Accepted Manuscript

Improved Formulations for Minimum Connectivity Interdiction Problems

Konstantin Pavlikov

PII: S0305-0548(18)30103-5  
DOI: 10.1016/j.cor.2018.04.012  
Reference: CAOR 4453

To appear in: Computers and Operations Research

Received date: 26 September 2017  
Revised date: 21 March 2018  
Accepted date: 17 April 2018

Please cite this article as: Konstantin Pavlikov, Improved Formulations for Minimum Connectivity Interdiction Problems, Computers and Operations Research (2018), doi: 10.1016/j.cor.2018.04.012

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
Highlights

- A simple example appeared demonstrating the comparison of linear relaxations of proposed models.
- The example motivates the following important result: the equivalence of two problem formulations from the linear relaxation point of view.
- As noted by some reviewers, the proof of main result contained some imprecision that has been fixed in this version of the paper.
- The process of filtering out redundant constraints was analyzed from computational point of view: reported CPU times to obtain it for several problem instances.
- The paper was improved in various other places in terms of language and precision.
Improved Formulations for Minimum Connectivity Interdiction Problems

Konstantin Pavlikov

Department of Business and Economics
University of Southern Denmark
Campusvej 55, 5230 Odense M, Denmark

E-mail: kop@sam.sdu.dk

First draft: June 2017, This draft: April 18, 2018

Abstract

The minimum connectivity interdiction problem seeks to remove at most \( k \) nodes from an undirected graph, such that a connectivity measure of the remaining subgraph is minimized. Examples of connectivity measures of a graph include the number of connected pairs of nodes, the size of the largest connected component, and the inverse of the number of connected components. This paper proposes a few improvements to mixed integer linear programming formulations known in the literature. Improvements correspond to the construction of formulations with either a smaller number of constraints, or tighter formulations with increased sparsity of constraint matrices. In addition, two of discussed problem formulations are demonstrated to be equivalent by providing the same value of linear relaxation. Computational experiments demonstrate that improved formulations allow us to solve some of the considered problem instances up to an order of magnitude times faster than using the recent benchmark.

Keywords: network connectivity, network interdiction, mixed integer linear programming

1. Introduction

In this paper we present several improvements for a class of minimum connectivity interdiction problems, also known as critical nodes detection problems. Let \( G = (V, E) \) be a connected undirected graph with \( |V| = n \) nodes and \( |E| = m \) edges. The critical nodes detection problem aims to find a subset of nodes \( B \) of specified cardinality that minimizes a connectivity measure of the remaining graph (Borgatti, 2006). The most common connectivity measures of a graph include:

- the number of connected pairs of nodes, denoted by \( F_1 \),
- the size of the largest connected component, denoted by \( F_2 \),
- the inverse of the number of connected components, denoted by \( F_3^{-1} \).

The range of application areas for such problems may potentially include homeland security (Salmeron et al., 2004), social network analysis (Borgatti, 2006), network reliability (Sun and Shayman, 2007; Matisziw et al., 2009), and other. Critical nodes detection problems with respect to the above three connectivity measures have been shown to be NP-hard on general graphs (Arulselvan et al., 2009; Shen et al., 2012), and are polynomially solvable on trees (Di Summa et al., 2011; Shen and Smith, 2012). In this study we focus on exact solution methods to such problems on general graphs that involve mixed integer linear programming algorithms. Moreover, efficient optimization approaches for critical node detection problems with respect to the first objective is fundamental for handling other connectivity measures as well (this will
become clear later). Hence, in this paper, we mainly focus on developing improvements for the critical nodes problem with respect to the first connectivity measure, and employ them further to demonstrate efficiency on problems with other connectivity measures. The problem can be formalized as follows: find a subset of nodes $B \subset V$ with cardinality at most $k$, such that the number of connected node pairs in $G(V \setminus B)$ is minimized.

Let $x_i = 1$ if and only if node $i$ belongs to the set $B$. Introduce a set of connectivity variables between pairs of nodes $i, j \in V$, $i \neq j$:

$$u_{ij} = \begin{cases} 
1, & \text{if } i, j \in V \setminus B, i \neq j \text{ and } i \text{ and } j \text{ are connected by a path in } G(V \setminus B), \\
0, & \text{otherwise}.
\end{cases}$$

(1.1)

In other words, $u_{ij}$ represents the value of function $\text{IS\_PATH}(G, i, j)$ that returns 1 if and only if a graph $G$ contains a path between nodes $i$ and $j$. Using this notation, the value of connectivity measure $F_1$ of the graph $G(V \setminus B)$ can be expressed as follows:

$$F_1(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i, j \in V} u_{ij}.$$  

(1.2)

Probably, the most straightforward way to formulate the minimum pairwise connectivity problem is to employ a set coverage approach, presented for instance in Di Summa et al. (2011), where it is observed that in order to disconnect $i$ and $j$, one needs to interdict every path that connects them. The optimization problem formulation then takes the following form:

$$\min_{x_i} F_1 : \min_{x_i, u_{ij}} \frac{1}{2} \sum_{i, j \in V, i \neq j} u_{ij}$$

subject to

$$u_{ij} \geq 1 - \sum_{t \in V(P_{ij})} x_t, \quad i, j \in V, i \neq j, \quad \forall P_{ij} \in P_{ij},$$

(1.4)

$$\sum_{i \in V} x_i \leq k,$$

(1.5)

$$u_{ij} \in \{0, 1\}, \quad i, j \in V, i \neq j,$$

(1.6)

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

(1.7)

where $V(P_{ij})$ is the set of nodes in a path $P_{ij}$ from node $i$ to node $j$, and $P_{ij}$ is the set of all possible paths from node $i$ to node $j$ in the network. Clearly, if $(i, j) \in E$, then the set of constraint (1.4) can be reduced to the one path through the link connecting them:

$$u_{ij} \geq 1 - x_i - x_j, \quad (i, j) \in E.$$  

(1.8)

