for the Traveling Salesman Problem (TSP), *cf.* [22]. Since this formulation ensures a path is created by making use of Sub-tour Elimination Constraints (SEC), we term this model CDP-SEC.

CDP-SEC:

$$z^* := \min \kappa$$

$$\text{s.t.} \sum_{j:(i,j)\in E} x_{ij} = \sum_{h:(h,i)\in E} x_{hi} \qquad \forall i \in V \setminus \{s,t\} \tag{1}$$

$$\sum_{j:(s,j)\in E} x_{sj} = 1 \tag{2}$$

$$\sum_{i:(i,t)\in E} x_{it} = 1 \tag{3}$$

$$\sum_{i,j\in S:\,(i,j)\in E} x_{ij} \leq |S| - 1 \qquad \forall S \subset V, \quad |S| \geq 2 \tag{4}$$

$$y_{ii} = 1 - \sum_{l:(l,i)\in E} x_{li} \qquad \forall i \in V \setminus \{s,t\} \tag{5}$$

$$y_{ik} \geq y_{ih} - \sum_{l:(l,k)\in E} x_{lk}$$

$$\forall h,i,k \in V \setminus \{s,t\} : \; k > i \; \wedge \; (h,k) \in E \tag{6}$$

$$\kappa \geq \sum_{j\in V\setminus\{s,t\} \wedge \; j \geq i} y_{ij} \qquad \forall i \in V \setminus \{s,t\} \tag{7}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{8}$$

$$y_{ik} \in \{0,1\} \qquad \forall i,k \in V \setminus \{s,t\} : \quad k \geq i \quad . \tag{9}$$

In CDP-SEC we desire to minimize the size of the largest connected component, κ, by removing a path (CDP) from the source s to the destination t of the network. The decision variables x_{ij} equal 1 if edge $(i,j) \in E$ belongs to the CDP and 0 otherwise. Decision variables y_{ik} equal 1 if both vertices $i \in V$ and $k \in V$ are contained in the same connected component; y_{ii} equals 1 if vertex $i \in V$ is not contained in the constructed CDP.

The balance constraints in (1) verify that every vertex in the CDP, with the exception of s and t, has an associated edge coming into it and an associated edge coming out. Constraints (2) and (3) ensure that the CDP contains exactly one edge that emanates from the source vertex and exactly one edge that leads to the destination vertex. The constraints described by (4) eliminate cycles or sub-tours in the CDP. We avoid the complexity associated with these sub-tour elimination constraints via a separation mechanism, inspired by algorithms for the TSP. We reduce the number of sub-tour elimination callbacks via the addition of the following constraints in CDP-SEC

$$x_{ij} + x_{ji} \leq 1 \qquad \forall (i,j) \in E \; \wedge \; j > i \quad . \tag{10}$$

These constraints ensure that, at most, one of the edges (i,j) or (j,i) is included in the CDP. Note that constraints (10) are a subset of those in (4). Together with the binary restrictions (8), constraints (1)-(4) ensure that variables x_{ij} define an $s-t$ path.

The connected components in the network are identified by constraints (5), (6), and (9). The constraints in (5) ensure that y_{ii} equals 0 if there is an edge in the CDP that includes vertex i, and equals 1 otherwise. The constraints in (6) force vertex $k \in V$ to be in the same connected component as vertex $i \in V$, if there exists a vertex $h \in V$, in the connected component,

which is connected via an edge to vertex k, and vertex k is not contained in the computed CDP. Note that in (6), node i can coincide with node h if there exists a node $k \in V$ with $k > i$ and the edge $(i, k) \in E$ exists.

Last, (7) calculate the largest connected component size and assign κ this value. The integrality condition in (9) together with the constraints in (7) ensure κ can only assume non-negative integer values.

We consider three modifications to CDP-SEC:

Relaxation of (5)−

$$y_{ii} \geq 1 - \sum_{l:(l,i)\in E} x_{li} \qquad \forall i \in V \setminus \{s, t\}; \tag{11}$$

Strengthening of (10)−

$$\sum_{j:(i,j)\in E} x_{ij} \leq 1 \qquad \forall i \in V \setminus \{s, t\}; \tag{12}$$

Adding the Connected Component Cut−

$$y_{ik} \leq 1 - \sum_{l:(l,k)\in E} x_{lk} \qquad \forall i, k \in V \setminus \{s, t\} \quad . \tag{13}$$

By relaxing the constraints in (5) we enlarge the (integer) solution space and do not impact the optimal objective function value. This relaxation potentially reduces the computational time required to find a feasible solution. By strengthening the constraints in (10), as shown in (12), we reduce the number of required sub-tour elimination separation calls. The cuts we present in (13), reduce the size of the feasible region, with the hope of obtaining better lower bounds.

5. Basic Branch-and-Price Algorithm

Next, we introduce our tailored solution methodology. Our methodology efficiently solves the CDPD problem. We begin by presenting a path-based modification to the monolith formulation presented above. This modification allows us to take advantage of path-based solution methodologies and algorithms.

5.1. Path-based Formulation (PF)

It is natural to formulate the CDPD problem using paths. Let set $\mathcal{P}$ be the collection of all $s-t$ paths in $\mathcal{G}$ and let binary decision variables x_p, with

$p \in \mathcal{P}$, denote the corresponding CDP. Specifically, binary variable x_p equals 1 if path p defines a CDP in the network and 0 otherwise. Additionally, suppose the indicator function $\mathcal{I}_{ip}$ equals 1 if vertex i is contained in path p and 0 otherwise. This enables us to reformulate CDP-SEC as a path (column) problem.

PF:

$$
\begin{aligned}
z^* = \min\ & \kappa \\
\text{s.t.}\ & \sum_{p \in \mathcal{P}} x_p = 1 && (14)\\
& y_{ii} = 1 - \sum_{p \in \mathcal{P}} \mathcal{I}_{ip} x_p \qquad \forall i \in V \setminus \{s,t\} && (15)\\
& y_{ik} \geq y_{ih} - \sum_{p \in \mathcal{P}} \mathcal{I}_{kp} x_p \\
& \qquad\qquad \forall h,i,k \in V \setminus \{s,t\} :\ k > i\ \wedge\ (h,k) \in E && (16)\\
& \kappa \geq \sum_{j \in V \setminus \{s,t\}\ \wedge\ j \geq i} y_{ij} \qquad \forall i \in V \setminus \{s,t\} && (17)\\
& x_p \in \{0,1\} \qquad \forall p \in \mathcal{P} \\
& y_{ik} \in \{0,1\} \qquad \forall i,k \in V \setminus \{s,t\} :\ k \geq i \quad .
\end{aligned}
$$

The constraints given in (14) ensure that exactly one path is selected as the CDP. Constraints (15)-(16) are the path equivalents to (5) and (6) in CDP-SEC. The constraints given in (17) are identical to the constraints in (7). Last, the group of constraints (1)-(4), that define a path in CDP-SEC, are implicitly included in PF via the definition of a path p and the indicator function $\mathcal{I}_{ip}$. The decision variables y_{ik} are defined exactly as they were previously in CDP-SEC.

When including all paths $p \in \mathcal{P}$ formulations PF and CDP-SEC are equivalent: they both contain the same set of CDPs leading to the same optimal objective function value. Clearly, the cuts given in (11) and (13) can be applied to PF.

5.2. Master and Pricing Problem

Since the number of paths $p \in \mathcal{P}$ may grow exponentially with the number of vertices in the network, we propose a tailored B&P algorithm. We first relax variables x_p and y_{ij} by allowing them to be continuous variables between

0 and 1, and consider only a subset $\mathcal{P}'$ of $\mathcal{P}$ (*i.e.*, $\mathcal{P}' \subseteq \mathcal{P}$ with $\mathcal{P}' \neq \emptyset$). This yields the Restricted and Relaxed Master Problem (RRMP).

