Efficient methods for the distance-based critical node detection problem in complex networks

Glory Uche Alozie[1], Ashwin Arulselvan[1], Kerem Akartunalı[1], and Eduardo L. Pasiliao Jr[2]

[1]*Department of Management Science, University of Strathclyde, 199 Cathedral Street, G4 0QU, Glasgow, United Kingdom*

[2]*Munitions Directorate, Air Force Research Laboratory, Building 13, Eglin AFB, Florida 32542*

Abstract

An important problem in network survivability assessment is the identification of critical nodes. The distance-based critical node detection problem addresses the issues of internal cohesiveness and actual distance connectivity overlooked by the traditional critical node detection problem. In this study, we consider the distance-based critical node detection problem which seeks to optimise some distance-based connectivity metric subject to budgetary constraints on the critical node set. We exploit the structure of the problem to derive new path-based integer linear programming formulations that are scalable when compared to an existing compact model. We develop an efficient algorithm for the separation problem that is based on breadth first search tree generation. We also study some valid inequalities to strengthen the formulations and a heuristic to improve primal bounds. We have applied our models and algorithm to two different classes of the problems determined by the distance based connectivity functions. Extensive computational experiments on both real-world and randomly generated network instances, show that the proposed approach is computationally more efficient than the existing compact model especially for larger instances where connections between nodes consist of a small number of hops. Our computational experiments on both classes of distance-based critical node detection problem provide good numerical evidence to support the importance of defining appropriate metrics for specific network applications.

Keyword: Critical node problem, Distance connectivity, Integer programming, Lazy constraints, Breadth first search

1 Introduction

Assessment of system vulnerability to adversarial attacks has become an important concern to organisations, especially in the wake of security threats around the world. Natural occurrences such as environmental disasters and disease epidemics with their cascading effects impact the overall performance of systems, in which they occur. System developers consider as a matter of necessity the issue of survivability at the very stage of building new networks. This means designing networks in which communications between nodes can still be established after the failure of a pre-defined number of nodes or links [23]. Motivated by telephone line failures, Grötschel et al. [25] proposed mathematical models for the survivable network design problem which is the problem of designing minimum cost networks that satisfy certain connectivity requirements. The minimum cost $k-$connected network problem amongst other network problems was used to model the connectivity requirements as a means of representing survivability. Recent studies in survivable network design combine survivability and quality of service by imposing hop-constraints. Two such problems are the hop-constrained survivable network design problem (see e.g [24],[15],[33]) and the network design problem with vulnerability constraints (see e.g [23], [22]). The former seeks a minimum cost subgraph that contains k edge-disjoint (or node-disjoint) paths of length at most H for every given commodity (s,t). The latter which is a less expensive problem seeks a minimum cost subgraph in which for every commodity (s,t), there is a path of length at most H_{st} and after the removal of $k-1$ edges, the resulting graph has a path of length at most H'_{st}.

For existing networks, the issue of vulnerability assessment requires efficient surveillance strategies to ensure minimal disruption in the health and performance of the networks. An important strategy is to identify elements of a given system that are critical in maintaining optimum system performance. In other words, we are identifying the parts of a system whose failure would result in a break down of the

system. System performance has varying definitions depending on the topology of the system and the application being considered. Nevertheless, the underlying problem is that of identifying important system elements. The associated problem as studied in the optimisation community is termed critical node detection problem (CNP), as introduced in [6]. Given an undirected graph $G = (V, E)$ with $n = |V|$ nodes (vertices) and $m = |E|$ edges (arcs), the problem is to identify a subset of nodes of limited cardinality whose deletion results in a subgraph of maximum disconnectivity with respect to a predefined connectivity metric.

The CNP finds interesting applications in various areas. For example, identifying the critical nodes of a telecommunication network and taking them offline would stop the spread of a virus over the network. It is also useful in jamming wired telecommunication networks [7] to disrupt communication. In epidemiology, it is too expensive or impractical to vaccinate every member of a target population. The CNP provides a good immunisation strategy that targets individuals to immunise (or quarantine) in order to curtail the transmission of the virus in a contact network [7, 35]. In a social network context, the concept of key players relates to the idea of critical nodes. They correspond to the members of a social network who are influential for diffusion of information or potential targets for rumours [13]. This is of practical benefit in a public health context where crucial health information has to be disseminated across communities or to thwart the propagation of misinformation or malicious news. In security and defence operations, neutralising certain individuals in a terrorist network ensures disruption in communication eventually limiting their chances to launch a large scale coordinated attack [7]. Other areas in which the critical node detection problem has found application include drug design [12], emergency response and transport engineering [34].

An essential aspect of the critical node detection problem is the identification of a network property relevant to the network under study and an appropriate metric for its description [14]. This is primarily determined by the application context. In relation to network properties associated with studies on the CNP, existing studies can be grouped into two broad categories, namely fragmentation and pairwise distances. A graph is said to be fragmented if it consists of more than

one connected component, that is, if there is no path connecting some nodes in the graph. A significant volume of research has been carried out in relation to the goal of fragmentation, where popular objectives are either minimising the number of connected node pairs [7, 18, 43], maximising the number of connected components or minimising the size of the largest connected component [39, 40]. Other studies have focused on specific graph structures such as trees, split graphs and bipartite graphs [2, 17, 39]. Modified versions include the cardinality-constrained and similar variants which aim to minimise the number of deleted nodes while bounding the connectivity structure in the resultant subgraph [8, 19]. Deletions involving edges as opposed to nodes have also been studied for example in [34] and [41]. It is also noteworthy to remark that studies of centrality measures in the network analysis community address relevant problems, albeit from a different perspective [36]. In the aforementioned studies, the CNP was predominantly formulated as a integer programming (IP) problem. The models were then either solved using state-of-the-art IP solvers or some appropriate decomposition procedures. The CNP is known to be $\mathcal{NP}$-hard in general. A dynamic program was proposed for the case of trees [17]. Several heuristics were proposed to solve the general case on large scale instances (see, e.g., [4], [3], [1]).

Although fragmentation objectives of the CNP capture interesting system properties as reflected in the vast areas of applications and volume of research outputs, it is often not possible to achieve actual fragmentation. This is due to the topological structures of input networks under study as well as the available budget limiting the cardinality of the critical node set. These two aspects, which are inherent to most problem instances, severely undermine solution procedures designed towards achieving a fragmentation objective present in the CNP (see [13] and [44]). So, when it is infeasible or expensive to achieve the goal of fragmentation, the question one is compelled to ask is, "are we stuck in a maze or are there alternative means of assessing network vulnerability consequently leading to identification of critical nodes?". The second category of the CNP, hereafter referred to as the distance-based critical node detection problem (DCNP), gives a positive response to this question. It has been argued that for many real world applications in communication and social

networks, disconnectivity is not limited to fragmentation (that is, when there is no path connecting some pairs of nodes). If the distance between two nodes is above some threshold, they can be seen as practically disconnected [13, 44]. Moreover, Borgatti [13] in his study of key players in social networks further identified another limitation of fragmentation-based objectives, which is the overlooking of internal structure (cohesiveness) of components. In a similar study [38], the authors noted that certain structural deviations from the input network are left undetected by the traditional metrics of the CNP and concluded that there is a need for alternative choices based on distance metrics.

Despite the importance of the DCNP, very few studies have focused on it. A formalised study was proposed in Veremyev et al. [45]. The authors defined five distance-based connectivity objectives for the DCNP, namely,

i Minimise the number of node pairs connected by a path of length at most k

ii Minimise the Harary index (equivalently, the efficiency)

iii Minimise the sum of power functions of distances

iv Maximise the generalised Wiener index (equivalently, the characteristic path length)

v Maximise the shortest path length between nodes s and t

They proposed a generic IP model, which admits different distance objectives and an exact truncate-and-resolve algorithm to solve the model. In [5], complexity analyses of the DCNP connectivity objectives i, ii and iv above were presented for graphs with special structures, such as trees, paths and series-parallel graphs. The authors then proposed dynamic programming algorithms for polynomially and pseudo-polynomially solvable cases. In [29], a complementary mixed integer programming (MIP) formulation and a Benders decomposition algorithm were proposed for the distance-based connectivity objective (iv) above. In a very recent and independent work [37], Salemi and Buchanan (2020) proposed new integer programming formulations for the first distance based connectivity objective (i). Their solution technique uses

warm-start solutions obtained from a heuristic as well as a preprocessing procedure that identifies "non-critical" nodes for which corresponding variables are fixed to zero. Their work and ours both deal with objective function (i) and use the idea of tree generation for the separation problem. However, the models and algorithm proposed in this study can handle other objective classes of the DCNP as we demonstrate in Sections 3 & 4.

Contributions

In this study, we focus on the first two distance classes defined in [45], for which the authors proposed a generic compact formulation that is valid for any distance function. We will use this as a base model to benchmark our study. The base model has $O(|V|^3)$ variables and $O(|V|^2|E|)$ constraints and is sensitive to edge-dense graphs as well as graphs with short average path lengths. Therefore, we propose a path-based formulation to minimise the number of node pairs connected by a path of length at most L. We exploit the structure of the problem to develop an efficient separation routine to use within a branch-and-cut framework. We extend our framework to the second distance (Harary Index) class of the DCNP to develop a model which like the compact model is valid for other distance functions. We performed extensive computational experiments on both real-world and synthetic graphs for these two classes to compare our new models with the base model. In order to demonstrate the computational advantages of our approach, these two distance classes would be enough as they broadly generalise the other classes. Comparing our implementation for the two distance DCNP classes on some real-world graphs provides computational evidence to the impact of connectivity objective on the possibility and ease of solving the CNP. We also propose two families of valid inequalities. Our contributions are two-fold:

i From the methodological side, we introduce new formulations that use a decomposition approach. It was designed to exploit certain classes of distance functions (when L is small) as the natural compact model can become too big. We also introduce two families of valid inequalities for these problems.

ii We perform a computational study to test and compare the performance of the proposed formulations on a number of real-world and computer generated instances. The implementation includes a) a modified breadth first search (BFS) algorithm to separate hop-constrained path constraints b) a primal heuristics to improve upper bounds c) the valid inequalities introduced.