Indeed, for any other path not through the arc, the corresponding constraint takes the following form:

$$u_{ij} \geq 1 - x_i - x_j - \sum_{t \in V(P_{ij}) \setminus \{i, j\}} x_t, \quad (i, j) \in E,$$

where $t \in V(P_{ij}) \setminus \{i, j\}$ denotes the set of other nodes in the path $P_{ij}$ besides $i$ and $j$. However, such a constraint becomes redundant if (1.8) is already a part of the problem formulation. Consequently, for the same argument, it is only required to consider simple paths between nodes, i.e., paths that contain no cycles. While there exist algorithms for generation of all simple paths of a graph (Rubin, 1978; Hershberger et al., 2007), the number of possible paths and hence constraints is exponential on general graphs and makes it prohibitive for usage in optimization.
Nevertheless, formulation (1.3) – (1.7) is conceptually important one and possesses the following property. Consider constraint (1.4): its right hand side takes values \(\{1, 0, -1, \ldots, -n + 1\}\) for any combination of binary variables \(x_i\), therefore, taking into account the objective of minimizing the sum of nonnegative variables \(u_{ij}\), it is evident that at optimality any \(u_{ij}\) will be either 0 or 1. Hence, the binary requirement (1.6) for variables \(u_{ij}\) can be relaxed to \(u_{ij} \geq 0\).

Due to the size limitation of the above formulation, other approaches to formulate the problem have been proposed. First of all, note that the properly defined set of \(u_{ij}\) variables should necessarily satisfy the following condition

\[
\sum_{i \in V} x_i \leq k,
\]

and hence only the ordered pairs \((i, j)\), \(i < j\) need to be considered. Let \(S\) be the set of ordered node pairs of interest:

\[
S = \{(i, j) \mid i, j \in V, i < j\}.
\]

The following notation is convenient to introduce:

\[
u_{ij}^* = \begin{cases} u_{ij}, & \text{if } i < j, \\ u_{ji}, & \text{if } i > j, \end{cases} \quad i, j \in V, i \neq j. \tag{1.10}
\]

Hence, when we refer to connectivity of nodes \(i\) and \(j\), and do not know in advance which index is smaller, we will refer to \(u_{ij}^*\).

The following minimum pairwise connectivity problem formulation (Arulselvan et al., 2009) is based on the triangular constraints introduced in Oosten et al. (2007):

\[
\min_{x_i, u_{ij}} \sum_{(i, j) \in S} u_{ij} \tag{1.11}
\]

subject to

\[
\begin{align*}
  u_{ij} & \geq 1 - x_i - x_j, & \quad (i, j) \in S, (i, j) \in E, \tag{1.12} \\
  u_{ij} & \geq u_{ik}^* + u_{kj}^* - 1, & \quad (i, j) \in S, (i, j) \notin E, k \in V \setminus \{i, j\}, \tag{1.13} \\
  \sum_{i \in V} x_i & \leq k, & \quad (i, j) \in S, (i, j) \notin E, \tag{1.14} \\
  u_{ij} & \geq 0, & \quad (i, j) \in S, (i, j) \in E, \tag{1.15} \\
  u_{ij} & \in \{0, 1\}, & \quad (i, j) \in S, (i, j) \notin E, \tag{1.16} \\
  x_i & \in \{0, 1\}, & \quad i \in V. \tag{1.17}
\end{align*}
\]

This problem formulation is based on the idea that in order for \(i\) and \(j\) to be connected, there should be a node \(k \neq i, j\) such that both pairs \(i\) and \(k\), \(k\) and \(j\) are connected by corresponding paths. Note that the above problem formulation only requires a polynomial number of constraints of the order \(O(n^3)\). While this is a considerable advantage compared to the original problem formulation, the number of constraints can make a problem intractable for graphs with the size of a few hundred nodes.

The next section briefly reviews improved problem formulations that have appeared in the literature. Section 3 describes further enhancements and establishes a result connecting the quality of linear relaxations of considered problem formulations. Section 4 provides problem formulations for other connectivity measures mentioned earlier. Section 5 provides a computational study. Section 6 concludes.

2. Smaller Sized Problem Formulations

In this section, we will discuss several improvements to formulation of the minimum pairwise connectivity interdiction problem that have appeared in the literature. Dinh and Thai (2011)
argue that the number of triangular inequalities (1.13) can be reduced. Instead of considering constraint
\[ u_{ij} \geq u^*_{ik} + u^*_{kj} - 1, \quad (i, j) \in S, (i, j) \notin E, k \in V \setminus \{i, j\}, \]
for every other node \(k\) in the network, note that in order for nodes \(i\) and \(j\) to be connected, there should exist at least one node in the neighborhood of \(i\) that is connected to \(j\). Therefore, the following reduced number of constraints is sufficient:
\[ u_{ij} \geq u^*_{ik} + u^*_{kj} - 1, \quad (i, j) \in S, (i, j) \notin E, k \in N(i), \]
where \(N(i)\) denotes the set of neighbors of node \(i\). Note that since \(k\) and \(i\) are connected by an arc, \(i\) and \(j\) are connected if and only if \(i, j\) and \(k\) are not in \(B\) and \(u^*_{kj} = 1\). The condition \(u^*_{kj} = 1\) implies that \(j, k \notin B\) by definition, therefore the above triangular constraint becomes equivalent to:
\[ u_{ij} \geq u^*_{kj} - x_i, \quad (i, j) \in S, (i, j) \notin E, k \in N(i). \tag{2.1} \]
The same logic applies to the neighborhood of \(j\): in order for \(i\) and \(j\) to be connected, there should exist a node in the neighborhood of \(j\) such that it is connected to \(i\), which results in an alternative set of inequalities:
\[ u_{ij} \geq u^*_{ik} - x_j, \quad (i, j) \in S, (i, j) \notin E, k \in N(j). \]
Since both sets of constraints are equivalent, we can employ only one of them that contains a smaller number of constraints. Let \(d(i)\) denote the size of the neighborhood of node \(i \in V\), \(d(i) = |N(i)|\), then, a more efficient way to represent (1.13) is as follows (Dinh and Thai, 2011):
\[
\begin{cases}
  u_{ij} \geq u^*_{kj} - x_i, & k \in N(i), \\
  u_{ij} \geq u^*_{ik} - x_j, & k \in N(j),
\end{cases} \quad \text{if } d(i) \leq d(j),
\begin{cases}
  u_{ij} \geq u^*_{kj} - x_i, & k \in N(i), \\
  u_{ij} \geq u^*_{ik} - x_j, & k \in N(j),
\end{cases} \quad \text{if } d(i) > d(j). \tag{2.2}
\]
Dinh and Thai (2011) estimate that this approach requires only \(O(nm)\) constraints instead of \(O(n^3)\) for the approach of Arulselvan et al. (2009). Moreover, the authors argue that binary requirement for \(u_{ij}\) in the above formulation can be safely relaxed to the continuous one:
\[ u_{ij} \geq 0, \quad (i, j) \in S. \]