RRMP:

$$
\begin{aligned}
z^*_{\text{RRMP}} = \min\ & \kappa \\
\text{s.t.}\ & \sum_{p \in \mathcal{P}'} x_p = 1 && (\pi^1) \quad (18) \\
& y_{ii} + \sum_{p \in \mathcal{P}'} \mathcal{I}_{ip} x_p = 1 \qquad \forall i \in V \setminus \{s,t\} && (\pi^2_i) \quad (19) \\
& y_{ik} - y_{ih} + \sum_{p \in \mathcal{P}'} \mathcal{I}_{kp} x_p \geq 0 \\
& \qquad \forall h,i,k \in V \setminus \{s,t\} : \ \ k > i \ \ \wedge \ \ (h,k) \in E && (\pi^3_{hik}) \quad (20) \\
& \kappa \geq \sum_{j \in V \setminus \{s,t\} \,\wedge\, j \geq i} y_{ij} \qquad \forall i \in V \setminus \{s,t\} \\
& x_p \in [0,1] \qquad \forall p \in \mathcal{P}' \\
& y_{ik} \in [0,1] \qquad \forall i,k \in V \setminus \{s,t\} : \ \ k \geq i \quad .
\end{aligned}
$$

RRMP is a Linear Programming (LP) problem with dual variables π^1, π^2_i and π^3_{hik}, corresponding to constraints (18), (19), and (20). In RRMP, if $\mathcal{P}' = \mathcal{P}$, then we simply call the above the Relaxed Master Problem (RMP). For RMP, $z^*_{\text{RRMP}} \leq z^*$.

In order for a basic feasible solution of RRMP to be optimal over all possible paths $p \in \mathcal{P} \setminus \mathcal{P}'$, each non-basic variable (representing a path) must have non-negative reduced cost, meaning

$$
\pi^1 + \sum_{i \in V \setminus \{s,t\}} \mathcal{I}_{ip} \pi^2_i + \sum_{i \in V \setminus \{s,t\}} \sum_{\substack{(h,k) \in E:\ k > i \\ h,k \notin \{s,t\}}} \mathcal{I}_{ip} \pi^3_{hik} \leq 0 \quad .
$$

We restate the non-negative reduced cost in terms of the primal variables of formulation CDP-SEC as

$$
\pi^1 + \sum_{i \in V \setminus \{s,t\}} \left(\pi^2_i \sum_{l:\ (l,i) \in E} x_{li} \right) + \sum_{i \in V \setminus \{s,t\}} \sum_{\substack{(h,k) \in E:\ k > i \\ h,k \notin \{s,t\}}} \left(\pi^3_{hik} x_{hi} \right) \leq 0 \quad ,
$$

where indicator function $\mathcal{I}_{ip}$ is replaced with binary decision variable x_{ij}. x_{ij} equals 1 if edge (i,j) is included in path p and 0 otherwise. This leads us to

the so-called Pricing Problem (PP).

$$\text{PP: } z^*_{\text{PP}} = \max \left(\pi^1 + \sum_{i \in V \setminus \{s,t\}} \left(\pi^2_i \sum_{l:\,(l,i) \in E} x_{li} \right) + \sum_{i \in V \setminus \{s,t\}} \sum_{\substack{(h,k) \in E:\ k > i \\ h,k \notin \{s,t\}}} \left(\pi^3_{hik} x_{hi} \right) \right)$$
$$\text{s.t. } (1), (2), (3), (4), (8) \quad .$$

If $z^*_{\text{PP}} > 0$, then we have found a new path p to be included in set $\mathcal{P}'$ of RRMP. The new path is found through the optimal solution values of the decision variables x_{ij}, $(i,j) \in E$, defining indicator function $\mathcal{I}_{ip}$. PP belongs to the class of weighted and/or longest path problems, because the cost coefficients can take positive values (duals π^3_{hik} are non-negative). Thus, PP is NP-hard.

If $z^*_{\text{PP}} \leq 0$, RRMP has been solved to optimality, any optimal solution of RRMP is an optimal solution to RMP, and $z^*_{\text{RRMP}} \leq z^*$. In this case, the optimal objective function value of RRMP defines a lower bound $\underline{z}$ for the optimal solution of the CDPD problem. The largest connected component, resulting from removing any $s-t$ path from the network, provides us with an upper bound $\bar{z}$ which can be calculated in polynomial time. Thus, any path generated by the pricing problem PP provides us an upper bound $\bar{z}$.

5.3. Branching

If the best lower bound $\underline{z}$ and the best upper bound $\bar{z}$ differ, then at least one variable in the optimal solution of RRMP has a fractional value and we need to implement a branching procedure. If we implement our B&P algorithm and solve RMP to optimality, but at least one variable is fractional (*i.e.*, does not have a binary value), then we use the branching procedure described below. If all the variables are binary, then the solution to RMP is the optimal solution to the original problem and we are done.

We consider three cases:

Case I: Some of the x_p variables are fractional. If there are fractional values in the x_p variables, then there are at least two.

Case II: All x_p variables are binary and at least one y_{ij} is fractional. If this is true, then $\underline{z} = \bar{z}$ and the CDPD problem has been solved to optimality.

Case III: κ is fractional.

Case I: Let there be $\overline{P}$ paths in RRMP, *i.e.*, $\overline{P} = |\mathcal{P}'|$. We generate $\overline{P}+1$ subproblems as follows:

Subproblem i $(p \in \mathcal{P}')$*:* $x_p = 1$; select path p as candidate for CDP

Subproblem $\overline{P}+1$*:* $x_p = 0 \ \ (\forall p \in \mathcal{P}')$; all previously generated paths are eliminated from RRMP and PP.

Subproblems 1 to $\overline{P}$ are special in that no columns need to be added, because of constraint (18). Thus, the optimal κ is given by the size of the largest connected component associated with the appropriate path. Given a CDP p and the disconnected induced subgraph $\mathcal{G}^p = (V^p, E^p)$, obtained via deletion of p, the size of the largest connected component is computed by executing a depth first search for each vertex $i \in V \setminus V(p)$.

The size of the largest connected component resulting from each of the $\overline{P}$ paths yields an upper bound $\bar{z}$ to the CDPD problem. Additionally, each subproblem can be fathomed due to integrality and removed from the branch-and-bound tree.

Subproblem $\overline{P}+1$ has to be solved by adding additional columns. In this case, the column generating subproblems (*i.e.*, the pricing problems PP) have to be changed so that the $\overline{P}$ previously generated paths can no longer be generated. This is achieved by adding the constraints

$$\sum_{(i,j)\in E(p)} x_{ij} \leq |E(p)| - 1 \qquad \forall p \in \mathcal{P}'$$

to the pricing problems PP. Once the reduced cost of the pricing problems are non-positive the resulting z^*_{RRMP} yields a lower bound $\underline{z}$. The branching then starts again for subproblem $\overline{P}+1$. Thus, the branch-and-bound tree always contains at most one vertex (in addition to the root vertex).

Case II: RRMP reads as follows:

$$\begin{aligned}
z^*_{\text{RRMP}} = \min\ & \kappa \\
\text{s.t.}\ & \\
& y_{ii} + \mathcal{I}_{ip} = 1 \qquad \forall i \in V \setminus \{s,t\} && (22)\\
& y_{ik} - y_{ih} + \mathcal{I}_{kp} \geq 0 \\
& \qquad\qquad \forall h,i,k \in V \setminus \{s,t\} :\ k > i \ \wedge\ (h,k) \in E && (23)\\
& \kappa \geq \sum_{j \in V\setminus\{s,t\}:\ j \geq i} y_{ij} \qquad \forall i \in V \setminus \{s,t\}
\end{aligned}$$

$$y_{ik} \in [0,1] \qquad \forall i,k \in V \setminus \{s,t\} : \quad k \geq i \quad ,$$

since $x_p = 1$ for some path $p \in \mathcal{P}'$. Let i be the smallest index (vertex) contained in the largest connected component. Then $y_{ii} = 1$ due to constraint (22). Constraints (23) imply that for all vertices j in the largest connected component, $y_{ij} = 1$.

This proves that κ^* is never fractional. Even if some of the y_{ij} variables are fractional, if at least one $p \in \mathcal{P}'$, $x_p = 1$, then κ^* is the optimal objective function value.

Case III: If κ is fractional, then y_{ij} has to have fractional value(s). Thus, case I or II applies.

The discussion above suggests, as our branching strategy, we remove all generated paths from RRMP and forbid their generation in the pricing problems PP. Since removing all paths leaves an "empty" RRMP, *i.e.*, $\mathcal{P}' = \emptyset$, in order to derive a meaningful solution from RRMP, we (randomly) select one of the generated paths and include it in RRMP as the only path.

5.4. Generic Branch-and-Price

A generic B&P algorithm for the CDPD problem is summarized below:

---

1. **Initialize:** $\underline{z} = 0$, $\bar{z} = \infty$, initialize set of paths $\mathcal{P}'$ (*i.e.*, use depth-first-search or breath-first-search)

2. **Solve RRMP:** Obtain dual information (value of dual variables)

3. **Solve PP:** If $z^*_{\text{PP}} > 0$, add new path to $\mathcal{P}'$ and update upper bound $\bar{z}$, go to step 2; else update lower bound $\underline{z} \leftarrow \max\{\underline{z}, z^*_{\text{RRMP}}\}$

4. **Check convergence:** If $\underline{z} = \bar{z}$, STOP (optimal solution found)

5. **Branch:** Remove all previously generated paths from RRMP and PP, with the exception of one path in RRMP, to ensure feasibility. Go to step 2.