Organisation

The rest of the paper is organised as follows. In section 2, we formally describe the distance-based critical node detection problem and give definition of two distance connectivity metrics of interest. In section 3, we present the base mixed integer programming formulations introduced by [45], followed by our new path-based mixed integer programming formulations for the corresponding distance metrics. In section 4, we present our modified breadth-first search algorithm for solving the separation problem associated with the new model as well as valid inequalities. We present our computational study with thorough comparisons of models in sections 5 and 6. Concluding remarks and potential future perspectives are provided in section 7.

2 Problem definition

Let $G = (V, E)$ be a simple unweighted graph with a finite set V of nodes (or vertices) and a finite set $E \subseteq V \times V$ of edges and let B be the available budget on the set of critical nodes. The distance-based critical node detection problem aims to find a subset of nodes $R \subset V$ with $|R| \leq B$, whose removal minimises some distance-based connectivity measure $f(d)$ written mathematically as:

$$\textbf{DCNP} : \min_{R \subset V} \sum_{i,j \in V : i<j} f(d_{G^R}(ij)) : |R| \leq B$$

where $d_{G^R}(ij)$ is the distance or shortest path length between nodes i and j in the resultant subgraph $G^R = G[V \setminus R]$. For simplicity of notation, we use $f(d)$ instead of $f(d_{G^R}(ij))$ hereafter where $f(d)$ is assumed to be a non-increasing function of

shortest path distances between pairs of nodes in the graph. We assume that the input graph G is connected, otherwise, the largest connected component is used. We focus on two distance connectivity classes while we refer the interested reader to [45] for detailed descriptions of other distance functions and their application contexts.

Class 1. *Minimise the number of node pairs connected by a path of length at most* L_1

$$f(d) = \begin{cases} 1, & \text{if } d \leq L_1 \\ 0, & \text{if } d > L_1 \end{cases} \tag{1}$$

where $d = d_{G^R}(ij)$ defined as length of shortest path and L_1 is a given positive integer representing the cut-off hop distance. The special case where $L_1 = n - 1$ is the traditional CNP version of minimising number of connected node pairs. Interesting instances for this class would be graphs with a small diameter and thus a large proportion of nodes connected within a small number of hops. From an application perspective, it makes sense to use small values of L_1, say $L_1 \in \{3, 4\}$, in function (1) for instances with small diameter. We will refer to the DCNP version corresponding to this class of distance function as DCNP-1.

Class 2. *Minimise the Harary index or efficiency of the graph*

$$f(d) = \begin{cases} d^{-1}, & \text{if } d < \infty \\ 0, & \text{if } d = \infty \end{cases}$$

This metric is based on the assumption in communication network analysis that communication efficiency between node pairs is inversely proportional to the distance between them [16]. Typically, two disconnected nodes are at a distance of ∞. This corresponds to the fragmentation objective of the CNP. In [45], the authors introduced a threshold model, where two nodes separated by some distance threshold, L_2, cannot communicate directly, resulting in the following modified Harary distance

function.

$$f(d) = \begin{cases} d^{-1}, & \text{if } d \leq L_2 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

We will also be using this threshold model and we will refer to the DCNP version corresponding to this class of distance function as DCNP-2.

3 Integer programming formulations

We first present simplified versions of the generic mixed integer programming formulations defined in [45] for the two distance functions and then introduce our proposed path-based formulations.

3.1 Base IP formulations

Let $G = (V, E)$ be an input graph with $n = |V|$ nodes and $m = |E \subseteq V \times V|$ edges, also let B and L_r be given positive integers. Associated with any node i, we define variable x_i as

$$x_i = \begin{cases} 1, & \text{if node } i \text{ is deleted,} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Similarly, we define connectivity variables y_{ij}^l by

$$y_{ij}^l = \begin{cases} 1, & \text{if } (i,j) \text{ are connected by a path of length } \leq l \text{ in } G^R \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

We will first present the constraints defined in the base IP model for the problem

DCNP-r ($r = 1, 2$) as provided in [45].

$$y^1_{ij} + x_i + x_j \geq 1, \quad \forall (i,j) \in E,\ i < j \tag{5}$$

$$y^l_{ij} = y^1_{ij}, \quad \forall (i,j) \in E,\ i < j,\ l \in \{2, \ldots, L_r\} \tag{6}$$

$$y^l_{ij} + x_i \leq 1, \quad \forall (i,j) \in V,\ i < j,\ l \in \{1,\ 2, \ldots, L_r\} \tag{7}$$

$$y^l_{ij} + x_j \leq 1, \quad \forall (i,j) \in V,\ i < j,\ l \in \{1,\ 2, \ldots, L_r\} \tag{8}$$

$$y^l_{ij} \leq \sum_{t:(i,t)\in E} y^{l-1}_{tj}, \quad \forall (i,j) \notin E,\ i < j,\ l \in \{2, \ldots, L_r\} \tag{9}$$

$$y^l_{ij} \geq y^{l-1}_{tj} - x_i, \quad \forall (i,t) \in E,\ (i,j) \notin E,\ i < j,\ l \in \{2, \ldots, L_r\} \tag{10}$$

$$\sum_{i \in V} x_i \leq B \tag{11}$$

$$x_i \in \{0,\ 1\}, \quad \forall i \in V \tag{12}$$

$$y^l_{ij} \in \{0,\ 1\}, \quad \forall (i,j) \in V,\ i < j,\ l \in \{1, \ldots, L_r\} \tag{13}$$

Constraints (5)–(6) ensure that $y^l_{ij} = 1$ for adjacent node pairs (i, j), if neither of the nodes i nor j is deleted. Constraints (7)-(8) enforce y^l_{ij} to be zero if either node i or j is deleted. Constraints (9)-(10) ensure that there is a path of length at most l between non-adjacent nodes i and j iff node i is not deleted and there is a path of length at most $l - 1$ between t and j for some non-deleted node t in the neighbourhood of node i. Constraint (11) limits the cardinality of the critical node set to the budget B. Constraints (12)-(13) are binary restrictions on the decision variables. The observation that the binary restrictions on the y variables can be relaxed was made in [45]. Intuitively, once the x variables are fixed, constraints (5) and (10) will fix a subset of y variables to either 0 or 1. The minimisation function incentivises the remaining y variables to take a value of 0.

As noted in [45], considering initial shortest-paths between each node pair (i, j) in the input graph enables us to set connectivity variables $y^l_{ij} = 0$ for all $l < d_{ij}$ as well as defining constraints (9)-(10) for only $l \geq d_{ij}$. In addition to this, we note that only neighbours t of i having shortest path $d_{tj} \leq l - 1$ should be considered in constraints (9)-(10). These and leaf-node based considerations reduce the number of

variables and constraints in the model, thereby improving performance of standard IP solvers. The set of possible solutions to the DCNP-r ($r = 1, 2$) is then given by the set

$$\mathcal{P}_r := \{x \in \{0,1\}^n, y \in \{0,1\}^{n^2 \times L_r} : (x, y) \text{ satisfies(5) to (13)}\}$$

The IP base model with distance function (1), which we refer to as DCNP-1a is given as follows:

DCNP-1a

$$\min_{(x,y)\in\mathcal{P}_1} \sum_{i,j\in V: i<j} y_{ij}^{L_1} \tag{14}$$

Objective (14) minimises the number of node pairs whose shortest path distance is at most L_1. Similarly, a simplified version of the base model corresponding to distance connectivity metric (2), which we refer to as DCNP-2a is given as follows:

DCNP-2a

$$\min_{(x,y)\in\mathcal{P}_2} \sum_{i,j\in V: i<j} \left(f(1) y_{ij}^1 + \sum_{l=2}^{L_2} f(l) \left(y_{ij}^l - y_{ij}^{l-1} \right) \right) \tag{15}$$

Note that the objective function (15) computes the sum of the inverse of distances of all pairs of nodes. We assume the distance past a certain threshold (L_2) as ∞.

3.2 New path-based formulation

We exploit the structure of the first distance function (1) to develop a new path-based formulation, which we refer to as **DCNP-1b**. Since only paths of length at most L_1 is of importance here, by keeping track of paths within this threshold, we can guarantee that any given node pair (i, j) is L_1-distance disconnected iff at least one node along all candidate paths $\mathcal{P}_{L_1}(i, j)$ is deleted. Observe that the value of the distance function (1) defined for DCNP-1 is not explicitly dependent on the

shortest path lengths l. Instead, the distance function takes on constant values 1 or 0 depending on whether or not pairs of nodes (i, j) are connected by a path of distance $l \leq L_r$. Following this line of thought enables us to eliminate the need for the l- index in the distance connectivity variable y_{ij}^l used in the base model to arrive at an alternative model with fewer variables. This new model which is based on the paths connecting node pairs has exponentially many constraints but the constraints corresponding to non-adjacent node pairs can be treated as lazy constraints and separated efficiently. We define a new set of connectivity variables y_{ij} as follows:

$$y_{ij} = \begin{cases} 1, & \text{if } (i,j) \text{ are connected by a path of length} \leq L_1 \text{ in } G^R \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Using the former set of node deletion variables (3) along with the new connectivity variables (16), the path-based model for distance objective function (1) is formulated as follows:

DCNP-1b

$$\min \quad \sum_{i,j \in V: i<j} y_{ij} \tag{17}$$

$$\text{s.t.} \quad \sum_{r \in V(P)} x_r + y_{ij} \geq 1, \quad \forall P \in \mathcal{P}_{L_1}(i,j),\ (i,j) \in V,\ i < j \tag{18}$$

$$\sum_{i \in V} x_i \leq B \tag{19}$$

$$x_i \in \{0, 1\}, \quad \forall i \in V \tag{20}$$

$$y_{ij} \in \{0, 1\}, \quad \forall (i,j) \in V,\ i < j \tag{21}$$

Objective function (17) minimises the number of connected node pairs within the required threshold distance L_1. Constraint (18) ensures that node pairs (i, j) are L_1-distance disconnected iff at least one node along all paths of length less or equal to L_1 connecting i and j is deleted. Since there are potentially many such constraints, some of which are redundant, we explicitly model the non-redundant

constraints ($y_{ij} + x_i + x_j \geq 1$) for edges and leave the rest to be identified in a separation routine. The budgetary constraint limiting the cardinality of the critical node set is represented by constraint (19) while constraints (20)-(21) are same with the constraints (12)-(13) of the base model. Following the same argument in [45], the integrality constraints can be relaxed for the connectivity variables y_{ij}. Note that the lower bound of the compact model DCNP-1a will be better than that of DCNP-1b as constraints (5) and (10) of the compact model would completely imply constraint (18). However, constraints (9) and (10) make the compact model grow with the graph size. We can get computationally more efficient solutions using DCNP1-b as the corresponding LP relaxation is relatively smaller, which we exploit to compute quicker bounds. These observations are supported in our experiments. Finally, for the second distance class (2), our new path-based formulation is defined as follows:

DCNP-2b

$$\min \quad \sum_{i,j \in V: i<j} \left(f(1) y_{ij}^1 + \sum_{l=2}^{L_2} f(l) \left(y_{ij}^l - y_{ij}^{l-1} \right) \right) \tag{22}$$

$$\text{s.t.} \quad \sum_{r \in V(P)} x_r + y_{ij}^{|P|} \geq 1, \quad \forall P \in \mathcal{P}_{L_2}(i,j),\ i,j \in V,\ i<j \tag{23}$$

$$y_{ij}^{l-1} \leq y_{ij}^l, \quad \forall (i,j) \in V,\ i<j,\ l \in \{2,\ldots,L_2\} \tag{24}$$

$$y_{ij}^l = y_{ij}^1, \quad \forall (i,j) \in E,\ i<j,\ l \in \{2,\ldots,L_2\} \tag{25}$$

$$\sum_{i \in V} x_i \leq B \tag{26}$$

$$x_i \in \{0,\,1\}, \quad \forall i \in V \tag{27}$$

$$y_{ij}^l \in \{0,\,1\}, \quad \forall (i,j) \in V,\ i<j,\ l \in \{1,\ldots,L_2\} \tag{28}$$

The ideas behind constraints (23); (26)- (28) are similar to those of constraints (18)- (21). Constraints (24) ensure that there is a path of length l, if there is a path of length $l-1$ while constraint (25) is similar to that of (6). We note that the new path-based model DCNP-2b is a valid formulation for other distance functions $f(d)$ and not just the Harary index. In particular, DCNP-2b is valid for distance function (1).

Exploiting the structure of the distance function (1) allows us to model DCNP-1b with fewer variables. We will not gain much in DCNP-1b by including the l-indexed variables used in DCNP-2b. Let

$$\mathcal{Q} := \{(\mathbf{y}, \mathbf{x}) : (\mathbf{y}, \mathbf{x}) \text{ satisfies constraints (23) to (28)}\}$$

where $\mathbf{y} = \mathbf{y}^L, \ldots, \mathbf{y}^1$. Then we can think of the feasible space of DCNP-1b as

$$Proj_{(\mathbf{y}^L, \mathbf{x})}\mathcal{Q} := \{(\mathbf{y^L}, \mathbf{x}) : \exists(\mathbf{y}^{L-1}, \ldots, \mathbf{y}^1) : (\mathbf{y}, \mathbf{x}) \in \mathcal{Q}\}$$

If we do Fourier elimination of variables systematically starting from y_{ij}^1 all the way to y_{ij}^{L-1}, we will get exactly DCNP-1b with no changes. On the other hand, if we interpret y_{ij} as continuous variables that define the value of the distance function between nodes i and j, then we can model DCNP-2b similar to DCNP-1b. Constraint set (23) should now be

$$\sum_{r \in V(P)} f(|P|)x_r + y_{ij} \geq f(|P|) \quad \forall P \in \mathcal{P}_{L_2}(i,j),\ i,j \in V,\ i < j \tag{29}$$

One could then remove the dependency on ℓ in the y variables. This can also accommodate positive edge weights in the distance measure:

$$\sum_{r \in V(P)} f(d)x_r + y_{ij} \geq f(d) \quad \forall P \in \mathcal{P}_L(i,j),\ (i,j) \in V,\ i < j \tag{30}$$

where d is the edge-weighted length of path P. The binary distance-based y_{ij}^l variables can be viewed as a disaggregated version of the new continuous y_{ij} variables. Thus, for edge weighted graphs with positive integral edge weights, an alternative aggregated formulation to the edge-weighted version of DCNP-2b would be derived by the aggregated continuous y_{ij} variables replacing y_{ij}^l and constraints (23)–(25) replaced by (30). Consequently, constraint (28) is relaxed to $y_{ij} \in [0,1]$ and the

objective function (22) becomes

$$\min \sum_{i,j \in V: i<j} y_{ij}$$

However, such a formulation would result in poor lower bounds. In order to see this, observe that for every solution to DCNP-2b there exist a solution to the aggregated formulation with the same objective value by setting $y_{ij} = \max(f(l)y^l_{ij})$. Although, theoretically DCNP-2b is tighter than the aggregated (reduced) formulation, it would suffer from scalability as it has pseudopolynomially many variables for arbitrary edge weights. It is not clear when the trade-off between the size of the formulation and its tightness would begin to payoff. Intuitvely, one could expect the DCNP-2b to outperform the reduced model when the distance threshold is relatively small. This investigation, however, needs to be undertaken as an independent computational study.

4 Solution Methods

In this section, we present details of the separation routine for the path-based formulations. We also present a heuristic framework for generation of good incumbent solution as well as valid inequalities for improvement of lower bounds.

4.1 Separation algorithm

Instead of solving a hop-constrained shortest path problem [26] or enumerating all paths of certain length for every pair of nodes in order to generate cuts, we use a customised approach to separate violated lazy cuts. This approach involves in generating a breadth first search tree for every candidate node $v \in V$. At each level i of the BFS tree rooted at v, there are k_i nodes $\{l^i_1, \ldots, l^i_{k_i}\}$ that are at a distance of i from v (see Figure 1). The unique path from a node in the BFS tree to the root node gives us an inequality of type (18) or (23). For DCNP-1, since we are only interested in paths up to a specific length L_1, we stop the traversal up to that particular depth.

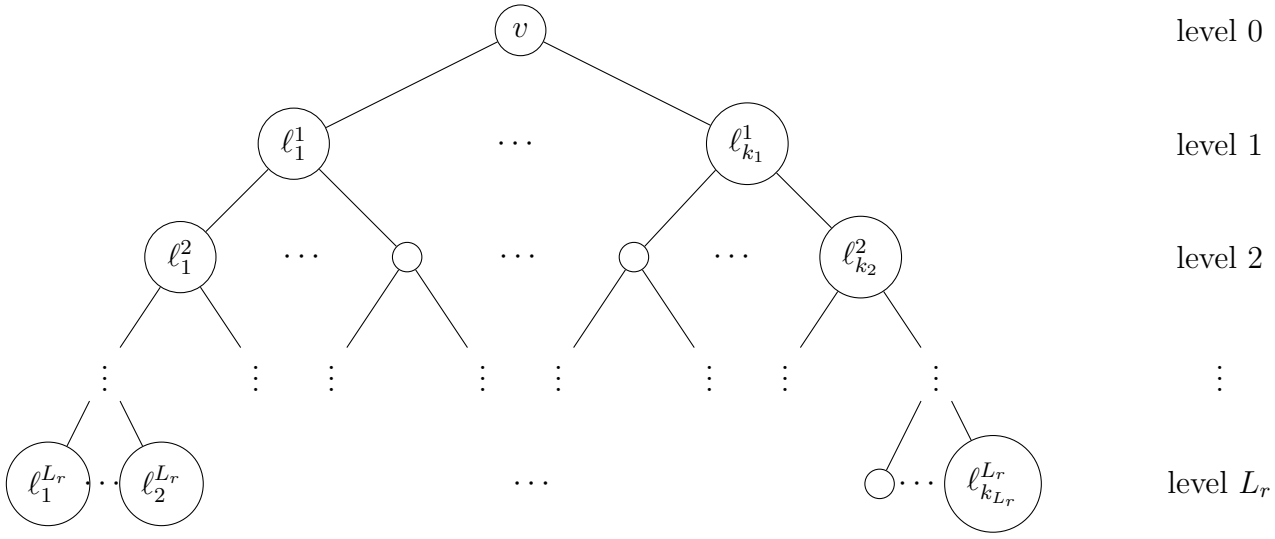


Figure 1: An example of a k-depth BFS tree generation

For DCNP-2, we continue traversal up to depth L_2. In the BFS tree, we identify the path from the root node i to the nodes in level $\ell = 1, \ldots, L_r$ and if this path is violated, we add it as a cut. The generation of these BFS trees is much more efficient and we get multiple paths for one such tree. BFS tree can be generated in $O(|E|)$ as compared to solving shortest path that will take $O(|E| + |V| \log |V|)$. Since we only explore L_r levels of the BFS tree, it takes far less time than $O(|E|)$ to generate these trees. A pseudocode of the algorithm is given by Algorithm 1.

The algorithm begins by selecting a root node r from the set of candidate root nodes and constructs a BFS tree up to a specified depth (or tree level) L_r. The algorithm follows a standard BFS algorithm with the only difference being that it keeps track of the depth of each discovered node (lines 6 & 10) with which it determines when to terminate the current tree generation and also to separate the corresponding path inequalities. For each node t that is being visited, the algorithm proceeds in one of three possible ways depending on its tree level $l[t]$.