Another optimization approach was proposed in Veremyev et al. (2014a). Constraint (2.1) implies that non-adjacent nodes \(i\) and \(j\) are connected once there is at least one \(k\) in the neighborhood of \(i\) such that \(u^*_{kj} = 1\), and \(x_i = 0\). This requirement can be alternatively modeled by a binary variable \(u_{ij}\) constrained as follows:
\[
\textbf{B : } u_{ij} \geq 1 - x_i - x_j, \quad (i, j) \in S, (i, j) \notin E, \tag{2.3}
\]
\[ u_{ij} \geq \frac{1}{d(i)} \sum_{k \in N(i)} u^*_{kj} - x_i, \quad (i, j) \in S, (i, j) \notin E, \tag{2.4} \]
\[ u_{ij} \geq 0, \quad (i, j) \in S, (i, j) \notin E, \tag{2.5} \]
\[ u_{ij} \in \{0, 1\}, \quad (i, j) \in S, (i, j) \notin E. \tag{2.6} \]
using the approach, similar in spirit to the one in Watters (1967) for linearization of polynomials. The main advantage of the proposed approach is the reduction of the number of required constraints to \(O(n^2)\). However, taking into account improvement (2.2), this idea can be more
efficiently presented in the following way:

\[
 u_{ij} \geq \begin{cases} 
 \frac{1}{d(i)} \sum_{k \in \mathcal{N}(i)} u_{kj} - x_i, & \text{if } d(i) \leq d(j), \\
 \frac{1}{d(j)} \sum_{k \in \mathcal{N}(j)} u_{ik} - x_j, & \text{if } d(i) > d(j),
\end{cases} \quad (i, j) \in S, (i, j) \notin E, \quad (2.7)
\]

\[
 u_{ij} \in \{0, 1\}, \quad (i, j) \in S, (i, j) \notin E. \quad (2.8)
\]

The advantage of this improvement is that the minimum of \(d(i)\) and \(d(j)\) is employed as the scaling coefficient, which is likely to improve the tightness of the linear relaxation of the model at no additional cost (at least theoretically) of solving this relaxation. This typically leads to a significant improvement of performance of a solver.

In addition, Veremyev et al. (2014a) proved the following useful property that helps to reduce further the number of binary variables to introduce. The authors demonstrate that there exists an optimal solution to (1.11) – (1.17) problem, such that \(x_i = 0\) for any \(i \in D_1\), where \(D_1 = \{i \in V \mid d(i) = 1\}\) denotes the set of vertices of degree 1 (the same property holds for critical node detection problems with respect to connectivity measures \(F_2\) and \(F_3\), Veremyev et al. (2014b)). Note that strictly speaking this result only applies to nontrivial network instances with \(n \geq 3\). With that result we introduce the following set of notations:

\[
\tilde{S} = \{(i, j) \mid i, j \in V \setminus D_1, i < j\} = \text{the set of node pairs of interest}, \quad (2.9)
\]

\[
\tilde{N}(i) = N(i) \setminus D_1 = \text{the set of neighbors of node } i \text{ with degree greater than 1}, \quad (2.10)
\]

\[
\tilde{d}(i) = \lvert \tilde{N}(i) \rvert, \quad (2.11)
\]

\[
w(i) = \lvert D_1 \cap N(i) \rvert. \quad (2.12)
\]

With that, we obtain two improved formulations of the minimum pairwise connectivity problem. One of them is an improved version of the formulation in Veremyev et al. (2014a):

\[
 u_{ij} \geq \begin{cases} 
 \frac{1}{d(i)} \sum_{k \in \tilde{N}(i)} u_{kj} - x_i, & \text{if } \tilde{d}(i) \leq \tilde{d}(j), \\
 \frac{1}{d(j)} \sum_{k \in \tilde{N}(j)} u_{ik} - x_j, & \text{if } \tilde{d}(i) > \tilde{d}(j),
\end{cases} \quad (i, j) \in \tilde{S}, (i, j) \notin E, \quad (2.13)
\]

\[
 u_{ij} \in \{0, 1\}, \quad (i, j) \in \tilde{S}, (i, j) \notin E. \quad (2.14)
\]

and the other one is an improved version of the formulation in Dinh and Thai (2011):

\[
\begin{cases} 
 u_{ij} \geq u_{kj} - x_i, & k \in \tilde{N}(i), \quad \text{if } \tilde{d}(i) \leq \tilde{d}(j), \\
 u_{ij} \geq u_{ik} - x_j, & k \in \tilde{N}(j), \quad \text{if } \tilde{d}(i) > \tilde{d}(j),
\end{cases} \quad (i, j) \in \tilde{S}, (i, j) \notin E, \quad (2.15)
\]

\[
 u_{ij} \geq 0, \quad (i, j) \in \tilde{S}, (i, j) \notin E. \quad (2.16)
\]

In addition, note that objective function (1.2) simplifies to consideration of node pairs where every node has a degree greater than one (see Veremyev et al. (2014a) for more details), and can be expressed through the following minimization problem:

\[
 F_1(x_1, \ldots, x_n) = \min_{u_{ij}} \sum_{(i, j) \in \tilde{S}} u_{ij} = \sum_{(i, j) \in \tilde{S}} (1 + w(i))(1 + w(j)) u_{ij} + \frac{1}{2} \sum_{i \in N(D_1)} w(i) (w(i) + 1) (1 - x_i) \quad (2.17)
\]
subject to
\[ u_{ij} \geq 1 - x_i - x_j, \quad (i, j) \in \tilde{S}, \quad (i, j) \in E, \quad (2.18) \]
\[ u_{ij} \geq 0, \quad (i, j) \in \tilde{S}. \quad (2.19) \]
where \( N(D_1) \) denotes the set of neighbors of nodes with degree 1. In the next section, we will describe several additional considerations to improve the above two problem formulations by further reducing the size of set \( \tilde{N}(i) \) via filtering out neighbors through which the connectivity can not be established.