---

6. Branch-and-Price Algorithm Intricacies

We further tailor the basic B&P algorithm discussed in Section 5 to the CDPD problem through the generation of an initial pool of paths (Section 6.1); the detection of a Hamiltonian Path, if it exists (Section 6.2); the derivation of lower bounds from computed paths (Section 6.3); the incorporation of cuts to the path generating subproblems (Section 6.4); and through acceleration, utilizing dual stabilization (Section 6.5). We close this section by summarizing the obtained algorithm (Section 6.6).

6.1. Obtaining Initial Columns

To generate an initial column for RRMP, in the B&P algorithm, we first create a pool of feasible solutions and we then choose a CDP, among them, which yields the smallest maximum connected component. The pool is filled using two strategies. The first strategy is based on the connectivity concept and generates paths using a Depth-First-Search (DFS) procedure. For each vertex $i \in V \setminus \{s,t\}$ we perform two DFSs: one from the source s to the vertex i and one from i to the destination t. Carefully concatenating the two paths leads to a simple $s-t$ path. The second strategy is motivated by the idea of generating extreme points of a polyhedron. Specifically, we seek to compute $s-t$ paths not already contained in the solution pool. To compute these paths, we consider the following problem:

COLr:

$$
\begin{aligned}
z^*_{\text{COLr}} = \min \sum_{(i,j)\in E} r_{ij} x_{ij} & \\
\text{s.t.} \sum_{(i,j)\in E} x_{ij} \geq |p^*| & \qquad (24)\\
\sum_{(i,j)\in E(p)} x_{ij} \leq |E(p)| - 1 \qquad \forall p \in \mathcal{P} & \qquad (25)\\
(1)-(3),(4),(8),(12) \quad . &
\end{aligned}
$$

In COLr, we compute a path which exceeds or matches the size of the longest path p^* in the path pool $\mathcal{P}$ (via constraints (24)), and is different than any of the paths already contained in the path pool $\mathcal{P}$ (via constraints (25)). Parameter r_{ij} is a random vector that is uniformly distributed over $[0, |V|]$. r_{ij} assigns different weights to edge (i,j) in the graph. We use this randomized objective function to generate different solutions, which makes the path pool more diverse (a strategy often used in meta-heuristics).

6.2. Existence of a Hamiltonian Path

If a Hamiltonian Path in $\mathcal{G}$ exists, then the associated connected component size is 0 and we can terminate the algorithm; the Hamiltonian Path defines a CDP. If no Hamiltonian Path exists, then the size of the minimum connected component is greater than or equal to 1.

We check for the existence of a Hamiltonian Path by solving the following feasibility problem:

HP:

$$\begin{aligned} z^*_{\text{HP}} = \max\ & 0 \\ \text{s.t.}\ & \sum_{(i,j)\in E} x_{ij} \geq |V| - 1 \\ & (1) - (4), (8), (12) \quad . \end{aligned}$$

HP solves an NP-hard problem. Thus if we know a priori that the instances solved do not contain a Hamiltonian Path, we skip this step.

6.3. Lower Bounds from Paths

In an effort to improve the lower bound, $\underline{z}$, we analyze substructures within the network. These substructures can reveal the existence of connected components, whose size cannot be reduced via the removal of any path p in the network.

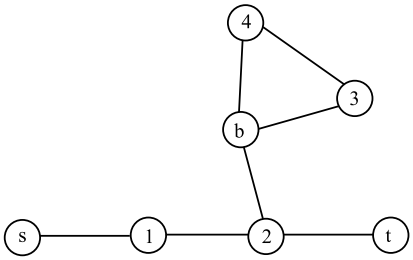
We begin with the definition of a special edge for CDPs:

Definition 4 (*Unobtainable Edge*).
We call an edge $(i, j) \in E$ *unobtainable*, for any CDP, if no $s - t$ path exists which contains edge (i, j).

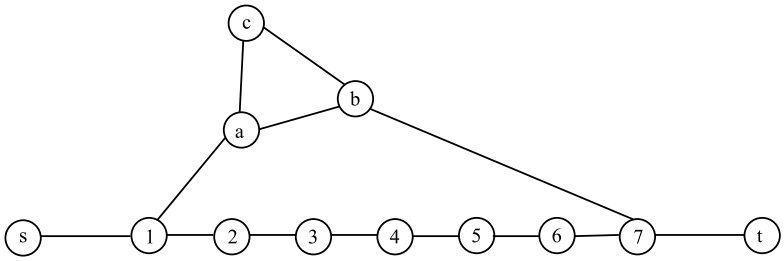
We check every path we generate throughout the B&P algorithm for the existence of unobtainable edge(s) and update the lower bound as described below. If the removal of path p yields a connected component C, such that there is exactly one edge (b, j) linking the component with the path (*i.e.*, $b \in C$ and $j \in p$), then edge $(b, j) \in E$ is an unobtainable edge and the connected component size $|C|$ yields a lower bound on z^*. Similarly, if there are L edges $e_l \in E$, $l = 1, \ldots, L$, linking the same vertex $b \in C$ to the path p, then all edges e_l are unobtainable edges and $|C|$ yields a lower bound on z^*. For example, in Figure 3(a), vertices b, 3, and 4 cannot be contained in

any $s - t$ path, thus edge $(2, b)$ is unobtainable and we have a lower bound of 3 for z^*.

Another substructure, we use to improve the lower bound, is a *chain* contained in an $s-t$ path. Suppose we are given a path p that contains a chain and suppose that vertices i and j are contained in this chain. Also suppose that vertices i and j lead to a sub-chain of length k. If, after the removal of path p, there exists a connected component C which contains exactly two edges linking the path with C and if these edges connect the component with vertices i and j, then $\min\{|C|, k\}$ defines a lower bound on z^*. As an example, consider path $p = \{(s, 1), (1, 2), (3, 4), (4, 5), (5, 6), (6, 7), (7, t)\}$ in Figure 3(b) with $i = 1$ and $j = 7$. The connected component $C = \{a, b, c\}$ yields a lower bound of 3 for any CDP.



(a) Edge $(2, b)$ is unobtainable



(b) Path containing a chain

Figure 3: *Deriving lower bounds from substructures via paths*

6.4. Cuts

In this section, we describe several families of valid inequalities, based on the topology of the network and on the degree of the network's vertices.

These inequalities provide additional information for the pricing problems. More specifically, these inequalities provide indirect information on the size of the connected component that results from removing a CDP.

The first family of inequalities we employ, are what we term the degree-resizing-upper constraints. These cuts aim to eliminate those feasible solutions from the pricing problem which resulted in a maximum connected component size greater than or equal to the current upper bound $\bar{z}$. The second family, the so-called degree-resizing-lower constraints, aim to improve the current lower bound $\underline{z}$. If these constraints lead to an infeasible subproblem, then $\underline{z}$ is not a valid lower bound and we update the lower bound to $\underline{z} + 1$.

We define $d[i]$ as the degree of vertex $i \in V$ and $\mathrm{Adj}(i)$ as the collection of all vertices adjacent to vertex $i \in V$, *i.e.*, $\mathrm{Adj}(i) = \{j \in V \mid (i,j) \in E\}$.

6.4.1. Degree-resizing-upper Constraints

These inequalities provide information to the pricing problem regarding the resulting connected component size. With these constraints, we generate paths with smaller connected component size than the smallest ones found thus far. Consider the following degree-resizing-upper inequalities for $\kappa > z^*$:

$$\mathrm{d}[i] \sum_{j:(i,j)\in E} x_{ij} + \sum_{(h,r)\in E:\ h\in \mathrm{Adj}(i)} x_{hr} \geq \mathrm{d}[i] - \kappa + 2 \qquad \forall i \in V:\ \mathrm{d}[i] \geq 2. \tag{26}$$

We show that inequalities (26) are valid inequalities for CDP-SEC by considering the case where $i \in V(p)$, and the case where $i \notin V(p)$, for a computed CDP p. If $i \in V(p)$, then from constraints (1), (2), and (3) we have

$$\sum_{j:(i,j)\in E} x_{ij} = 1 \quad \text{and} \quad \sum_{(h,r)\in E:\ h\in Adj(i)} x_{hr} \geq 2.$$

Because $\mathrm{d}[i] + 2 \geq \mathrm{d}[i] + 2 - \kappa$, for $\kappa \geq 0$, inequality (26) holds.