Case 1: $l[t] = 1$ which implies that node t is a direct neighbour of the root node r that is $(r, t) \in E$. In this case, we only add t to the queue Q and mark it as visited

Algorithm 1: Separation algorithm for the proposed model

Input : Graph $G = (V, E)$, Incumbent $(\tilde{x}, \tilde{y})$ or LP-relaxation solution $(\bar{x}, \bar{y})$, set of candidate root nodes R_c, and tree depth L_r

Output: Queue of violated inequalities Q_c

for $r \in R_c$ **do**
  $Q \leftarrow \{r\}$; `// initialise queue` Q `with root node` r
  $T \leftarrow \{r\}$; `// mark` r `as visited`
  $l[r] = 0$;
  while $Q \neq \emptyset$ **do**
    $s \leftarrow Q.remove$; `// retrieve the first element in queue` Q
    `/* Explore node` s`, where` δ_s `denotes neighbours of` s `*/`
    for $t \in \delta_s \setminus T$ **do**
      $l[t] = l[s] + 1$;
      if $l[t] = 1$ **then**
        $Q.add(t)$; `// add` t `to queue` Q
        $T \leftarrow T \cup \{t\}$; `// mark` t `as visited`
      else if $l[t] \in [2, L_r]$ **then**
        $Q.add(t)$;
        $T \leftarrow T \cup \{t\}$; `// mark` t `as visited`
        `/* check violation of (18) (resp. (23) for DCNP-2b) for the path` P_{rt} `*/`
        if $\sum_{i \in V(P_{rt})} \tilde{x}_i + \tilde{y}_{rt} < 1$ *(resp.* $\sum_{i \in V(P_{rt})} \tilde{x}_i + \tilde{y}_{rt}^{l[t]} < 1$*)* **then**
          $Q_c.add(\sum_{i \in V(P_{rt})} x_i + y_{rt} \geq 1)$
          (resp. $Q_c.add(\sum_{i \in V(P_{rt})} x_i + y_{rt}^{l[t]} \geq 1)$);
        end
      else
        $Q \leftarrow \emptyset$;
        break;
      end
    end
  end
end

(lines 11-13) since path inequalities for edges are already present in the formulation. **Case 2:** $1 < l[t] \leq L_r$, which implies that $(r,t) \notin E$ but hop distance between root node r and t is less than or equal to L_r. This is the case of interest. Hence we not only add t to the queue Q and mark it as visited, but we also check if path inequalities (18) or (23) of the corresponding shortest path P_{rt} are violated (lines 14-17) and add the violated inequality to the queue Q_c.
Case 3: $l[t] > L_r$, this implies that all nodes reachable within hop distance L_r from the root r have been explored. Therefore, we stop the current tree generation (lines 18-20) and return to line 3 where we select the next root and begin the process all over.

We note a slight difference in the implementation of Algorithm 1 for separation of integer and fractional solutions. Observe that in any incumbent solution, the variables are either 0 or 1. Since constraints (18) or (23) are only violated when left-hand sum is less than 1, only nodes with value $\tilde{x}_i = 0$ in any given incumbent solution would potentially lead to violated path constraints. Therefore, set of candidate root nodes R_c in step 3 of Algorithm 1 consists only of nodes with value $\tilde{x}_i = 0$ in the current integer solution. Following this thought, in exploring a node in step 9, we also limit it to unvisited neighbors whose $\tilde{x}$ variable value in the current incumbent solution is zero. Hence, only nodes with zero solution value ($\tilde{x}_i = 0$) feature in the constructed trees. This helps us to detect and add all violated constraints for any given integer solution and avoid spending time in unpromising branches.

The separation problem for the most violated cut of a fractional solution involves solving a weighted shortest path problem in which the edge weights are given by the LP relaxation values $\bar{x}$. This can still be done in polynomial time through a transformation to a directed graph as the weights are positive. This, however, increases the graph size and the shortest path problem has to be solved for all pairs of nodes. We instead adapt our BFS algorithm as a heuristic to separate a violated fractional solution. The BFS trees are built based on the LP relaxation values $\bar{x}_i$, i.e., candidate root nodes with smaller LP relaxation values are chosen first for BFS tree generation. Also, nodes are explored in increasing order of their neighbours' LP-relaxation values, this means that the unvisited neighbours with

smaller LP relaxation values $\bar{x}_i$ are visited first before other neighbours. This ensures that the most violated constraints for all paths of a particular hop distance between node pairs are separated. Furthermore, for LP relaxation solutions, we set limits on the number of cuts added at which we end the call on the separation algorithm and re-solve the problem. We set parameter for cut limit to 150 cuts, after trying different values within the neighborhood of ± 100. For integer solutions, we only set this limit to stop the current BFS tree generation and return to line 3 to generate a new BFS tree rooted at the next candidate root node. Nevertheless, this cut limit is only applied after the optimisation process begins branching to ensure all violated cuts are separated. This is because at the root node of a branch-and-bound tree, many integer solutions are infeasible to the original problem, hence more constraints would be potentially violated.

For the new reduced model (that is path-based model for DCNP-2 with the aggregated continuous y_{ij}) mentioned in the latter part of Section 3.2, the separation technique for the hop-based distances for constraints (29) would be similar to the initial discussions made in this section. Specifically, the BFS tree construction would be done in the same manner. For integer solutions, since only nodes with $\tilde{x}_i = 0$ feature in the BFS tree, it suffices to check that $\tilde{y}_{rt} < f(l[t])$ in line 17 of Algorithm 1 for node t at hop distance $l[t], (1 < l[t] \leq L_r)$ from root node r and then to add the corresponding path inequality

$$\sum_{i \in V(P_{rt})} f(l[t]) x_i + y_{rt} \geq f(l[t]) \tag{31}$$

where violated. Using the same presentation made for the old path based model in this section, we can adapt the BFS algorithm as a heuristic to separate fractional solutions. One needs to check if

$$\sum_{i \in V(P_{rt})} f(l[t]) \bar{x}_i + \bar{y}_{rt} < f(l[t])$$

and then add the corresponding constraint (31). For edge-weighted distances, we

cannot use BFS trees. Instead, shortest path trees would need to be generated using an appropriate algorithm such as Dijkstra's while keeping track of the shortest paths as well as their lengths. For integer solutions, shortest path trees can be generated in a manner similar to the generation of BFS trees by using an auxiliary graph of the input graph where nodes with $\tilde{x}_r = 1$ are removed from the graph. Using the information on the shortest paths, the path constraint (30) corresponding to the shortest path from the root node to nodes in each shortest path tree can be separated based on the same ideas presented for the hop-based version.

4.2 Valid inequalities

In addition to the lazy cuts, we propose two families of strong valid inequalities that are not implied by the constraints in the formulations. These inequalities can be viewed as $\{0, \frac{1}{2}\}$ (or $\{0, \frac{2}{3}\}$)-Chvátal-Gomory (CG) cuts, where the constraints are multiplied by constants in the set $\{0, \frac{1}{2}\}$ (or $\{0, \frac{2}{3}\}$), added together and then rounded up. Note that this generalises the procedure that was provided in [18] and we can obtain a broader class of inequalities. For the purpose of our computational experiments, we focused on odd cycles of lengths 3 and 5 as larger holes are atypical in small world networks.

4.2.1 Odd holes of length 3

Let H be an odd hole of length 3 in $G = (V, E)$ with node set $V(H)$ and edge set $E(H)$. Also for simplicity, let $V(H) = \{1, 2, 3\}$ and $E(H) = \{12, 23, 13\}$.

Proposition 1. *The following is a valid inequality for the path-based constraints in $E(H)$:*

$$x_1 + x_2 + x_3 + y_{12} + y_{13} + y_{23} \geq 2 \tag{32}$$

Proof. Consider the following valid inequalities for each edge in H:

$$x_1 + x_2 + y_{12} \geq 1$$
$$x_1 + x_3 + y_{13} \geq 1$$
$$x_2 + x_3 + y_{23} \geq 1$$

Summing together we get $2x_1 + 2x_2 + 2x_3 + y_{12} + y_{13} + y_{23} \geq 3$. Then by applying $\{0, \frac{1}{2}\}$-CG procedure we obtain inequality (32). It is easy to show that inequality (32) is not implied by inequalities (18)-(21) of DCNP-1b. The fractional point, $x_i = 1/2$ for all $i \in V(H), y_{ij} = 0$ for all $i, j \in E(H)$, is feasible to the LP-relaxation of DCNP-1b but violates (32). □

4.2.2 Odd holes of length 5

Let H be an odd hole of length 5 in $G = (V, E)$ with node set $V(H)$ and edge set $E(H)$. Also for simplicity, let $V(H) = \{1, 2, 3, 4, 5\}$, $E(H) = \{12, 23, 34, 45, 15\}$ and let $P_2(i, j)$ denote the path of length 2 connecting i and j in H. Formulation (17)-(21) contains the following constraints:

path of length 1: $x_i + x_j + y_{ij} \geq 1 \quad i, j \in E(H)$
path of length 2: $x_i + x_j + x_k + y_{ij} \geq 1 \quad i, j \notin E(H), k \in V(P_2(i, j)) \setminus \{i, j\}$

Proposition 2. *The following is a valid inequality for path-based constraints for paths of length 2 in $E(H)$:*

$$2x_1 + 2x_2 + 2x_3 + 2x_4 + 2x_5 + y_{13} + y_{24} + y_{35} + y_{14} + y_{25} \geq 4 \tag{33}$$

Proof. Consider the following valid inequalities corresponding to paths of length 2

in H:

$$x_1 + x_2 + x_3 + y_{13} \geq 1$$
$$x_2 + x_3 + x_4 + y_{24} \geq 1$$
$$x_3 + x_4 + x_5 + y_{35} \geq 1$$
$$x_1 + x_5 + x_4 + y_{14} \geq 1$$
$$x_2 + x_1 + x_5 + y_{25} \geq 1$$

Summing together we get $3x_1 + 3x_2 + 3x_3 + 3x_4 + 3x_5 + y_{13} + y_{24} + y_{35} + y_{14} + y_{25} \geq 5$. Then applying $\{0, \frac{2}{3}\}$-CG procedure, we obtain inequality (33). Inequality (33) is not implied by inequalities (18)-(21) of DCNP-1b. The fractional point, $x_i = 1/3$ for all $i \in V(H), y_{ij} = 1/3$ for all $i, j \in E(H)$, $y_{ij} = 0$ for all $i, j \notin E(H)$, is feasible to the LP-relaxation of DCNP-1b but violates (33). □

Following the same procedure, the corresponding odd hole inequalities for DCNP-2b formulation (22)–(28) are:

$$x_1 + x_2 + x_3 + y^1_{12} + y^1_{13} + y^1_{23} \geq 2 \tag{34}$$

and

$$2x_1 + 2x_2 + 2x_3 + 2x_4 + 2x_5 + y^2_{13} + y^2_{24} + y^2_{35} + y^2_{14} + y^2_{25} \geq 4 \tag{35}$$

Separation of the odd hole inequalities is based on simple enumeration whereby a pool of cycles of given length is generated based on the cycle enumeration scheme proposed in [31] and the corresponding odd hole inequalities are routinely checked for violations.