3. Contributions of the Current Paper

In this section we argue that not every node \( k \) in the neighborhood of \( i \) or \( j \) is meaningful for presence in constraint (2.2) or in (2.7). Suppose we have a pair of nodes \((i, j)\) and a node \( k \) is the neighborhood of \( i \), \( k \in \tilde{N}(i) \). Consider the following problem: find a path between \( k \) and \( j \) that does not include \( i \). If such a path does not exist, that means a possible connection from \( k \) to \( j \) should necessarily pass through \( i \), which is why the constraint \( u_{ij} \geq u_{*kj} - x_i \) is redundant. See an illustration of this idea at Figure 1.

Consider again the example in Figure 1. Node 6 belongs to the neighborhood of node 4, however, the only possible path from 6 to 3, not going through 4, includes node 5, which is another neighbor of 4. Therefore, including node 6 in a set of neighbors of 4 that potentially can establish connectivity of 3 and 4 is not needed, as long as that set contains node 5. Denote by \( G(i, k) \) the subgraph of \( G \) that does not include node \( i \), neighbors of \( i \) except to the neighbor \( k \), and their corresponding edges:

\[ G(i, k) = G(V \setminus \{i\} \cup \tilde{N}(i) \setminus \{k\}) \], \quad i \in V \setminus D_1, \ k \in \tilde{N}(i). \quad (3.1) \]

Define

\[ N(i, j) = \{k \in \tilde{N}(i) \mid \mathbf{IS\_PATH}(G(i, k), k, j) = 1\} \quad (3.2) \]

to be the “essential” neighborhood of \( i \) with respect to \( j \), and let \( d(i, j) = |N(i, j)| \). Then, the improved version of the Veremyev et al. (2014a) approach to constrain connectivity variables

\[ u_{ij} \geq 1 - x_i - x_j, \quad (i, j) \in \tilde{S}, \quad (i, j) \in E, \quad (2.18) \]
\[ u_{ij} \geq 0, \quad (i, j) \in \tilde{S}. \quad (2.19) \]
$u_{ij}$ is presented below:

**P1**: $u_{ij} \geq 1 - x_i - x_j$, \hspace{1cm} $(i, j) \in \tilde{S}, (i, j) \in E$, \hspace{1cm} (3.3)

$$u_{ij} \geq \begin{cases} \frac{1}{d(i, j)} \sum_{k \in N(i, j)} u_{ik}^* - x_i, & \text{if } d(i, j) \leq d(j, i), \\ \frac{1}{d(j, i)} \sum_{k \in N(j, i)} u_{jk}^* - x_j, & \text{if } d(i, j) > d(j, i), \end{cases} \hspace{1cm} \hspace{1cm} (i, j) \in \tilde{S}, (i, j) \notin E, \hspace{1cm} (3.4)$$

$u_{ij} \geq 0$, \hspace{1cm} $(i, j) \in \tilde{S}, (i, j) \in E$, \hspace{1cm} (3.5)

$u_{ij} \in \{0, 1\}$, \hspace{1cm} $(i, j) \in \tilde{S}, (i, j) \notin E$. \hspace{1cm} (3.6)

Likewise, the improved triangular-based approach can now be expressed as follows:

**P2**: $u_{ij} \geq 1 - x_i - x_j$, \hspace{1cm} $(i, j) \in \tilde{S}, (i, j) \in E$, \hspace{1cm} (3.7)

$$u_{ij} \geq \begin{cases} u_{kj}^* - x_i, & k \in N(i, j), \hspace{1cm} \text{if } d(i, j) \leq d(j, i), \\ u_{ik}^* - x_j, & k \in N(j, i), \hspace{1cm} \text{if } d(i, j) > d(j, i), \end{cases} \hspace{1cm} (i, j) \in \tilde{S}, (i, j) \notin E, \hspace{1cm} (3.8)$$

$u_{ij} \geq 0$, \hspace{1cm} $(i, j) \in \tilde{S}$. \hspace{1cm} (3.9)

Finally in this section, we will discuss the quality of linear programming relaxations of the proposed mixed integer linear problem formulations. Intuitively speaking, formulation (1.3) – (1.7) should provide a stronger linear relaxation compared to other considered problem formulations given an exponential number of constraints it requires. Consider the linear programming relaxations of the $\text{min } F_1$ problem using the network instance presented on Figure 2. The path-based formulation (1.3) – (1.7) is easy to construct in this case, since every nonadjacent pair of nodes has only two paths that may help to establish connectivity. For example, node 2 can be connected to node 4 either via the $2 - 3 - 4$ path or via the $2 - 1 - 5 - 4$ path.

![Figure 2: An undirected network with 5 nodes.](image)

Table 1 presents values of linear programming relaxations of the $\text{min } F_1$ problem using the network instance from Figure 2.

<table>
<thead>
<tr>
<th></th>
<th>Relaxed (1.3) – (1.7)</th>
<th>Relaxed P2</th>
<th>Relaxed P1</th>
<th>Relaxed B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.0</td>
<td>5.0</td>
<td>4.25</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Table 1: Linear programming relaxations of different formulations of the $\text{min } F_1(x_1, \ldots, x_n)$ problem with $k = 1$.

By construction, formulation **P1** is not weaker than **B**; it is also clear that formulation **P2** is at least as strong as **P1**. Observe that formulation **P2** provides the same linear relaxation as the path-based formulation (1.3) – (1.7). We prove next that this observation appears not by coincidence and holds true in general, i.e., that the quality of linear programming relaxation of the formulation **P2** is exactly the same as that of the original set cover-type formulation (1.3) – (1.7) for any undirected graph.
Proposition 3.1. The original minimum pairwise connectivity problem formulation (1.3) – (1.7) with an exponential number of constraints provides the same linear programming relaxation value as the improved triangular formulation (2.17), (3.7) – (3.9) with $O(nm)$ constraints.