If $i \notin V(p)$, then constraints (1), (2), and (3) yield the following equality:

$$\sum_{j:(i,j)\in E} x_{ij} = 0,$$

and we must show that

$$\sum_{(h,r):\ h\in \mathrm{Adj}(i)} x_{hr} \geq \mathrm{d}[i] - \kappa + 2.$$

In this case, each vertex i forms a connected component of size greater than or equal to $\mathrm{d}[i]+1$, where the connected component is comprised of itself and its adjacent vertices, only. If vertex i, or one of its adjacent vertices, is in the CDP, then the sizes of the connected components that contain i are reduced by at least one unit. Thus, given a CDP with maximum component size z^*, the CDP must contain at least $\mathrm{d}[i]+1-z^*$ vertices of this connected component, otherwise a connected component with size greater than z^* must be present in the graph. The deletion of any of the vertices $j \in \mathrm{Adj}(i)$ reduces the size of this connected component by exactly 1 and this leads us to

Corollary 2. *The degree-resizing-upper inequalities* (26) *are valid inequalities for* CDP-SEC *for any* $\kappa > z^*$.

Corollary 2 allows us to add constraints (26) to the pricing problem PP to obtain

$\mathrm{PP}_{\mathrm{DR}}$:

$$z^*_{\mathrm{PP}_{\mathrm{DR}}} = \max \left(\pi^1 + \sum_{i \in V \setminus \{s,t\}} \left(\pi^2_i \sum_{l:\,(l,i)\in E} x_{li} \right) + \sum_{i \in V \setminus \{s,t\}} \sum_{\substack{(h,k)\in E:\, k>i \\ h,k \notin \{s,t\}}} \left(\pi^3_{hik} x_{hi} \right) \right)$$

$$\text{s.t. } (1), (2), (3), (4), (8), (26) \quad .$$

6.4.2. Degree-resizing-lower constraints

For $\kappa \geq 1$, if there exists a CDP with maximum connected component size of at most κ, then the following degree-resizing-lower inequalities hold:

$$\mathrm{d}[i] \sum_{j:(i,j)\in E} x_{ij} + \sum_{(h,r):\, h \in \mathrm{Adj}(i)} x_{hr} \geq \mathrm{d}[i] - \kappa + 1 \qquad \forall i \in V:\ \mathrm{d}[i] \geq \kappa, \tag{27}$$

$$\mathrm{d}[i] \sum_{j:(i,j)\in E} x_{ij} + \sum_{(h,r):\, h \in \mathrm{Adj}(i)} x_{hr} \geq 1 \qquad \forall i \in V:\ 2 \leq d[i] = \kappa - 1 \tag{28}$$

Following the same line of reasoning as was done for constraints (26), we obtain

Corollary 3. *The degree-resizing-lower inequalities* (27) *and* (28) *are valid inequalities for* CDP-SEC *for* $\kappa = z^*$.

Cuts (27) and (28) can then be applied to decide whether κ equals z^* by solving

DRL:

$$z^*_{\text{DRL}} = \max \sum_{(i,j)\in E} x_{ij}$$
$$\text{s.t. } (1), (2), (3), (4), (8), (12), (27), (28) \quad .$$

In practice we apply the degree-resizing-lower inequalities (27) and (28) in cases where $\kappa = \underline{z}$. If (27) and/or (28) are violated, then we add $\kappa > \underline{z}$, as a cut, to RRMP.

6.5. Acceleration via Dual Stabilization

The generic column generation approach shows very slow convergence for the tested instances due to degeneracy (multiple dual solutions are associated with each primal solution), thus we apply a stabilization method similar to the methods that Rousseau et al. [23] and Lübbecke and Desrosiers [24] present. The interior point stabilization is most promising among the tested approaches, especially compared to box stabilization. The interior point stabilization technique generates a dual solution that is interior to the convex hull of the optimal dual solutions, in place of using extreme points. To obtain a point in this polyhedron, we consider a randomized objective function yielding the stabilized pricing problem (SPP^u).

SPP^u:

$$z^*_{\text{SPP}^u} = \max \left(\pi^1 + \sum_{i\in V\setminus\{s,t\}} \left(\pi^2_i u^2_i \sum_{l:\ (l,i)\in E} x_{li} \right) + \sum_{i\in V\setminus\{s,t\}} \sum_{\substack{(h,k)\in E:\ k>i \\ h,k\notin\{s,t\}}} (\pi^3_{hik} u^3_{hik} x_{hi}) \right)$$
$$\text{s.t. } (1)-(4), (8), (12) \quad ,$$

where every element of the vectors u^2 and u^3 are uniformly distributed between 0 and 1. Generating values for u^2 and u^3 results in multiple objective functions and several extreme points. A quick way to obtain an interior point is to take the average of all the extreme points.

6.6. Branch-and-Price: Putting Everything Together

The tailored B&P algorithm for the CDPD problem is summarized below:

1. **Initialize:** $\underline{z} = 0$, $\overline{z} = |V| - 2$, generate a pool of path solutions (see Section 6.1)

2. **Solve RRMP:** Obtain dual information (value of dual variables)

3. **Solve $\mathbf{PP}_{\mathrm{DR}}$:** If $z^*_{\mathrm{PP}} > 0$, add new path to $\mathcal{P}'$ and go to step 4; else update lower bound $\underline{z} \leftarrow \max\{\underline{z}, z^*_{\mathrm{RRMP}}\}$ and go to step 7

4. **Compute $\underline{z}$ and $\overline{z}$:** Update the upper bound by computing the size of the maximum connected component obtained by removing the path found in step 3 and update, if possible, the lower bound $\underline{z}$ (see Section 6.3)

5. **Improvements:** Every three iterations without improvement of both the lower and the upper bound, solve the model DRL that uses the degree-resizing-lower constraints (see Section 6.4.2) or apply the interior point stabilization method (see Section 6.5)

6. **Remove columns at iteration limit:** Every fifty iterations without improvement of the lower bound $\underline{z}$, remove all generated paths, except one randomly chosen one

7. **Check convergence:** If $\underline{z} = \overline{z}$, STOP (optimal solution found)

8. **Branch:** Remove all previously generated paths from RRMP and PP, with the exception of one path in RRMP, to ensure feasibility. Go to step 2.

---

7. Computational Results

We use C++ to evaluate our B&P algorithms. All mathematical optimization problems are solved using CPLEX 12.2 and the computations are performed on an Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz with 14 GBs of RAM. We set the time limit to 3,600 seconds and use CPLEX settings `MemoryEmphasis=1` and `NodeFileInd=3`.

We perform our computational tests on random networks which are generated according to two parameters: the number of vertices $n = |V|$ and

the density of the network d. The number of edges in the network is then computed as $m := |E| = d \cdot \frac{n(n-1)}{2}$. The instances are carefully generated, to ensure the network is connected and to avoid the creation of a Hamiltonian Path. For each group of networks, defined by the number of vertices and the density, we generate 50 problem instances.

Tables 2 - 3 report the performance of the monolith model: the mathematical programming formulation CDP-SEC with the additional constraints given in (12). We found that this is the most computationally efficient configuration among the cuts discussed in Section 4. The first three columns, which summarize the characteristics of the instances solved, are: the number of vertices n, the number of edges m, and the resulting density d. The fourth column states the number of instances which can be solved by CPLEX within the time limit, while the fifth column reports the number of instances where CPLEX was not able to compute a feasible solution within the time limit. Column six lists the number of cases in which a feasible solution is found, but optimality is not proven and the average optimality gap for these cases. The column labeled $\overline{\mathbf{T}}^*$, reports the average running time of CPLEX, considering only those instances which are solved to optimality within the time limit. The average number of callbacks required to eliminate sub-tours are reported in the last column.

We make the following observations from the results reported in Tables 2 - 3:

- CPLEX is able to solve 3,013 out of the 6,500 instances, within the time limit
- For most of the problems which could not be solved by CPLEX, no feasible solution was generated. More specifically, out of the 3,487 instances which could not be solved, CPLEX was able to find a feasible solution for 53 instances
- Of the instances not solved by CPLEX, 3,440 are not solved to optimality due to lack of memory and 47 are not solved to optimality in the time limit
- CPLEX tends to have difficulties solving the instances on networks with density above 20%
- The larger the density (above 10%), the more challenging the instances tend to be for CPLEX

- The average number of sub-tour elimination calls are small, compared to the number of existing sub-tours in the network

Table 4 contrasts the performance of the generic B&P algorithm to the B&P algorithm with enhancements. The columns labeled "gap" report the average relative difference between the best known upper bound ($\overline{z}$) and lower bound ($\underline{z}$), obtained for the instances which cannot be solved to optimality within the time limit. The gap is calculated as $\frac{\overline{z}-\underline{z}}{\underline{z}}$. The columns labeled "# PPs" report the number of pricing problems solved. The computational results demonstrate the superiority of the developed B&P algorithm of Section 6.6 over the generic B&P algorithm.