4.3 Primal heuristic

Generation of good incumbent solution helps in pruning branch-and-bound nodes. Thus we propose a primal heuristic which incorporates information from LP-relaxation

solutions into centrality-based ranking to arrive at a good primal bound. First, we extend the budget requirement to $\hat{B} = 1.5B$ instead of B, then based on the degree centrality measure, we obtain the top $\hat{B}$ ranking nodes. Using the LP-relaxation values of those nodes as selection probability, we randomly select B distinct nodes. We fix the x variables of the selected nodes to 1 ($x_i = 1$) while the x variables for the rest of the nodes are fixed to zero.

5 Computational experiments

5.1 Hardware & Software

Our computational study was performed on an HP computer equipped with Windows 8.1 x64 operating system, an Intel Core i3-4030 processor(CPU 1.90 GHz) and RAM 8GB. The models and algorithms were written in Python 3.6 (Anaconda 5) using Gurobi 8.1.0 [27] as optimisation suite. We use NetworkX [28] for random graph generation as well as for drawing and manipulating of the graphs. All experiments were run with time limit of 3600 seconds thus the CPU times presented are in seconds.

5.2 Test Instances

Our test instances comprise both real-world and randomly generated graphs. Real-world instances are a subset of networks from the Pajek and UCINET datasets [11, 42]. All instances are connected graphs or the largest connected component of the original graph if the original graph is disconnected.

- **Hi-tech**($|V| = 33, |E| = 91$): Friendship network of employees in a hi-tech firm [11, 42].
- **Karate** ($|V| = 34, |E| = 78$): A social network of a karate club at a U.S University in the 1970s [11, 42, 46].
- **Mexican** ($|V| = 35, |E| = 117$) A network of relations (family, political and business) of political elite in Mexico [11, 42].

- **Sawmill** ($|V| = 36, |E| = 62$): Communication network of employees within a small enterprise [11, 42].
- **Chesapeake**($|V| = 39, |E| = 170$): Chesapeake Bay Mesohaline network [9, 11].
- **Dolphins** ($|V| = 62, |E| = 159$): A social network that represents frequent associations between dolphins in a community in New Zealand [11, 32, 42].
- **Lesmiserable** ($|V| = 77, |E| = 254$): Network of co-appearance of characters in the novel Les Miserable [30]
- **Santafe** ($|V| = 118, |E| = 200$): Collaboration network of scientists at the Santa Fe Institute [21].
- **SmallWorld** ($|V| = 233, |E| = 994$): A citation network [11].
- **Sanjuansur** ($|V| = 75, |E| = 155$), and **attiro** ($|V| = 59, |E| = 128$) : Social networks of families in a rural area in Costa Rica [11, 42].
- **LindenStrasse** ($|V| = 232, |E| = 303$): Network of friendly relationships between characters of the soap opera "Lindenstrasse" [11].
- **USAir97** ($|V| = 332, |E| = 2126$): Transportation network of US airlines [11].
- **NetScience** ($|V| = 379, |E| = 914$): Co-authorship network of scientists in science [42].

We also used 3 classes of random graphs which were generated using the networkX random graph generators. For any graph in a particular class, 10 different instances were generated and results averaged and compared across those instances. The three classes are described as follows:

I **Barabasi-Albert random graphs:** The Barabasi-Albert model [10] is known for its preferential attachment mechanism wherein nodes with high degree have a higher propensity to be connected to a new node as the graph is grown.

The degree distribution, defined to be the fraction of nodes with degree k, of the Barabasi-Albert model is known to follow a power law distribution $p_k \approx k^{-3}$. Instances of this random graph class were generated using networkX random graph generator with parameters $n = |V|$ and p, which denote the graph size and the number of edges to attach from a new node to existing nodes respectively. For the Barabasi-Albert graph class, two sets of graphs were generated namely, **ba1** and **ba2**, both with parameter $n = 100$ but with $p = 5$ and $p = 10$ respectively.

II **Erdos-Renyi random graphs:** Erdos-Renyi model [20] for random graph generation defines a set of graphs having the same parameters $n = |V|$, the size of the graph and p, probability of adding an edge between any two node pairs. Starting with an empty graph with $|V|$ nodes, the model creates a random graph by adding an edge between every pair of nodes $i, j \in V$ with probability p. The degree distribution of Erdos-Renyi graphs follow the Poisson distribution. Instances were generated using networkX random graph generator with parameter n denoting the graph size and p denoting the probability of edge creation. Two sets of graphs were generated using the Erdos-Renyi model. The first which is named **er1** has parameters $n = 80$ and $p = 0.15$ while the second named **er2** has parameters $n = 200$ and $p = 0.05$.

III **Uniform random graphs:** Given two input parameters n and m, the uniform random graph model $G_{n,m}$ returns a graph selected uniformly at random from set of all graphs having n nodes and m edges. Three sets of instances were generated using networkX random graph generator that takes n and m as parameters. The sets of instances are named **gnm1**, **gnm2** and **gnm3**.

The properties of the random graphs averaged over the 10 instances generated for each set are summarised in Table 1.

Graph	n	m	diam	density (%)	k-DistConn (%)	efficiency(%)
ba1	100	475	4.0	9.6	99.9	49.6
ba2	100	900	3.0	18.0	100	58.4
er1	80	470	3.0	14.9	100	55
er2	200	1004	4.0	5.0	97.7	42.8
gnm1	200	1000	4.2	5.0	97.9	42.8
gnm2	300	1500	4.4	3.3	94.1	39.6
gnm3	300	2000	4.0	4.5	99.6	43.4

Table 1: Characteristics of each set of random graph instances

5.3 Parameter settings and preprocessing

For our computational experiments, we set the distance threshold $L_1 = 3$ for DCNP-1 models, and for DCNP-2 model, we set L_2 to the diameter of the input graph. For experiments on randomly generated graphs, we set budget B to 5% and 10% of graph size. We varied this percentage for the real world graphs from 1% to 10% of number of nodes in the input graph. The budget values for different instances are specified on the corresponding tables and figures in Section 6. We also explored the enhancements discussed in Section 3.1 for the base model as a preprocessing stage prior to running the Gurobi optimiser while fixing values for nodes with degree one to zero ($x_i = 0$) for both models.

6 Results and Discussion

The computational experiments were performed for varying sizes (nodes, edges) and classes of graphs with different edge densities and diameters. We compare performance of the base formulation (DCNP-1a and DCNP-2a) implemented with the suggested enhancements and our path-based formulations (DCNP-1b and DCNP-2b) labeled as ECM and PBM respectively. In each table, along with graph characteristics such as number of nodes, edges and diameter, we present computational times in seconds and/or percentage gaps for both the ECM and PBM for different budget settings. Columns labelled *InitObj* represent the initial objective values in the input

graph prior to solving the model. For distance function class 1, this is the percentage of node pairs connected by paths of length at most L_1 while for class 2, it is the initial communication efficiency of the input graph. Similarly, columns labelled *FinObj* represent the final objective value (in percentage) at the end of optimisation or the best objective realised within the specified time limit. Columns labelled *ECMt* and *PBMt* represent the computational time for the base and path-based models respectively. Recall that all experiments were run with time limit of 3600 seconds, hence, where the optimisation process could not terminate within the given time limit, the corresponding entry is marked $' > 3600'$. Similarly, when a problem instance runs out of memory, we indicate this by an "M". For instances which were unsolved within the specified time limit, the percentage gaps are calculated as

$$\%gap = 100 \cdot \frac{BestObj - FinalLowerBound}{FinalLowerBound}\%$$

For the synthetic graphs, the percentage gaps are averaged over all ten instances generated for each graph class.

6.1 Comparisons of results for DCNP class 1 models (DCNP-1a and DCNP-1b)

					B=0.05n			B=0.1n		
Graph	**n**	**m**	**diam**	**InitObj**	**FinObj**	**ECMt (s)**	**PBMt (s)**	**FinObj**	**ECMt (s)**	**PBMt (s)**
Hi-tech	33	91	5	88.3%	75.2%	**0.12**	0.2	55.5%	**0.59**	0.64
Karate	34	78	5	85.6%	57.8%	0.16	**0.13**	26.2%	**0.11**	**0.11**
Mexican	35	117	4	98.0%	88.6%	0.31	**0.23**	60.2%	**0.33**	0.39
Sawmill	36	62	8	63.0%	34.1%	0.08	**0.06**	21.4%	**0.08**	**0.08**
Chesapeake	39	170	3	100.0%	93.9%	**0.6**	0.61	69.1%	1.36	**1.18**
Dolphins	62	159	8	58.5%	43.4%	1.91	**1.03**	30.8%	1.91	**1.71**
Lesmiserable	77	254	5	85.4%	31.8%	**0.52**	0.92	11.0%	**0.97**	1.03
Santafe	118	200	12	32.9%	4.4%	**0.2**	0.45	1.7%	**0.59**	0.72
Sanjuansur	75	155	7	48.7%	28.9%	0.44	**0.35**	16.5%	**0.39**	0.58
Attiro	59	128	8	68.0%	43.4%	0.39	**0.31**	26.0%	0.67	**0.61**

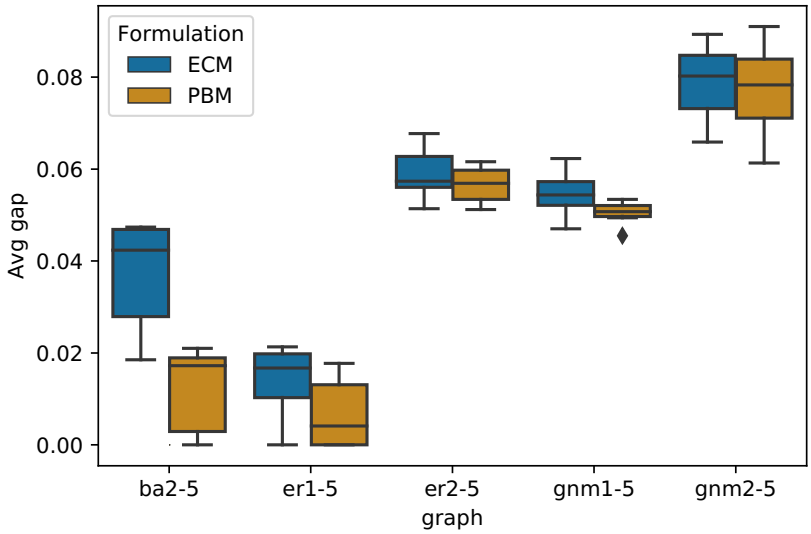
Table 2: Results of ECM and PBM models with DCNP-1 on realworld networks (small size)