Proof. Let the feasible space $X$ be defined by

$$ X = \left\{ x \in \mathbb{R}^n \mid x_i \in [0, 1] \; \forall i \in V \setminus D_1, \; \sum_{i \in V \setminus D_1} x_i \leq k, \; x_i = 0 \; \forall i \in D_1 \right\} $$

and function $f(x, u) = \sum_{(i, j) \in \tilde{S}} (1 + w(i))(1 + w(j)) u_{ij} + \frac{1}{2} \sum_{i \in N(D_1)} w(i)(w(i) + 1)(1 - x_i)$ denote the objective function. Then, the problem (1.3) – (1.7) can be expressed as

$$ \min_{x \in X, u \in U_1(x)} f(x, u), \quad (3.10) $$

where $U_1(x)$ is defined by the following set of constraints:

$$ u_{ij} \geq 1 - x_i - x_j, \quad (i, j) \in \tilde{S}, \; (i, j) \in E, \quad (3.11) $$

$$ u_{ij} \geq 1 - \sum_{t \in V(\mathcal{P}_{ij})} x_t, \quad (i, j) \in \tilde{S}, \; (i, j) \in E, \; \forall \mathcal{P}_{ij} \in \mathcal{P}_{ij}, \quad (3.12) $$

$$ u_{ij} \geq 0, \quad (i, j) \in \tilde{S}. \quad (3.13) $$

At the same time, the problem (2.17), (3.7) – (3.9) can be expressed by the same objective $\min_{x \in X, u \in U_2(x)} f(x, u)$ and $U_2(x)$ defined by:

$$ u_{ij} \geq 1 - x_i - x_j, \quad (i, j) \in \tilde{S}, \; (i, j) \in E, \quad (3.14) $$

$$ \begin{cases} u_{ij} \geq u_{kj} - x_i, & k \in N(i, j), \; \text{if} \; d(i, j) \leq d(j, i), \\ u_{ij} \geq u_{jk} - x_j, & k \in N(j, i), \; \text{if} \; d(i, j) > d(j, i), \end{cases} \quad (i, j) \in \tilde{S}, \; (i, j) \not\in E, \quad (3.15) $$

$$ u_{ij} \geq 0, \quad (i, j) \in \tilde{S}. \quad (3.16) $$

The proposition claims that

$$ \min_{x \in X, u \in U_1(x)} f(x, u) = \min_{x \in X, u \in U_2(x)} f(x, u). \quad (3.17) $$

Throughout the proof, we fix $x \in X$ and deal with the optimal solution $\{u_{ij}\} \in U_1(x)$ to problem $\min_{u \in U_1(x)} f(x, u)$:

$$ u_{ij} = \max_{\mathcal{P}_{ij} \in \mathcal{P}_{ij}} \left( 0, \max_{t \in V(\mathcal{P}_{ij})} 1 - \sum_{t \in V(\mathcal{P}_{ij})} x_t \right) = \max \left( 0, 1 - \sum_{t \in V(\mathcal{P}_{ij}^{\text{opt}})} x_t \right), \quad (i, j) \in \tilde{S}, \quad (3.18) $$

for the optimal simple path $\mathcal{P}_{ij}^{\text{opt}}$ for each $(i, j) \in \tilde{S}$. Note that the value $\max_{\mathcal{P}_{ij} \in \mathcal{P}_{ij}} 1 - \sum_{t \in V(\mathcal{P}_{ij})} x_t$ is unique for a given $x \in X$, which is why the solution $u_{ij}$ is unique too. Consider first the following auxiliary statement: if $(i, j) \not\in E$ and $k$ is the neighbor of $i$ in the path $\mathcal{P}_{ij}^{\text{opt}}$, then the
optimal path $P^{opt}_{ij}$ between $i$ and $j$ defines the optimal path between $k$ and $j$ as well:

$$
\max_{P_{kj} \in P_{kj}} \left( 1 - \sum_{t \in V(P_{kj})} x_t \right) = 1 - \sum_{t \in V(P^{opt}_{ij}) \setminus \{i\}} x_t.
$$

(3.19)

Indeed, suppose there exists another simple path $P_{kj}$ such that

$$
1 - \sum_{t \in V(P_{kj})} x_t > 1 - \sum_{t \in V(P^{opt}_{ij}) \setminus \{i\}} x_t.
$$

If $P_{kj}$ is a simple path that does not include $i$, then $P_{ij} = P_{kj} \cup \{(i, k)\}$ is a simple path that connects $i$ and $j$, with the property

$$
1 - \sum_{t \in V(P_{ij})} x_t = 1 - \sum_{t \in V(P_{kj})} x_t - x_i > 1 - \sum_{t \in V(P^{opt}_{ij}) \setminus \{i\}} x_t - x_i = 1 - \sum_{t \in V(P^{opt}_{ij})} x_t,
$$

which again contradicts the optimality of path $P^{opt}_{ij}$.

The proof of the proposition consists of two parts. First, we demonstrate that for any $x \in X$, the value of any feasible solution $u_{ij} \in U_2(x)$ is not less than the value of the optimal solution $u_{ij}$ to the $\min_{u \in U_2(x)} f(x, u)$ problem. Given that the objective function $f(x, u)$ is a linear combination of variables $u_{ij}$ with nonnegative coefficients, this observation will establish the following inequality:

$$
\min_{u \in U_1(x)} f(x, u) \leq \min_{u \in U_2(x)} f(x, u).
$$

Second, we demonstrate that the optimal solution $u_{ij}$ to the $\min_{u \in U_2(x)} f(x, u)$ problem belongs to the set $U_2(x)$; this way we will prove that

$$
\min_{u \in U_1(x)} f(x, u) \geq \min_{u \in U_2(x)} f(x, u),
$$

and finalize proving the main statement (3.17) of the proposition.