Tables 5 - 6 summarize the performance of the B&P algorithm with enhancements, in comparison to the CDP-SEC formulation solved by CPLEX. The last five columns compare CDP-SEC with the B&P algorithm. The column "solved" (for CDP-SEC vs. B&P) first lists the number of instances solved to optimality by CPLEX and then lists, out of those instances, how many the B&P algorithm is able to solve. Out of the instances which are solved by both CPLEX and the B&P algorithm, the average running times are reported ($\overline{\mathbf{T}}^*$), and the number of instances CPLEX was able to solve faster (at least 10 times and at least 100 times) compared to the B&P algorithm and vise versa are documented.

We draw the following conclusions from Tables 5 - 6:

- The B&P algorithm is able to solve 5,024 out of the 6,500 instances, within the time limit
- The B&P algorithm solves 2,918 of the 3,013 instances that CPLEX is able to solve
- CPLEX solves 2,918 of the 5,024 instances the B&P algorithm is able to solve
- For the 2,918 instances which are solved by both the B&P algorithm and CPLEX:
  - the B&P algorithm is faster than CPLEX in 2,758 instances, compared to 160 instances where CPLEX is faster than the B&P algorithm

Instance				No feasible	Feasible solution -		Average # of
n	m	d	solved	solution found	not optimal/gap	$\overline{\mathbf{T}}^*$	callbacks used
40	39	0.05	50	0	– / –	0.03	1.00
40	46	0.06	50	0	– / –	0.31	6.76
40	54	0.07	50	0	– / –	3.14	110.86
40	62	0.08	50	0	– / –	14.35	488.96
40	70	0.09	49	0	1 / 8.00	47.08	986.82
40	78	0.10	47	0	3 / 2.34	32.03	872.57
40	156	0.20	46	3	1 / 0.01	6.08	47.74
40	234	0.30	47	3	0 / –	11.44	41.64
40	312	0.40	45	5	0 / –	12.99	29.62
40	390	0.50	44	5	1 / 1.00	17.19	21.43
40	468	0.60	48	2	0 / –	35.27	68.69
40	546	0.70	47	3	0 / –	45.45	74.66
40	624	0.80	44	6	0 / –	50.59	60.50
40	702	0.90	47	3	0 / –	72.49	76.89
			664 of 700				
60	70	0.04	50	0	– / –	1.95	10.56
60	88	0.05	48	1	1 / 5.00	70.00	581.46
60	106	0.06	46	1	3 / 3.34	228.08	1353.98
60	123	0.07	44	1	5 / 3.00	91.51	463.70
60	141	0.08	38	7	5 / 2.20	19.30	86.24
60	159	0.09	37	6	7 / ∞	206.88	1017.27
60	177	0.10	45	3	2 / 0.50	99.16	420.20
60	354	0.20	45	5	0 / –	62.33	100.96
60	531	0.30	44	6	0 / –	89.31	73.95
60	708	0.40	44	6	0 / –	125.35	61.34
60	885	0.50	39	10	1 / ∞	259.39	184.44
60	1062	0.60	43	7	0 / –	244.61	57.65
60	1239	0.70	40	9	1 / ∞	400.81	141.58
60	1416	0.80	37	12	1 / 7.00	392.79	53.65
60	1593	0.90	42	8	0 / –	458.81	40.07
			642 of 750				
80	94	0.03	50	0	– / –	3.74	6.66
80	126	0.04	44	2	4 / 1.89	138.56	314.89
80	158	0.05	42	3	5 / 1.40	192.20	417.40
80	189	0.06	39	5	6 / 3.34	259.32	500.41
80	221	0.07	39	8	3 / 0.77	132.96	226.95
80	252	0.08	42	5	3 / 2.00	222.01	337.52
80	284	0.09	46	4	0 / –	123.50	138.50
80	316	0.10	39	11	0 / –	174.29	189.46
80	632	0.20	40	10	0 / –	339.43	152.35
80	948	0.30	39	11	0 / –	544.40	139.08
80	1264	0.40	34	16	0 / –	638.02	80.15
80	1580	0.50	7	43	0 / –	1210.23	141.71
80	1896	0.60	0	50	0 / –	–	–
80	2212	0.70	0	50	0 / –	–	–
80	2528	0.80	0	50	0 / –	–	–
80	2844	0.90	0	50	0 / –	–	–
			461 of 800				
100	99	0.02	50	0	– / –	0.51	1.00
100	148	0.03	41	9	0 / –	181.79	191.27
100	198	0.04	42	8	0 / –	36.67	24.12
100	247	0.05	41	9	0 / –	210.08	138.24
100	297	0.06	36	14	0 / –	519.13	325.92
100	346	0.07	39	11	0 / –	380.74	193.38
100	396	0.08	39	11	0 / –	289.96	117.36
100	445	0.09	38	12	0 / –	511.62	193.45
100	495	0.10	41	9	0 / –	619.77	208.63
100	990	0.20	37	13	0 / –	950.97	101.41
100	1485	0.30	0	50	0 / –	–	–
100	1980	0.40	0	50	0 / –	–	–
100	2475	0.50	1	49	0 / –	811.30	18.00
100	2970	0.60	0	50	0 / –	–	–
100	3465	0.70	0	50	0 / –	–	–
100	3960	0.80	0	50	0 / –	–	–
100	4455	0.90	0	50	0 / –	–	–
			405 of 850				

Table 2: Computational results for monolith model (CDP-SEC) using Equation (12) for instances with 40 to 100 vertices

Instance				No feasible	Feasible solution -		Average # of
n	m	d	solved	solution found	not optimal/gap	$\overline{\mathbf{T}}^*$	callbacks used
120	142	0.02	50	0	– / –	154.66	73.14
120	214	0.03	37	13	0 / –	387.23	130.27
120	285	0.04	37	13	0 / –	488.17	154.30
120	357	0.05	39	11	0 / –	455.42	95.64
120	428	0.06	41	9	0 / –	899.11	170.63
120	499	0.07	41	9	0 / –	870.55	123.07
120	571	0.08	33	17	0 / –	1016.51	116.79
120	642	0.09	33	17	0 / –	944.25	90.91
120	714	0.10	32	18	0 / –	948.67	77.22
120	1428	0.20	0	50	0 / –	–	–
120	2142	0.30	3	47	0 / –	1493.99	28.67
120	2856	0.40	0	50	0 / –	–	–
120	3570	0.50	0	50	0 / –	–	–
120	4284	0.60	0	50	0 / –	–	–
120	4998	0.70	0	50	0 / –	–	–
120	5712	0.80	0	50	0 / –	–	–
120	6426	0.90	0	50	0 / –	–	–
			346 of 850				
140	194	0.02	39	11	0 / –	276.13	39.85
140	291	0.03	46	4	0 / –	627.93	67.43
140	389	0.04	37	13	0 / –	1258.33	116.81
140	486	0.05	33	17	0 / –	971.76	70.82
140	583	0.06	28	22	0 / –	1441.42	84.93
140	681	0.07	28	22	0 / –	1234.54	54.50
140	778	0.08	6	44	0 / –	1659.86	74.50
140	875	0.09	8	42	0 / –	2344.85	96.63
140	973	0.10	3	47	0 / –	2483.58	90.33
140	1946	0.20	1	49	0 / –	1460.32	46.00
140	2919	0.30	0	50	0 / –	–	–
140	3892	0.40	0	50	0 / –	–	–
140	4865	0.50	0	50	0 / –	–	–
140	5838	0.60	0	50	0 / –	–	–
140	6811	0.70	0	50	0 / –	–	–
140	7784	0.80	0	50	0 / –	–	–
140	8757	0.90	0	50	0 / –	–	–
			229 of 850				
160	254	0.02	44	6	0 / –	344.54	21.66
160	381	0.03	39	11	0 / –	843.34	41.18
160	508	0.04	37	13	0 / –	974.32	36.22
160	636	0.05	22	28	0 / –	1202.58	33.64
160	763	0.06	5	45	0 / –	2068.29	52.60
160	890	0.07	0	50	0 / –	–	–
160	1017	0.08	1	49	0 / –	2773.37	169.00
160	1144	0.09	10	40	0 / –	989.29	16.00
160	1272	0.10	3	47	0 / –	1380.67	23.67
160	2544	0.20	0	50	0 / –	–	–
160	3816	0.30	0	50	0 / –	–	–
160	5088	0.40	0	50	0 / –	–	–
160	6360	0.50	0	50	0 / –	–	–
160	7632	0.60	0	50	0 / –	–	–
160	8904	0.70	0	50	0 / –	–	–
160	10176	0.80	0	50	0 / –	–	–
160	11448	0.90	0	50	0 / –	–	–
			161 of 850				
180	322	0.02	42	8	0 / –	659.65	20.29
180	483	0.03	35	15	0 / –	1431.81	34.63
180	644	0.04	15	35	0 / –	2217.42	54.00
180	805	0.05	0	50	0 / –	–	–
180	966	0.06	3	47	0 / –	1521.01	18.67
180	1127	0.07	9	41	0 / –	1821.82	22.11
180	1288	0.08	1	49	0 / –	2211.07	19.00
180	1449	0.09	0	50	0 / –	–	–
180	1611	0.10	0	50	0 / –	–	–
180	3222	0.20	0	50	0 / –	–	–
180	4833	0.30	0	50	0 / –	–	–
180	6444	0.40	0	50	0 / –	–	–
180	8055	0.50	0	50	0 / –	–	–
180	9666	0.60	0	50	0 / –	–	–
180	11277	0.70	0	50	0 / –	–	–
180	12888	0.80	0	50	0 / –	–	–
180	14499	0.90	0	50	0 / –	–	–
			105 of 850				