The first set of experiments provides comparison between the base model (ECM) and the path-based model (PBM) for distance function class 1. Tables 2 & 3 summarise results for real world network instances. What we see from Table 2 is that

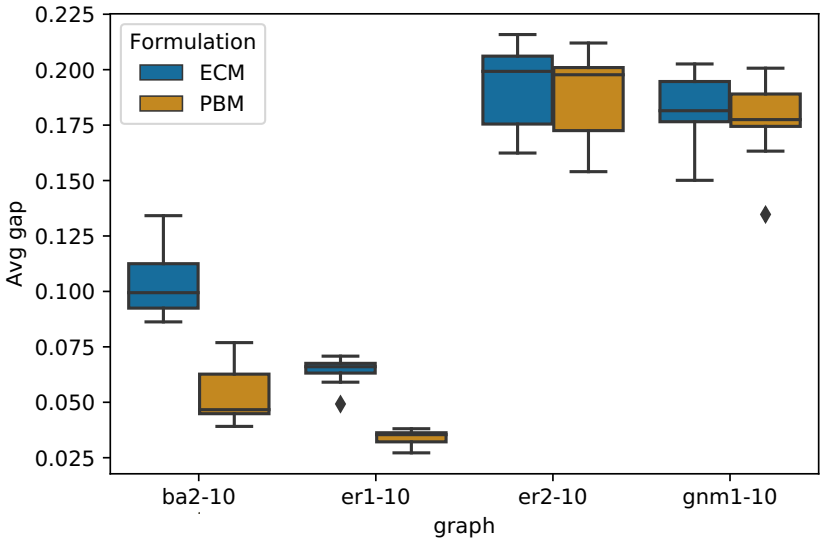
Graph	**n**	**m**	**diam**	**InitObj**	**B**	**FinObj**	**ECMt (s)**	**PBMt (s)**
USAir97	332	2126	6	84.8%	0.1n	5.64%	> 3600	**1277.38**
					0.05n	19.33%	426.88	**377.83**
					0.03n	39.35%	952.22	**692.04**
					0.02n	49.57%	**515.49**	582.16
					0.01n	63.90%	456.41	**240.67**
LindenStrasse	232	303	13	12.1%	0.04n	4.70%	**0.91**	1.17
					0.03n	6.15%	**0.88**	1.82
					0.015n	8.32%	**0.86**	1.25
SmallWorld	233	994	4	95.2%	0.1n	6.27%	144.56	**117.54**
					0.04n	19.45%	78.66	**53.61**
					0.03n	23.41%	46.75	**18.46**
					0.015n	40.56%	85.17	**41.17**
NetScience	379	914	17	13.3%	0.03n	4.32%	**4.08**	5.11
					0.01n	8.43%	4.72	**4.52**

Table 3: Results of ECM and PBM models with DCNP-1 on real world networks (medium size)

the path based model (PBM) competes well with the base model for small graph sizes (< 120 nodes). As the real-world network instances increase in size with increase in initial objective, the performance of PBM over ECM becomes more glaring. For example, in **Smallworld** and **USAir97** graphs with initial $\%k$-distConn greater than 80%, the PBM on average is almost twice as fast as the ECM. In particular, for budget setting of 0.1n, ECM is fails to solve the **USAir97** instance terminating with a 10.3% gap whereas this is solved under 1300 seconds by PBM (see Table 3). This is also the case for the smaller class of Barabasi-Albert random network (**ba1)**, in which PBM is more than thrice as fast as the ECM for B=0.05n (**PBMt=1263.18s, ECMt=374.33s**) and more than twice faster for B=0.1n (**PBMt=1158.20s, ECMt=418.94s**). For the rest of the synthetic networks, both models are unable to solve these instances within the specified time limit, thus we compare the average



(a) Budget=0.05n



(b) Budget=0.1n

Figure 2: Variation of average percentage gaps of ECM and PBM models for DCNDP class 1 on some random network.(a) Budget=5% of graph size (b) Budget=10% of graph size. The average gaps are taken over 10 problem instances for each network class.

percentage gaps for both models. The instances are labelled **network-budget**. For instance, **er1-5** represents the **er1** network with 0.05n budget setting. From the graphs in Figure 2, the average %gap for the ECM is larger than those of PBM. In particular, the %gap for ECM is twice that of PBM for **ba2-5 (ECM=3.75%, PBM=1.24%)**; **ba2-10 (ECM=10.32%, PBM=5.35%)** and **er1-10 (ECM=6.44%, PBM=3.41%)**. Moreover, for **ba2-5** and **er1-5** random network classes, we observed that among the 10 instances generated for each class, PBM successfully solved 30% and 50%, respectively, while the success for ECM was 0% and 20%, respectively. Furthermore, percentage gap is observed to increase with the budget for both models. The difference in average % gap for both models is seen to be even more pronounced for larger instances of the uniform random graph model (see Figure 3). This is

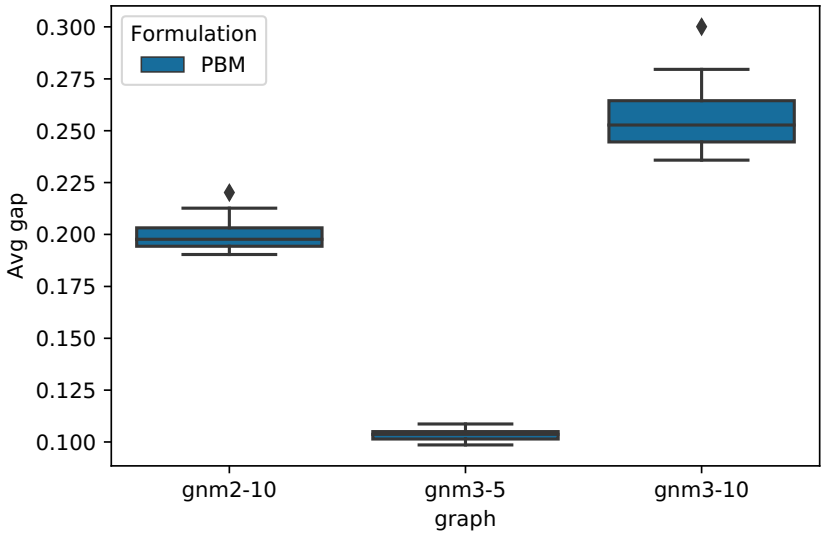


Figure 3: Variation of average gaps of PBM with DCNP-1 distance function on uniform random network instances. ECM gap remains at 100% for all instances of gnm3-5 and gnm3-10. It solves 50% of gnm2-10 graphs to an average percentage gap of 20% and the rest of instances have 100% gap.

understandable as these instances are larger than the rest both in terms of number of nodes and edges. Moreover, the topology of the network is characterised by a

small diameter and a large proportion of nodes being connected by very few hops. Similar to other random network instances, the average percentage gap increases with the budget for both models. For all instances and budget setting, PBM is seen to outperform ECM. PBM in conjunction with the BFS cuts excelled particularly for large edge densed instance (n=300, m=2000) where almost all node pair connections are within short distances. We see the ECM struggling to solve the LP relaxations as can be seen from the average gap of 100%, whereas with our approach, we are still able to achieve competitive bounds and gaps in these instances.

6.2 Comparisons of results for DCNP class 2 models (DCNP-2a & DCNP-2b)

We now present results of computational experiments for the second DCNP class. Recall that for this class, the objective is to minimise the communication efficiency. We use the same set of real-world graphs with budget setting of 5% and 10% of graph size. For randomly generated graphs, we use the same instances of Barabasi-Albert and Erdos-Renyi classes howbeit with budget setting of only 5% of graph size due to the observation of computational difficulty when the budget increases. Results for

					B=0.05n			**B=0.1n**		
Graph	**n**	**m**	**diam**	**InitObj**	**FinObj**	**ECMt (s)**	**PBMt (s)**	**FinObj**	**ECMt (s)**	**PBMt (s)**
Hi-tech	33	91	5	50.8%	43.69%	0.47	**0.45**	32.81%	2.38	**1.53**
Karate	34	78	5	49.2%	33.74%	0.3	**0.19**	16.69%	0.38	**0.28**
Mexican	35	117	4	55.0%	49.06%	**0.35**	0.41	36.58%	0.92	**0.71**
Sawmill	36	62	8	39.9%	27.46%	0.75	**0.44**	14.17%	0.67	**0.34**
Chesapeake	39	170	3	60.4%	53.71%	**0.25**	0.27	35.87%	**0.33**	0.69
Dolphins	62	159	8	37.9%	29.33%	243.13	**23.03**	18.63%	609.81	**24.73**
Lesmiserable	77	254	5	43.5%	18.44%	**3.02**	3.85	7.88%	7.27	**5.02**
Santafe	118	200	12	27.0%	2.95%	14.07	**3.41**	1.39%	21.39	**4.83**
Sanjuansur	75	155	7	34.2%	25.90%	141.1	**25.96**	14.41%	224.22	**28.62**
Attiro	59	128	8	38.9%	31.11%	29.47	**6.17**	22.30%	175.81	**41.93**
SmallWorld	233	994	4	45.4%	9.28%	642.7	**264.88**	4.02%	> 3600	**822.96**
NetScience	379	914	17	20.3%	2.09%	M	**1166**	0.94%	M	**181.55**

Table 4: Results of ECM and PBM models with DCNP-2 on realworld networks

these experiments are summarised in Table 4 for real-world network instances and in Figure 4 for random network instances. Analysing results for the random network