1. Let $\{u_{ij}\} \in U_2(x)$. It is sufficient to show that

$$
u_{ij} \geq u_{ij}, \quad \forall (i, j) \in \tilde{S}.
$$

(3.20)

The above inequality is true if $u_{ij} = 0$. Hence, we only consider pairs $(i, j) \in \tilde{S}$ such that

$$
u_{ij} = 1 - \sum_{t \in V(P^{opt}_{ij})} x_t > 0.
$$

(3.21)

The proof will be conducted by induction over the length of the path $|P^{opt}_{ij}|$ between nodes $i$ and $j$. The inequality (3.20) is clearly true if $i$ and $j$ are adjacent, since constraints (3.11) and (3.14) are identical for adjacent node pairs in both formulations. Suppose that the inequality (3.20) is true for all node pairs $(s, t) \in \tilde{S}$ with optimal paths $P^{opt}_{st}$ of length at most $d$. Then,
for a pair \((i, j) \in \tilde{S}\) with \(d(i, j) \leq d(j, i)\), and the optimal path of length \(d + 1\), consider the constraint from the constraint pool (3.15) with \(k \in V(P_{ij}^{opt})\):

\[
    u_{ij} \geq u_{kj}^* - x_i \geq u_{kj}^* - x_i = 1 - \sum_{t \in V(P_{ij}^{opt}) \setminus \{i\}} x_t - x_i = u_{ij},
\]

where the second inequality follows from the induction assumption: since \(k\) and \(j\) have an optimal path of length \(d\), then \(u_{kj}^* \geq u_{kj}^*\). Hence, we proved the claim (3.20).

2. Here, we need to show that \(\forall x \in X, \min_{u \in U_1(x)} f(x, u) \geq \min_{u \in U_2(x)} f(x, u)\). As before, let \(\{u_{ij}\}\) be the optimal solution to problem \(\min_{u \in U_1(x)} f(x, u)\) defined according to (3.18). Suppose, without loss of generality, that \(d(i, j) \leq d(j, i)\) and consider an arbitrary constraint from the pool (3.15) for \(k \in N(i, j)\):

\[
    u_{ij} \geq u_{kj}^* - x_i.
\]

We demonstrate that \(u_{ij}\) and \(u_{kj}^*\) satisfy it as well. Indeed,

\[
    u_{ij} = \max \left( 0, 1 - \sum_{t \in V(P_{ij}^{opt})} x_t \right) \geq \max \left( -x_i, 1 - \sum_{t \in V(P_{ij}^{opt})} x_t \right),
\]

\[
    \max \left( -x_i, 1 - x_i - \sum_{t \in V(P_{ij}^{opt})} x_t \right) = \max \left( 0, 1 - \sum_{t \in V(P_{ij}^{opt})} x_t \right) - x_i = u_{kj}^* - x_i, \quad (3.22)
\]

where we employed relation (3.19) in the last transition. Hence,

\[
    u_{ij} \geq u_{kj}^* - x_i. \quad (3.23)
\]

Since \(k\) was chosen arbitrarily, this argument proves the statement for any \(k \in N(i, j)\). Therefore, the optimal solution \(\{u_{ij}\}\) to the \(\min_{u \in U_1(x)} f(x, u)\) problem belongs to the set \(U_2(x)\), and we can conclude that \(\min_{u \in U_1(x)} f(x, u) \geq \min_{u \in U_2(x)} f(x, u)\).

4. Other Connectivity Measures

In this section we will briefly present optimization problem formulations for critical node detection with respect to other connectivity measures.

4.1. Minimizing the Maximum Size of Connected Component

Following Veremyev et al. (2014b), the size of the maximum connected component connectivity measure can be expressed as follows:

\[
    \mathcal{F}_2(x_1, \ldots, x_n) = \min_{u_{ij}} \tau \quad (4.1)
\]

subject to

\[
    \tau \geq \sum_{j \in V, i \neq j} u_{ij}^* + 1 - x_i, \quad i \in V, \quad (4.2)
\]

where variables \(u_{ij}\), \((i, j) \in \tilde{S}\) are constrained according to either (2.3) – (2.6), (3.3) – (3.6) or (3.7) – (3.9). Note that formally speaking constraint (4.2) contains variables that have not been defined, i.e., when \(i\) or \(j\) are in the \(D_1\) set. However, it is easy to see that if \(i \in D_1\) and
\[ j \in V \setminus D_1, \ i < j, \] the following identity holds:

\[ u_{ij} = u^*_N(i), j, \]

or, if \( i \in V \setminus D_1 \) and \( j \in D_1, \ i < j, \) then

\[ u_{ij} = u^*_i, N(j). \]

Likewise, if both \( i, j \in D_1, \ i < j, \) then

\[ u_{ij} = u^*_N(i), N(j). \]

Incorporation of all these constraints completes the problem formulation.

### 4.2. Minimizing the Inverse of the Number of Connected Components

First of all, the objective to minimize the inverse of the number of connected components is equivalent to maximizing the number of connected components. The following maximization problem formulation was proposed in Veremyev et al. (2014b). Let \( t_i \in \{0, 1\} \) be a set of variables, such that \( t_i = 1 \) if node \( i \) is selected to be the unique representative of the connected component which node \( i \) belongs to. With that, the connectivity measure equal to the number of connected components can be obtained as:

\[
F_3(x_1, \ldots, x_n) = \max_{t, u} \sum_{i \in V} t_i \tag{4.3}
\]

subject to

\[
\begin{align*}
    u_{ij} &= u^*_N(i), j, & i \in D_1, \ j \in V \setminus D_1, \ i < j, & \tag{4.4} \\
    u_{ij} &= u^*_i, N(j), & i \in V \setminus D_1, \ j \in D_1, \ i < j, & \tag{4.5} \\
    u_{ij} &= u^*_N(i), N(j), & i, j \in D_1, \ i < j, & \tag{4.6} \\
    t_i &\leq 1 - x_i, & i \in V, & \tag{4.7} \\
    t_i + t_j &\leq 2 - u_{ij}, & (i, j) \in S, & \tag{4.8} \\
    t_i &\in \{0, 1\}, & i \in V, & \tag{4.9}
\end{align*}
\]

with variables \( u_{ij}, (i, j) \in \tilde{S} \) constrained according to either (2.3) – (2.6), (3.3) – (3.6) or (3.7) – (3.9). The formulation is enhanced by the following set of “symmetry” (i.e., the ambiguity of selecting a representative node of every connected component) breaking constraints:

\[
t_i \leq 1 - u_{ij}, \quad i \in V, (i, j) \in S. \tag{4.10}
\]

The above condition ensures that the node with maximum index is selected to be equal to 1 in every connected component. Note that Shen et al. (2012) presented another approach to find the number of connected components, which required the introduction of continuous variables only. For simplicity of presentation, we restricted our computational experiments only to the above described formulation.