Table 3: Computational results for monolith model (CDP-SEC) using Equation (12) for instances with 120 to 180 vertices

Instance			Generic B&P				B&P			
n	m	d	Solved	Gap	# PPs	$\overline{\mathbf{T}}^*$	Solved	Gap	# PPs	$\overline{\mathbf{T}}^*$
40	39	0.05	50	–	0.00	0.00	50	–	0.00	0.00
40	46	0.06	33	0.64	59.03	0.05	50	–	0.32	0.01
40	54	0.07	26	0.61	60.92	0.06	50	–	1.78	0.09
40	62	0.08	19	0.49	94.58	0.07	50	–	8.46	1.04
40	70	0.09	16	0.19	69.44	0.08	49	–	5.63	2.15
40	78	0.10	13	0.25	91.15	273.52	50	–	3.22	0.34
40	156	0.20	4	0.20	387.50	2628.94	48	1.00	7.90	0.76
40	234	0.30	0	0.15	–	–	49	0.80	8.69	1.23
40	312	0.40	2	0.19	720.00	0.00	50	–	5.82	1.03
40	390	0.50	2	0.19	672.00	0.00	50	–	2.40	0.41
40	468	0.60	2	0.04	106.50	0.00	50	–	0.98	0.30
40	546	0.70	0	0.19	–	–	50	–	0.00	0.00
40	624	0.80	4	0.06	155.25	0.00	50	–	0.00	0.00
40	702	0.90	5	0.21	228.20	0.00	50	–	0.00	0.00
			176 of 700				696 of 700			

Table 4: Computational results for generic B&P algorithm (Section 5.4) for instances with 40 vertices compared to results obtained with the final B&P algorithm (Section 6.6)

  - the B&P algorithm is at least 10 times faster than CPLEX in 2,089 instances, compared to 23 instances where CPLEX is at least 10 times faster than the B&P algorithm
  - the B&P algorithm is at least 100 times faster than CPLEX in 1,482 instances, compared to 3 instances where CPLEX is at least 100 times faster than the B&P algorithm

- The B&P algorithm finds feasible solutions for all the instances in which CDPD cannot be solved to optimality. The large gaps found in some of these solutions can be attributed to the inability of the B&P algorithm to sufficiently increase the lower bound within the allotted computational time.

- The B&P algorithm tends to perform better, compared to CPLEX, for both larger problems and for more dense problems

- There are no instances in which our B&P algorithm runs out of memory, which means, if we allow for additional computational time, the B&P algorithm can solve more instances

- The generation of the initial pool (Section 6.1), in combination with the computation of lower bounds, from the paths in the pool (Section 6.3), lead to a very effective preprocessing strategy. Through this strategy, especially sparse instances are often solved in the preprocessing phase, explaining the "close-to-zero" running times.

Our results clearly demonstrate that the B&P algorithm we develop outperforms CPLEX in solving the monolith formulation CDP-SEC in three

aspects: (1) the number of instances solved to optimality; (2) the running times; and (3) the feasible solutions computed.

8. Conclusions

In this paper, we introduce a novel method that optimally destroys a network's connectivity. Existing literature disrupts or destroys a network's connectivity by removing a set number of vertices or edges. Our approach degrades or destroys a network's connectivity by removing a path that minimizes the number of vertices in the largest connected component in the resulting subgraphs. We term such a path a Critical Disruption Path (CDP). We measure a network's connectivity by counting the number of vertices in the largest connected component; the smaller the size of the largest connected component, the more disconnected the network.

After establishing the NP-hardness associated with finding a CDP, we provide a monolith formulation. Following this, we develop a tailored Branch-and-Price algorithm. Extensive computational tests, on 6,500 random networks, show that our B&P algorithm outperforms the monolith formulation by two orders of magnitude.

As future work, one might employ a penalty factor so when there are multiple paths that minimize the largest component size, we select the "best" one. In doing so, one may determine that the best path is the path that leaves the largest number of total components in the remaining subgraph. Alternatively, one may define the best path as the path that minimizes the sum of the scaled component sizes.

9. Acknowledgements

The authors thank Panos Pardalos (University of Florida) for insights provided in the early stages of this research. We also thank Behnam Behdani (University of Florida), Josef Kallrath (BASF), and Timo Lohmann (Colorado School of Mines) for their comments on the paper.

References

[1] P. M. Pardalos, S. Rebennack, Computational challenges with cliques, quasi-cliques and clique partitions in graphs, in: P. Festa (Ed.), Experimental Algorithms, volume 6049/2010 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 13–22.