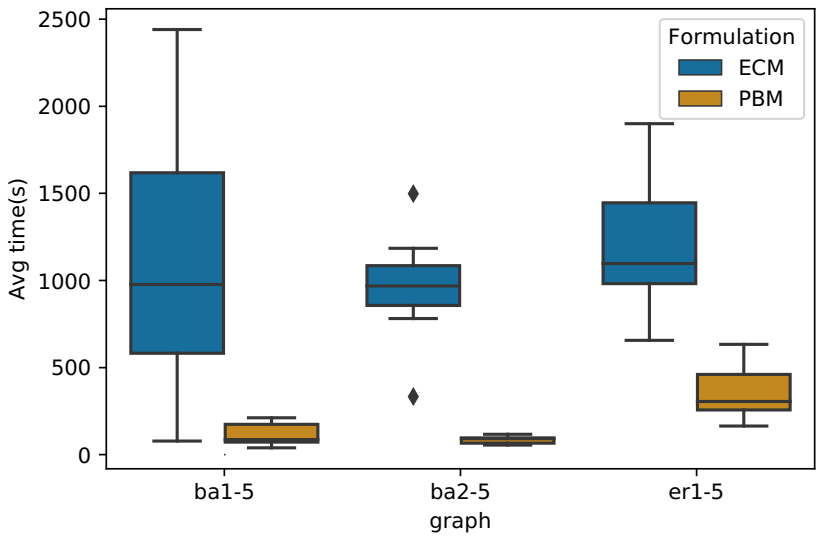


Figure 4: Variation of average computational times of ECM and PBM models with DCNP-2 distance function on Barabasi-Albert and Erdos-Renyi network instances for B=0.05n

instances, it can be seen from Figure 4 that PBM is on average 3 times faster than the ECM for all three sets of instances. Moreover for the larger Erdos-Renyi class **(er2-5)**, a striking observation is that while PBM achieves an average percentage gap of 6.56%, ECM is seen to struggle with solving the linear programming relaxation of these instances within the given time limits. We notice a similar trend for the real-world graphs, PBM competes well with ECM for the smaller instances and indeed performs a lot better in terms of computational times as the graph size increases (see Table 4). For instance, for **Dolphins** network, PBM is more than 10 times faster for both budget settings $B = 0.05n$ and $B = 0.1n$. Similarly, for **Santafe, Sanjuansur & Attiro** networks, PBM is atleast 4 times faster than ECM for both budget settings. Moreover, for the **NetScience graph** which has a large diameter (diameter=17), we see that while PBM is able to solve the problem to optimality in 1166 and 182 seconds respectively for budget settings $B = 0.05n$ and $B = 0.1n$, the ECM runs out of memory. This behaviour might be related to

the issue of sensitivity to large diameter of the base model (ECM) as reported in [29] for a different class of the DCNP (Wiener index). For **USAir97** and **LindenStrasse**, both models fail to close the gap within the time limit. In particular, with **LindenStrasse**, both ECM and PBM struggle to obtain good dual bounds leading to gaps of 108.6% and 102.3%, respectively. Moreover, comparison with the results obtained for the same graphs earlier reported for the first distance class provides strong insights for our discussion in Section 6.4.

6.3 Computational comparison of path-based models DCNP-1b and DCNP-2b version for distance function class 1

As noted earlier in Section 3, the proposed path-based model DCNP-2b is a valid formulation for distance function class 1. By exploiting the structure of distance function 1, we arrived at a formulation which has fewer variables (DCNP-1b). We compare the computational time of both formulations on real-world network instances as summarised in Table 5. The column labelled **PBMt(s)** contains the running time of the model with fewer variables (i.e DCNP-1b) while the column labelled **PBMLt(s)** contains the running time for the model with more variables (i.e DCNP-2b type). As can be seen in Table 5, the model with fewer variables is computationally more competitive which supports the motivation for the reformulation.

6.4 Result comparisons for DCNP-1 and DCNP-2 on some real-world graphs

In the next set of experiments, we present a comparison of results from both classes of distance-based critical node detection problem. Comparing results for the larger real-world graphs, we see that the optimisation solver seems to be having a difficult time solving DCNP-2 in comparison with DCNP-1 (see Tables 3 & 4). In particular, for **SmallWorld** graph with 10% budget, Gurobi is able to solve DCNP-1 in less than 150 seconds whereas it takes over 3600 and 800 seconds to solve DCNP-2 with the base model and path-based models respectively. Similarly, for **USAir97** with

Graph	n	m	diam	InitObj	B	FinObj	PBMt (s)	PBMLt (s)
USAir97	332	2126	6	84.8%	0.1n	5.64%	**1277.38**	2375.76
					0.05n	19.33%	**377.83**	526.39
					0.03n	39.35%	**692.04**	1054.04
					0.02n	49.57%	**582.16**	925.6
					0.01n	63.90%	**240.67**	474.64
SmallWorld	233	994	4	95.2%	0.1n	6.27%	**117.54**	156.78
					0.05n	17.13%	202.87	**170.95**
					0.03n	23.41%	**18.46**	25.71
					0.015n	40.56%	**41.17**	77.65
NetScience	379	914	17	13.3%	0.03n	4.32%	**5.11**	14.76
					0.01n	8.43%	**4.52**	7.8
LindenStrasse	232	303	13	12.1%	0.4n	4.70%	**1.17**	2.03
					0.03n	6.15%	**1.82**	3.32
					0.015n	8.32%	**1.25**	2.11

Table 5: Computational comparisons of DCNP-1b and DCNP-2b type model for distance function class 1 on some real-world network instances

5% budget, Gurobi is unable to solve both the base model and path-based model for DCNP-2. However with DCNP-1, both models are solved under 430 seconds. Even more striking difference is observed for the **Lindenstrasse** and **NetScience** graphs, even though both graphs have very small initial communication efficiency (20%) and k-DistConn (13%). For the random networks, however, the second distance function (2) appears to be less computationally demanding than the first distance function (1). For example, while both models were solved to optimality on just a single random network class **(ba1)** using distance function (1), two additional classes **(ba2-5 & er1-5)** were solved using distance function (2). These differences in computational ease confirm earlier observations that the choice of objective metric is an important part of the critical node detection problem.

6.5 Comparison of results of path-based model with oddhole inequalities and primal heuristics

We conclude our discussion by comparing the path-based model (*PBM*) with the inclusion of the oddhole inequalities and primal heuristics (*PBM+oddhole+heur*). We focused our computational tests on those instances which were unsolved by PBM within the time limit This consists of the random graph instances with $B = 0.1n$ for DCNP-1 and **USAir97** and **LindenStrasse** for DCNP-2 with $B = 0.05n$. Results are reported in Table 6 for network classes where there is a significant difference between *PBM* and *PBM+oddhole+heur*. We omit results for uniform random graphs (**gnm1, gnm2 & gnm3** instances) and Erdos-Renyi graph **er2** where no improvement was observed. From the table, we observe some improvement

				PBM			**PBM+oddhole+heur**		
Instances	**n**	**m**	**density**	**LB**	**UB**	**%gap**	**LB**	**UB**	**%gap**
er1(1)	80	456	14.4%	2400.65	2466	2.72%	**2402.94**	2466	**2.62%**
er1(2)	80	472	14.9%	2398.17	2484	3.45%	**2408.61**	**2481**	**3.01%**
er1(3)	80	474	15.0%	2394.56	**2474**	3.32%	**2404.32**	2483	**2.90%**
er1(4)	80	446	14.1%	2383.29	2474	3.81%	**2393.60**	2474	**3.36%**
er1(5)	80	475	15.0%	2398.21	2483	3.45%	**2413.72**	**2481**	**2.79%**
er1(6)	80	476	15.1%	2394.74	2482	3.60%	**2413.25**	**2481**	**2.81%**
er1(7)	80	474	15.0%	2395.02	2484	3.59%	**2410.33**	**2481**	**2.93%**
er1(8)	80	471	14.9%	2393.31	**2479**	3.58%	**2407.59**	2481	**2.97%**
er1(9)	80	447	14.1%	2378.37	2452	3.10%	**2403.22**	2452	**2.03%**
er1(10)	80	505	16.0%	2398.67	2475	3.14%	**2417.92**	**2474**	**2.32%**
ba2(1)	100	900	18.2%	3674.25	4005	6.58%	**3735**	**4004**	**4.85%**
ba2(2)	100	900	18.2%	3744	**3914**	4.54%	**3753.14**	3916	**4.29%**
ba2(3)	100	900	18.2%	3716	4004	7.29%	**3722.8**	**4003**	**7.10%**
ba2(4)	100	900	18.2%	3751.18	3913	4.31%	**3759.25**	3913	**4.09%**
ba2(5)	100	900	18.2%	3737	**3916**	4.79%	**3741.44**	4003	**4.67%**
ba2(6)	100	900	18.2%	3717.45	3916	5.26%	**3745.66**	**3913**	**4.47%**
ba2(7)	100	900	18.2%	3745.14	3915	4.51%	**3769**	**3914**	**3.85%**
ba2(8)	100	900	18.2%	3767.83	3915	3.85%	**3790.42**	**3913**	**3.23%**
ba2(9)	100	900	18.2%	3702.2	4005	5.77%	**3755.38**	**3916**	**4.28%**
ba2(10)	100	900	18.2%	3744.49	**3912**	4.47%	**3744.76**	3914	4.47%
USAir97	332	2126	3.9%	5703.83	10745.58	88.39%	**5758.62**	10745.58	**86.60%**
LindenStrasse	232	303	1.1%	**1162.01**	2350.18	**102.25%**	1150.86	2350.18	104.21%

Table 6: Results of PBM and PBM+oddhole+heur

of version *PBM+oddhole+heur* with respect to *PBM* over the denser Barabasi and

Erdos-Renyi classes (**Ba2** & **er1**) instances, where tighter bounds are realised with the inclusion of the odd hole inequalities. However, across all of the uniform random graphs (**gnm1, gnm2 & gnm3**) instances as well as over Erdos-Renyi **er2** instances, *PBM* and *PBM+oddhole+heur* behave in a similar way. A possible explanation for this is the very few number of cycles reported in these graphs which could have reduced the chances of having violated inequalities. With respect to the cycle enumeration, we observed that about four times as many cycles were enumerated for the Barabasi graph class **Ba2** and twice as many cycles for Erdos-Renyi **er1** thus leading to more usercuts violations. A clear evidence of this structural difference is in the edge densities of these graph classes. While **er1** and **Ba2** instances have average densities of 14.9% and 18.2% respectively, the **gnm1, gnm2, gnm3** and **er2** instances have smaller average densities of 5.0%, 3.3%, 4.5% and 5% respectively. For the DCNP-2 instances, *PBM+oddhole+heur* improved in lower bounds for the **USAir97** graph instance but fell behind of its counterpart *PBM* for **LindenStrasse** graph.