### 5. Computational Experiments

Computational experiments aim to demonstrate numerically the efficiency of proposed improvements compared to a benchmark. They are conducted using the following five real-life mid size network instances:

- 118 ieee bus network, a part of the American Electric Power System as of 1962 with \( n = 118, |E| = 179, D_1 = 7, \) obtained from https://www2.ee.washington.edu/research/pstca.

• Collaborators 379 network with \( n = 379, |E| = 914, D_1 = 27 \), available from Rossi and Ahmed (2015).

• Erdos 472 collaboration network with \( n = 472, |E| = 1,314, D_1 = 83 \), available from Davis and Hu (2011), which is a disconnected graph. While the above theory applied for connected graphs, only minor modifications are needed to adopt it to disconnected networks. For example, a variable \( u_{ij} \) will be defined only for \( i \) and \( j \) that belong to the same connected component in the original graph, otherwise we let \( u_{ij} = 0 \) in the above problem formulations.

• 494 bus network with \( n = 494, |E| = 586, D_1 = 146 \), available from Davis and Hu (2011).

Computational results presented below were obtained using a machine equipped with Windows 8.1x64 operating system, Intel Core(TM) i5-4200M CPU 2.5GHz processor, 6GB RAM, and Gurobi 6.5.1 64-bit solver, Gurobi Optimization (2015), all four available threads enabled. Optimization models were implemented using 64-bit Python 3.5.1. We experimented with all three connectivity objectives \( F_1, F_2, F_3 \). Regardless of the connectivity measure, the following five optimization approaches are tested:

1. The benchmark approach, denoted by \( B \), involves connectivity constraints (2.3) – (2.6) proposed in Veremyev et al. (2014a).

2. The improved version of the benchmark formulation (3.3) – (3.6), denoted by \( P1 \).

3. The improved version of the triangular approach (3.7) – (3.9), denoted by \( P2 \).

The following two modifications of \( P2 \) are inspired by additional tools provided by modern commercial optimization solvers. Gurobi Optimization (2015) allows to specify all or some part of the constraint as lazy ones. The pool of lazy constraints is originally omitted from the problem formulation and is brought back to the model when an incumbent solution violates it. This corresponds to lazy constraint parameter equal to 2. Moreover, incorporation of such constraints back to the model can be more aggressive and be done during the earlier stage, when a relaxation solution violates them. This corresponds to lazy parameter set to 3. Note that the pool of constraints (3.8) is large, therefore not all of them will be active. In order to use only the active ones, the following two additional approaches are suggested:

4. Improved version of the triangular approach (3.7) – (3.8), with set of constraints (3.8) specified as lazy with parameter 3, denoted by \( P2+lazy(3) \).

5. Improved version of the triangular approach (3.7) – (3.8), with set of constraints (3.8) specified as lazy with parameter 2, denoted by \( P2+lazy(2) \).

Using considered network instances we illustrate first to what extent the proposed improvements help to reduce the size of the pool of triangular constraints. Table 2 demonstrates that simple arguments presented in the previous section sometimes allow to reduce the size of constraint pool (forth column) by more than two times compared, for instance, to the required pool of constraints according to Dinh and Thai (2011) in column three. Constructing \( N(i, j) \) neighborhoods to create the new pool of constraints (3.8) was performed using the NetworkX package, Hagberg et al. (2008).

In order to test the problem formulation, only three network instances were selected for each connectivity measure. The choice of network instances was made empirically in order to make corresponding problems neither very difficult nor very easy to solve by any considered problem formulation. The choice of parameter \( k \) was driven by the same objective: some \( k \) makes a problem instance very easy to solve, some – very complicated. The goal was to solve the majority of problem instances within 50,000 seconds of time limit by most of the problem
formulations. Therefore, network instances and values of $k$ considered are not the same for all connectivity measures.

Tables 3, 4 and 5 compare the quality of linear relaxation of three optimization models and the time to obtain it. As expected, approach $P2$ outperforms the other two with respect to the quality of linear relaxation at the expense of computational effort to obtain it in most of the cases.

<table>
<thead>
<tr>
<th>Network</th>
<th>$k$</th>
<th>$\sum_{(i,j)\in S\setminus E} d(i)$</th>
<th>$\sum_{(i,j)\in S\setminus E} \min(\tilde{d}(i), \tilde{d}(j))$</th>
<th>$\sum_{(i,j)\in S\setminus E} \min(d(i,j), d(j,i))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>118 iee bus</td>
<td>10</td>
<td>6,136.62</td>
<td>6,672.34</td>
<td>24,393.00</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>3,168.11</td>
<td>4,548.56</td>
<td>16,804.50</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1,937.80</td>
<td>2,694.30</td>
<td>11,319.00</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1,250.87</td>
<td>1,085.99</td>
<td>5,275.90</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>863.08</td>
<td>1,085.99</td>
<td>5,275.90</td>
</tr>
<tr>
<td>US Airlines 332</td>
<td>100</td>
<td>275.27</td>
<td>294.47</td>
<td>407.73</td>
</tr>
<tr>
<td></td>
<td>105</td>
<td>244.34</td>
<td>258.56</td>
<td>338.33</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>216.95</td>
<td>226.47</td>
<td>281.95</td>
</tr>
<tr>
<td></td>
<td>115</td>
<td>192.00</td>
<td>199.24</td>
<td>236.07</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>169.13</td>
<td>174.05</td>
<td>197.25</td>
</tr>
<tr>
<td>Erdos 472</td>
<td>10</td>
<td>2,052.37</td>
<td>3,468.64</td>
<td>19,638.89</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>1,834.35</td>
<td>2,555.22</td>
<td>9,759.10</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1,360.10</td>
<td>1,988.70</td>
<td>6,000.86</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1,171.07</td>
<td>1,635.44</td>
<td>3,951.37</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>1,022.39</td>
<td>1,381.53</td>
<td>2,830.00</td>
</tr>
</tbody>
</table>

Table 3: Computational results of solving the linear programming relaxation of minimization of the number of connected pairs of nodes, $P_1$, problem. Computational times are presented in seconds.