Instance			B&P			CDP-SEC vs. B&P				
								Faster		
n	m	d	Solved	Gap	$\overline{\mathbf{T}}^*$	Solved	$\overline{\mathbf{T}}^*$	$> 1\times$	$\geq 10\times$	$\geq 100\times$
40	39	0.05	50	–	0.00	50 / 50	0.03 / 0.00	0 / 50	0 / 50	0 / 50
40	46	0.06	50	–	0.01	50 / 50	0.31 / 0.01	0 / 50	0 / 49	0 / 49
40	54	0.07	50	–	0.09	50 / 50	3.14 / 0.09	0 / 50	0 / 49	0 / 48
40	62	0.08	50	–	1.04	50 / 50	14.35 / 1.04	2 / 48	1 / 45	0 / 44
40	70	0.09	49	3.0	2.15	49 / 48	47.89 / 2.19	1 / 47	0 / 42	0 / 40
40	78	0.10	50	–	0.34	47 / 47	32.03 / 0.37	0 / 47	0 / 41	0 / 38
40	156	0.20	48	2.5	0.76	46 / 44	6.00 / 0.80	4 / 40	0 / 24	0 / 21
40	234	0.30	49	4.0	1.23	47 / 46	11.65 / 1.27	1 / 45	0 / 23	0 / 21
40	312	0.40	50	–	1.03	45 / 45	12.99 / 1.15	0 / 45	0 / 30	0 / 30
40	390	0.50	50	–	0.41	44 / 44	17.19 / 0.38	0 / 44	0 / 41	0 / 41
40	468	0.60	50	–	0.30	48 / 48	35.27 / 0.31	0 / 48	0 / 46	0 / 45
40	546	0.70	50	–	0.00	47 / 47	45.45 / 0.00	0 / 47	0 / 47	0 / 47
40	624	0.80	50	–	0.00	44 / 44	50.59 / 0.00	0 / 44	0 / 44	0 / 44
40	702	0.90	50	–	0.00	47 / 47	72.49 / 0.00	0 / 47	0 / 47	0 / 47
			696 of 700			664/660		8/652	1/578	0/565
60	70	0.04	50	–	0.00	50 / 50	1.95 / 0.00	0 / 50	0 / 50	0 / 50
60	88	0.05	50	–	0.67	48 / 48	70.00 / 0.70	0 / 48	0 / 45	0 / 43
60	106	0.06	49	5.0	0.65	46 / 45	73.59 / 0.71	0 / 45	0 / 43	0 / 42
60	123	0.07	50	–	178.19	44 / 44	8.74 / 4.37	2 / 42	0 / 39	0 / 37
60	141	0.08	46	4.8	387.41	38 / 36	19.87 / 196.92	3 / 33	2 / 31	1 / 30
60	159	0.09	46	3.3	501.31	37 / 35	12.79 / 354.70	4 / 31	4 / 27	1 / 27
60	177	0.10	47	4.7	227.09	45 / 43	18.79 / 83.56	2 / 41	1 / 34	1 / 30
60	354	0.20	50	–	4.72	45 / 45	62.33 / 4.75	2 / 43	0 / 29	0 / 25
60	531	0.30	49	1.0	7.47	44 / 43	89.65 / 6.56	1 / 42	0 / 29	0 / 27
60	708	0.40	49	1.0	5.25	44 / 43	127.08 / 4.86	0 / 43	0 / 36	0 / 35
60	885	0.50	49	3.0	140.15	39 / 38	169.78 / 171.31	10 / 28	2 / 10	0 / 7
60	1062	0.60	49	31.0	140.75	43 / 42	240.15 / 155.08	6 / 36	1 / 12	0 / 3
60	1239	0.70	49	12.0	159.51	40 / 40	311.23 / 163.86	4 / 36	1 / 13	0 / 5
60	1416	0.80	49	26.0	159.12	37 / 36	397.65 / 166.21	3 / 33	0 / 8	0 / 2
60	1593	0.90	49	24.0	255.97	42 / 41	457.37 / 269.52	2 / 39	1 / 8	0 / 1
			731 of 750			642/629		39/590	12/414	3/364
80	94	0.03	50	–	0.01	50 / 50	3.74 / 0.01	0 / 50	0 / 46	0 / 41
80	126	0.04	50	–	0.59	44 / 44	138.56 / 0.63	2 / 42	0 / 32	0 / 20
80	158	0.05	50	–	3.46	42 / 42	21.55 / 3.83	6 / 36	0 / 20	0 / 12
80	189	0.06	48	8.0	4.89	39 / 38	77.56 / 3.90	1 / 37	0 / 23	0 / 14
80	221	0.07	48	7.0	26.66	39 / 38	41.94 / 12.41	6 / 32	2 / 20	0 / 9
80	252	0.08	48	12.0	34.82	42 / 40	139.35 / 31.52	7 / 33	2 / 17	0 / 5
80	284	0.09	46	8.8	116.61	46 / 42	126.64 / 125.20	8 / 34	1 / 17	0 / 5
80	316	0.10	49	1.0	35.99	39 / 39	174.29 / 40.97	7 / 32	1 / 18	0 / 2
80	632	0.20	45	12.0	54.49	40 / 36	352.37 / 60.68	1 / 35	0 / 14	0 / 0
80	948	0.30	48	41.5	126.70	39 / 37	531.82 / 140.98	5 / 32	0 / 11	0 / 3
80	1264	0.40	48	12.0	333.87	34 / 33	649.61 / 214.10	4 / 29	0 / 8	0 / 2
80	1580	0.50	48	25.0	525.86	7 / 7	1210.23 / 532.48	1 / 6	0 / 2	0 / 1
80	1896	0.60	47	32.0	521.82	0 / 0	–	–	–	–
80	2212	0.70	45	18.8	677.73	0 / 0	–	–	–	–
80	2528	0.80	48	40.0	840.13	0 / 0	–	–	–	–
80	2844	0.90	46	44.5	984.10	0 / 0	–	–	–	–
			764 of 800			461/446		48/398	6/228	0/114
100	99	0.02	50	–	0.01	50 / 50	0.51 / 0.01	0 / 50	0 / 50	0 / 14
100	148	0.03	49	7.0	3.57	41 / 41	94.43 / 4.26	0 / 41	0 / 30	0 / 17
100	198	0.04	48	3.0	2.85	42 / 42	36.67 / 3.22	4 / 38	0 / 28	0 / 15
100	247	0.05	49	2.0	132.24	41 / 41	210.08 / 40.48	2 / 39	1 / 28	0 / 10
100	297	0.06	48	12.5	66.65	36 / 35	226.85 / 21.92	1 / 34	0 / 24	0 / 10
100	346	0.07	50	–	96.56	39 / 39	380.74 / 74.45	2 / 37	1 / 26	0 / 11
100	396	0.08	47	6.3	212.78	39 / 38	296.11 / 250.64	5 / 33	0 / 19	0 / 7
100	445	0.09	48	17.5	84.03	38 / 36	489.76 / 71.69	4 / 32	0 / 21	0 / 10
100	495	0.10	46	7.3	65.85	41 / 38	528.38 / 47.58	1 / 37	0 / 27	0 / 11
100	990	0.20	46	6.5	235.77	37 / 33	981.73 / 241.37	3 / 30	0 / 14	0 / 3
100	1485	0.30	47	33.7	483.22	0 / 0	–	–	–	–
100	1980	0.40	43	39.1	925.85	0 / 0	–	–	–	–
100	2475	0.50	45	43.0	1039.45	1 / 0	–	–	–	–
100	2970	0.60	44	50.8	1192.11	0 / 0	–	–	–	–
100	3465	0.70	33	30.6	1364.97	0 / 0	–	–	–	–
100	3960	0.80	28	35.6	1621.69	0 / 0	–	–	–	–
100	4455	0.90	27	37.8	1224.80	0 / 0	–	–	–	–
			748 of 850			405/393		22/371	2/267	0/108

Continued with Table 6

Table 5: Computational results for B&P algorithm compared to the monolith ("CDP-SEC") for instances with 40 to 100 vertices