With regards to the primal bounds, better upper bounds were recorded for some of the **Ba2** and **er1** instances. Although this improvement is mild, it is indicative of the potential of the primal heuristics.

7 Conclusion

In this paper, we considered two classes of distance-based critical node detection problem namely, minimisation of node pairs connected within specific threshold length and communication efficiency. We presented new integer programming formulations, separation heuristics and valid inequalities that exploit the structure of the problems to strengthen the formulations. Extensive computational experiments on both real-world and synthetic graphs shed light on the scalability of our approach. The effectiveness of our approach is more evident as the graph size grows, when the existing compact model struggles to solve even the linear programming relaxation. Results also provides insights into the influence of graph topology, objective metric and budget constraint on the identification of critical nodes in a network. This emphasises the need for

appropriate choice of objective metric for specific application setting in order to design efficient solution methods. In the future, it would be interesting to see extensions of our approach to other distance functions as well as computational comparisons of the different distance functions on more graphs. Since the distance-based critical node detection problem is $\mathcal{NP}$-hard, development of efficient heuristics to handle larger instances is also a fruitful future research direction.

References

[1] Addis, B., Aringhieri, R., Grosso, A. and Hosteins, P. [2016], 'Hybrid constructive heuristics for the critical node problem', *Annals of Operations Research* **238**(1-2), 637–649.

[2] Addis, B., Di Summa, M. and Grosso, A. [2013], 'Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth', *Discrete Applied Mathematics* **161**(16-17), 2349–2360.

[3] Aringhieri, R., Grosso, A., Hosteins, P. and Scatamacchia, R. [2016*a*], 'A general evolutionary framework for different classes of critical node problems', *Engineering Applications of Artificial Intelligence* **55**, 128–145.

[4] Aringhieri, R., Grosso, A., Hosteins, P. and Scatamacchia, R. [2016*b*], 'Local search metaheuristics for the critical node problem', *Networks* **67**(3), 209–221.

[5] Aringhieri, R., Grosso, A., Hosteins, P. and Scatamacchia, R. [2019], 'Polynomial and pseudo-polynomial time algorithms for different classes of the distance critical node problem', *Discrete Applied Mathematics* **253**, 103–121.

[6] Arulselvan, A., Commander, C. W., Elefteriadou, L. and Pardalos, P. M. [2009], 'Detecting critical nodes in sparse graphs', *Computers & Operations Research* **36**(7), 2193–2200.

[7] Arulselvan, A., Commander, C. W., Pardalos, P. M. and Shylo, O. [2007],

'Managing network risk via critical node identification', *Risk management in telecommunication networks, Springer* pp. 79–92.

[8] Arulselvan, A., Commander, C. W., Shylo, O. and Pardalos, P. M. [2011], Cardinality-constrained critical node detection problem, *in* 'Performance models and risk management in communications systems', Springer, pp. 79–91.

[9] Baird, D. and Ulanowicz, R. E. [1989], 'The seasonal dynamics of the chesapeake bay ecosystem', *Ecological monographs* **59**(4), 329–364.

[10] Barabási, A.-L. and Albert, R. [1999], 'Emergence of scaling in random networks', *Science* **286**(5439), 509–512.

[11] Batagelj, V. and Mrvar, A. [2006], 'Pajek datasets', `http://vlado.fmf.uni-lj.si/pub/networks/data/`. Last accessed: 02-11-2018.

[12] Boginski, V. and Commander, C. W. [2009], Identifying critical nodes in protein-protein interaction networks, *in* 'Clustering challenges in biological networks', World Scientific, pp. 153–167.

[13] Borgatti, S. P. [2006], 'Identifying sets of key players in a social network', *Computational & Mathematical Organization Theory* **12**(1), 21–34.

[14] Borgatti, S. P. and Everett, M. G. [2006], 'A graph-theoretic perspective on centrality', *Social networks* **28**(4), 466–484.

[15] Botton, Q., Fortz, B., Gouveia, L. and Poss, M. [2013], 'Benders decomposition for the hop-constrained survivable network design problem', *INFORMS journal on computing* **25**(1), 13–26.

[16] Crucitti, P., Latora, V., Marchiori, M. and Rapisarda, A. [2003], 'Efficiency of scale-free networks: error and attack tolerance', *Physica A: Statistical Mechanics and its Applications* **320**, 622–642.

[17] Di Summa, M., Grosso, A. and Locatelli, M. [2011], 'Complexity of the critical node problem over trees', *Computers & Operations Research* **38**(12), 1766–1774.

[18] Di Summa, M., Grosso, A. and Locatelli, M. [2012], 'Branch and cut algorithms for detecting critical nodes in undirected graphs', *Computational Optimization and Applications* **53**(3), 649–680.

[19] Dinh, T. N., Xuan, Y., Thai, M. T., Pardalos, P. M. and Znati, T. [2012], 'On new approaches of assessing network vulnerability: hardness and approximation', *IEEE/ACM Transactions on Networking (ToN)* **20**(2), 609–619.

[20] Erdös, P., Rényi, A. et al. [1959], 'On random graphs', *Publicationes mathematicae* **6**(26), 290–297.

[21] Girvan, M. and Newman, M. E. [2002], 'Community structure in social and biological networks', *Proceedings of the national academy of sciences* **99**(12), 7821–7826.

[22] Gouveia, L., Joyce-Moniz, M. and Leitner, M. [2018], 'Branch-and-cut methods for the network design problem with vulnerability constraints', *Computers & Operations Research* **91**, 190–208.

[23] Gouveia, L. and Leitner, M. [2017], 'Design of survivable networks with vulnerability constraints', *European Journal of Operational Research* **258**(1), 89–103.

[24] Gouveia, L., Patrício, P. and de Sousa, A. [2006], Compact models for hop-constrained node survivable network design: An application to mpls, *in* 'Telecommunications planning: Innovations in pricing, network design and management', Springer, pp. 167–180.

[25] Grötschel, M., Monma, C. L. and Stoer, M. [1995], 'Design of survivable networks', *Handbooks in operations research and management science* **7**, 617–672.

[26] Guérin, R. and Orda, A. [2002], 'Computing shortest paths for any number of hops', *IEEE/ACM transactions on networking* **10**(5), 613–620.

[27] Gurobi Optimization, I. [2016], 'Gurobi optimizer reference manual'.
URL: *http://www.gurobi.com*

[28] Hagberg, A., Swart, P. and S Chult, D. [2008], Exploring network structure, dynamics, and function using networkx, Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[29] Hooshmand, F., Mirarabrazi, F. and MirHassani, S. [2020], 'Efficient benders decomposition for distance-based critical node detection problem', *Omega* **93**.

[30] Knuth, D. E. [1993], *The Stanford GraphBase: a platform for combinatorial computing*, AcM Press New York.

[31] Liu, H. and Wang, J. [2006], A new way to enumerate cycles in graph, *in* 'Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)', IEEE, pp. 57–57.

[32] Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E. and Dawson, S. M. [2003], 'The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations', *Behavioral Ecology and Sociobiology* **54**(4), 396–405.

[33] Mahjoub, A. R., Simonetti, L. and Uchoa, E. [2013], 'Hop-level flow formulation for the survivable network design with hop constraints problem', *Networks* **61**(2), 171–179.

[34] Matisziw, T. C. and Murray, A. T. [2009], 'Modeling s–t path availability to support disaster vulnerability assessment of network infrastructure', *Computers & Operations Research* **36**(1), 16–26.

[35] Nandi, A. K. and Medal, H. R. [2016], 'Methods for removing links in a network to minimize the spread of infections', *Computers & Operations Research* **69**, 10–24.

[36] Paton, M., Akartunalı, K. and Higham, D. J. [2017], 'Centrality analysis for modified lattices', *SIAM Journal on Matrix Analysis and Applications* **38**(3), 1055–1073.

[37] Salemi, H. and Buchanan, A. [2020], 'Solving the distance-based critical node problem', Optimization Online. `http://www.optimization-online.org/DB_HTML/2020/04/7751.html`.

[38] Schieber, T. A., Carpi, L., Frery, A. C., Rosso, O. A., Pardalos, P. M. and Ravetti, M. G. [2016], 'Information theory perspective on network robustness', *Physics Letters A* **380**(3), 359–364.

[39] Shen, S. and Smith, J. C. [2012], 'Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs', *Networks* **60**(2), 103–119.

[40] Shen, S., Smith, J. C. and Goli, R. [2012], 'Exact interdiction models and algorithms for disconnecting networks via node deletions', *Discrete Optimization* **9**(3), 172–188.

[41] Shen, Y., Nguyen, N. P., Xuan, Y. and Thai, M. T. [2013], 'On the discovery of critical links and nodes for assessing network vulnerability', *IEEE/ACM Transactions on Networking (TON)* **21**(3), 963–973.

[42] *UCINET software datasets* [n.d], `https://sites.google.com/site/ucinetsoftware/datasets/`. Last accessed: 06-11-2018.

[43] Veremyev, A., Boginski, V. and Pasiliao, E. L. [2014], 'Exact identification of critical nodes in sparse networks via new compact formulations', *Optimization Letters* **8**(4), 1245–1259.

[44] Veremyev, A., Prokopyev, O. A. and Pasiliao, E. L. [2014], 'An integer programming framework for critical elements detection in graphs', *Journal of Combinatorial Optimization* **28**(1), 233–273.

[45] Veremyev, A., Prokopyev, O. A. and Pasiliao, E. L. [2015], 'Critical nodes for distance-based connectivity and related problems in graphs', *Networks* **66**(3), 170–195.

[46] Zachary, W. W. [1977], 'An information flow model for conflict and fission in small groups', *Journal of anthropological research* **33**(4), 452–473.