Instance			B&P			CDP-SEC vs. B&P				
								Faster		
n	m	d	Solved	Gap	$\overline{\mathbf{T}}^*$	Solved	$\overline{\mathbf{T}}^*$	$> 1\times$	$\geq 10\times$	$\geq 100\times$
120	142	0.02	50	–	0.04	50 / 50	154.66 / 0.04	0 / 50	0 / 47	0 / 34
120	214	0.03	50	–	3.71	37 / 37	96.95 / 2.68	1 / 36	0 / 34	0 / 15
120	285	0.04	47	5.7	123.60	37 / 37	197.27 / 41.83	6 / 31	1 / 26	0 / 17
120	357	0.05	45	9.2	40.15	39 / 35	425.25 / 15.34	0 / 35	0 / 26	0 / 9
120	428	0.06	46	11.0	88.43	41 / 38	552.82 / 85.93	1 / 37	0 / 27	0 / 13
120	499	0.07	45	16.0	135.35	41 / 36	809.28 / 106.63	3 / 33	0 / 22	0 / 10
120	571	0.08	47	26.3	224.69	33 / 30	945.95 / 211.25	3 / 27	0 / 20	0 / 8
120	642	0.09	47	15.0	418.29	33 / 31	874.35 / 387.92	3 / 28	0 / 18	0 / 7
120	714	0.10	43	16.7	164.20	32 / 26	873.21 / 188.65	3 / 23	0 / 18	0 / 6
120	1428	0.20	42	42.4	626.15	0 / 0	–	–	–	–
120	2142	0.30	39	39.8	1080.82	3 / 2	1314.38 / 956.50	1 / 1	0 / 0	0 / 0
120	2856	0.40	32	40.5	1462.00	0 / 0	–	–	–	–
120	3570	0.50	25	41.6	1131.31	0 / 0	–	–	–	–
120	4284	0.60	24	38.7	1459.77	0 / 0	–	–	–	–
120	4998	0.70	18	52.7	1331.87	0 / 0	–	–	–	–
120	5712	0.80	12	52.7	1114.41	0 / 0	–	–	–	–
120	6426	0.90	6	47.8	1447.47	0 / 0	–	–	–	–
			618 of 850			346/322		21/301	1/238	0/119
140	194	0.02	50	–	70.72	39 / 39	92.89 / 0.43	0 / 39	0 / 33	0 / 25
140	291	0.03	48	10.0	18.62	46 / 45	492.28 / 5.27	1 / 44	0 / 41	0 / 28
140	389	0.04	47	7.0	33.54	37 / 36	851.51 / 23.89	1 / 35	0 / 33	0 / 23
140	486	0.05	41	13.8	170.20	33 / 28	729.01 / 51.60	2 / 26	0 / 21	0 / 10
140	583	0.06	45	17.6	139.26	28 / 26	987.73 / 85.04	2 / 24	0 / 17	0 / 6
140	681	0.07	46	24.3	198.05	28 / 26	1022.35 / 140.34	2 / 24	0 / 19	0 / 10
140	778	0.08	46	27.0	257.72	6 / 6	1659.86 / 181.41	1 / 5	0 / 4	0 / 1
140	875	0.09	40	25.8	254.81	8 / 6	2376.18 / 357.11	0 / 6	0 / 3	0 / 0
140	973	0.10	50	–	512.85	4 / 4	1304.13 / 44.52	0 / 3	0 / 3	0 / 2
140	1946	0.20	44	52.8	1050.86	1 / 0	–	–	–	–
140	2919	0.30	32	40.6	1251.37	0 / 0	–	–	–	–
140	3892	0.40	22	53.2	1418.21	0 / 0	–	–	–	–
140	4865	0.50	15	56.9	1496.19	0 / 0	–	–	–	–
140	5838	0.60	16	64.6	1986.02	0 / 0	–	–	–	–
140	6811	0.70	10	59.0	1443.63	0 / 0	–	–	–	–
140	7784	0.80	8	71.0	1140.67	0 / 0	–	–	–	–
140	8757	0.90	7	72.0	1539.68	0 / 0	–	–	–	–
			567 of 850			229/215		9/206	0/174	0/105
160	254	0.02	50	–	15.52	44 / 44	344.54 / 4.23	0 / 44	0 / 36	0 / 26
160	381	0.03	45	–	21.77	39 / 37	783.68 / 7.72	3 / 34	0 / 30	0 / 20
160	508	0.04	40	7.8	44.00	37 / 34	935.36 / 41.22	3 / 31	0 / 26	0 / 9
160	636	0.05	42	3.1	252.14	22 / 21	1094.00 / 233.02	1 / 20	0 / 17	0 / 4
160	763	0.06	46	–	342.72	5 / 4	2466.58 / 536.68	1 / 3	0 / 2	0 / 0
160	890	0.07	40	2.3	449.04	0 / 0	–	0 / 0	0 / 0	0 / 0
160	1017	0.08	41	7.0	560.79	1 / 1	2773.37 / 485.36	0 / 1	0 / 0	0 / 0
160	1144	0.09	41	19.9	657.92	10 / 10	989.29 / 399.53	2 / 8	1 / 5	0 / 3
160	1272	0.10	38	28.3	610.99	3 / 3	1380.67 / 345.95	1 / 2	0 / 1	0 / 1
160	2544	0.20	26	49.3	1342.28	0 / 0	–	–	–	–
160	3816	0.30	20	59.3	1248.03	0 / 0	–	–	–	–
160	5088	0.40	14	69.6	1546.65	0 / 0	–	–	–	–
160	6360	0.50	9	65.6	1352.24	0 / 0	–	–	–	–
160	7632	0.60	4	68.8	2610.25	0 / 0	–	–	–	–
160	8904	0.70	6	70.4	1542.32	0 / 0	–	–	–	–
160	10176	0.80	8	74.1	1431.05	0 / 0	–	–	–	–
160	11448	0.90	2	86.2	1259.62	0 / 0	–	–	–	–
			472 of 850			161/154		11/143	1/117	0/63
180	322	0.02	47	6.3	14.66	42 / 42	659.65 / 10.68	2 / 40	0 / 35	0 / 22
180	483	0.03	37	9.3	67.75	35 / 32	1464.17 / 71.50	0 / 32	0 / 22	0 / 14
180	644	0.04	40	4.8	92.43	15 / 14	2157.00 / 157.12	0 / 14	0 / 7	0 / 2
180	805	0.05	43	21.0	127.16	0 / 0	–	–	–	–
180	966	0.06	37	18.6	310.71	3 / 3	1521.01 / 11.26	0 / 3	0 / 3	0 / 2
180	1127	0.07	43	24.7	480.04	9 / 8	1617.36 / 91.05	0 / 8	0 / 6	0 / 4
180	1288	0.08	41	18.8	514.46	1 / 0	–	–	–	–
180	1449	0.09	37	26.7	731.47	0 / 0	–	–	–	–
180	1611	0.10	40	27.1	796.16	0 / 0	–	–	–	–
180	3222	0.20	16	58.2	1306.81	0 / 0	–	–	–	–
180	4833	0.30	14	62.4	1232.21	0 / 0	–	–	–	–
180	6444	0.40	3	71.7	1006.79	0 / 0	–	–	–	–
180	8055	0.50	6	73.8	1331.77	0 / 0	–	–	–	–
180	9666	0.60	4	72.8	1721.21	0 / 0	–	–	–	–
180	11277	0.70	6	85.7	943.15	0 / 0	–	–	–	–
180	12888	0.80	4	74.1	2103.98	0 / 0	–	–	–	–
180	14499	0.90	10	96.9	3009.12	0 / 0	–	–	–	–
			428 of 850			105/99		2/97	0/73	0/44

Table 6: Computational results for B&P algorithm compared to the monolith ("CDP-SEC") for instances with 120 to 180 vertices

[2] E. Israeli, R. K. Wood, Shortest-path network interdiction, Networks 40 (2002) 97–111.

[3] J. O. Royset, R. K. Wood, Solving the bi-objective maximum-flow network-interdiction problem, INFORMS Journal on Computing 19 (2007) 175–184.

[4] S. P. Borgatti, Identifying sets of key players in a social network, Computational and Mathematical Organization Theory 12 (2006) 21–34.

[5] A. Arulselvan, C. W. Commander, P. M. Pardalos, O. Shylo, Managing network risk via critical node identification, in: B. Rustem, N. Gulpinar (Eds.), Risk Management in Telecommunication Networks, Springer, 2007.

[6] A. Arulselvan, C. W. Commander, L. Elefteriadou, P. M. Pardalos, Detecting critical nodes in sparse graphs, Computers & Operations Research 36 (2009) 2193–2200.

[7] T. N. Dinh, Y. Xuan, M. T. Thai, E. K. Park, T. Znati, On approximation of new optimization methods for assessing network vulnerability, in: Proceedings of the 29th conference on Information communications, INFOCOM'10, IEEE Press, Piscataway, NJ, USA, 2010, pp. 2678–2686.

[8] B. Addis, M. Di Summa, A. Grosso, Removing critical nodes from a graph: complexity, results, and polynomial algorithms for the case of bounded treewidth, 2011. Submitted to Optimization Online.

[9] S. Shen, J. C. Smith, R. Goli, Exact interdiction models and algorithms for disconnecting networks via node deletions, Discrete Optimization 9 (2012) 172 – 188.

[10] T. E. Harris, F. S. Ross, Fundamentals of a method for evaluating rail net capacities, Research Memorandum RM-1573, The Rand Corporation, Santa Monica, CA, 1955.

[11] L. R. Ford, D. R. Fulkerson, Maximal flow through a network, Canadian Journal of Mathematics 8 (1956) 399–404.

[12] L. R. Ford, D. R. Fulkerson, A simple algorithm for finding maximal network flows and an application to the hitchcock problem, Canadian Journal of Mathematics 9 (1957) 210–218.

[13] R. Wollmer, Removing arcs from a network, Operations Research 12 (1964) 934–940.

[14] A. W. McMasters, T. M. Mustin, Optimal interdiction of a supply network, Naval Research Logistics Quarterly 17 (1970) 261–268.

[15] M. O. Ball, B. L. Golden, R. V. Vohra, Finding the most vital arcs in a network, Operations Research Letters 8 (1989) 73–76.

[16] R. K. Wood, Deterministic network interdiction, Mathematical and Computer Modelling 17 (1993) 1–18.

[17] P. S. Whiteman, A target selection tool for network interdiction, in: Proceedings of the 64th MORS Symposium.

[18] Y. S. Myung, H. J. Kim, A cutting plane algorithm for computing k-edge survivability of a network, European Journal of Operational Research 156 (2004) 579 – 589.

[19] C. Lim, J. C. Smith, Algorithms for discrete and continuous multicommodity flow network interdiction problems, IIE Transactions 39 (2007) 15–26.

[20] M. Di Summa, A. Grosso, M. Locatelli, Branch and cut algorithms for detecting critical nodes in undirected graphs, Computational Optimization and Applications (2012) 1–32.

[21] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, San Francisco, CA: W.H. Freeman and Company, 1979.

[22] D. L. Applegate, R. Bixby, V. Chvatal, W. J. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, Princeton, NJ, USA, 2007.

[23] L. Rousseau, M. Gendreau, D. Feillet, Interior point stabilization for column generation, Operations Research Letters 35 (2007) 660 – 668.

[24] M. E. Lübbecke, J. Desrosiers, Selected topics in column generation, Operations Research 53 (2005) 1007–1023.