Finally, Tables 6, 7 and 8 provide results of computational experiments with $F_1$, $F_2$ and $F_3$ connectivity metrics, respectively. For every problem instance, the best CPU time is highlighted with bold font. Results of these tables do not suggest a single solution approach that dominates the others. However, it can be noticed that the $P2$ approach, either with lazy constraints or without them, demonstrates the best performance in the majority of cases. Yet, in some cases and despite weaker linear relaxations, the benchmark or its modification $P1$ solution approaches remain the most efficient ones from the computational point of view, see especially Table 7.

The computational advantages of the improved problem formulations, demonstrated using a set real-life networks, are presumably based on network sparsity. Indeed, for dense networks the sets of essential neighbors $N(i, j)$ are likely to be the same with regular sets of neighbors $N(i)$. Hence, the question one might ask is as follows: how sparse a network should be in order to enjoy the computational benefits of improved formulations. We run a set of experiments using uniform random graphs with 80 nodes and a range of densities from 2.5% to 10%. Density is defined as the fraction of existing edges in a graph to the maximum possible number of edges in a complete graph, $n(n−1)$. Table 9 provides the results of experiments, where computational
Table 4: Computational results of solving the linear programming relaxation of minimization of the size of the largest connected component, $F_2$, problem. Computational times are presented in seconds.

<table>
<thead>
<tr>
<th>Network</th>
<th>$k$</th>
<th>$B$ relaxation</th>
<th>$P_1$ relaxation</th>
<th>$P_2$ relaxation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Obj</td>
<td>CPU</td>
<td>Obj</td>
<td>CPU</td>
</tr>
<tr>
<td></td>
<td>Obj</td>
<td>CPU</td>
<td>Obj</td>
<td>CPU</td>
</tr>
<tr>
<td></td>
<td>Obj</td>
<td>CPU</td>
<td>Obj</td>
<td>CPU</td>
</tr>
<tr>
<td>IEEE 118 bus</td>
<td>31</td>
<td>1.84 0.5</td>
<td>1.86 2.0</td>
<td>2.17 2.4</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>1.77 0.6</td>
<td>1.79 2.0</td>
<td>2.04 2.5</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>1.71 0.5</td>
<td>1.72 2.0</td>
<td>1.93 2.5</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>1.65 0.5</td>
<td>1.66 2.0</td>
<td>1.83 2.4</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>1.59 0.5</td>
<td>1.59 2.0</td>
<td>1.73 2.4</td>
</tr>
<tr>
<td>US Airlines 332</td>
<td>10</td>
<td>93.47 119.6</td>
<td>123.30 161.5</td>
<td>217.68 941.2</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>51.62 85.7</td>
<td>67.80 179.3</td>
<td>175.30 929.7</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>30.80 58.5</td>
<td>39.27 152.9</td>
<td>132.93 696.6</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>18.88 42.2</td>
<td>23.40 119.6</td>
<td>91.81 965.7</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>12.11 28.0</td>
<td>14.52 99.3</td>
<td>58.02 729.2</td>
</tr>
<tr>
<td>Collaborators 379</td>
<td>10</td>
<td>14.13 17.7</td>
<td>25.67 50.0</td>
<td>52.33 64.9</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>11.03 14.2</td>
<td>17.69 42.0</td>
<td>29.18 44.0</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>9.07 13.8</td>
<td>12.94 38.4</td>
<td>19.35 48.8</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>7.74 13.0</td>
<td>10.00 38.1</td>
<td>14.01 37.9</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>6.81 12.3</td>
<td>8.35 36.9</td>
<td>10.60 37.7</td>
</tr>
</tbody>
</table>

Table 5: Computational results of solving the linear programming relaxation of maximization of the number of connected components, $F_4$, problem. Computational times are presented in seconds.

<table>
<thead>
<tr>
<th>Network</th>
<th>$k$</th>
<th>$B$ relaxation</th>
<th>$P_1$ relaxation</th>
<th>$P_2$ relaxation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Obj</td>
<td>CPU</td>
<td>Obj</td>
<td>CPU</td>
</tr>
<tr>
<td></td>
<td>Obj</td>
<td>CPU</td>
<td>Obj</td>
<td>CPU</td>
</tr>
<tr>
<td></td>
<td>Obj</td>
<td>CPU</td>
<td>Obj</td>
<td>CPU</td>
</tr>
<tr>
<td>US Airlines 332</td>
<td>10</td>
<td>67.87 96.9</td>
<td>66.95 82.1</td>
<td>57.88 94.6</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>87.30 57.1</td>
<td>86.66 81.1</td>
<td>74.50 134.8</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>102.91 39.4</td>
<td>102.01 79.3</td>
<td>89.17 159.3</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>115.83 22.1</td>
<td>115.05 78.7</td>
<td>102.53 169.1</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>127.94 18.7</td>
<td>127.36 73.5</td>
<td>115.50 155.7</td>
</tr>
<tr>
<td>Collaborators 379</td>
<td>10</td>
<td>79.04 14.7</td>
<td>61.01 40.9</td>
<td>48.25 60.8</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>95.23 14.1</td>
<td>79.74 40.2</td>
<td>66.92 43.6</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>109.71 13.4</td>
<td>95.85 39.8</td>
<td>83.19 42.3</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>122.66 13.3</td>
<td>110.24 40.1</td>
<td>98.19 40.4</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>134.48 13.1</td>
<td>122.81 39.8</td>
<td>112.86 40.9</td>
</tr>
<tr>
<td>494 bus</td>
<td>10</td>
<td>90.17 27.3</td>
<td>76.10 44.7</td>
<td>45.76 229.0</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>110.32 21.9</td>
<td>97.72 46.4</td>
<td>65.78 211.6</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>128.01 22.3</td>
<td>116.75 41.9</td>
<td>85.45 143.8</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>143.91 20.6</td>
<td>133.53 37.8</td>
<td>104.22 82.1</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>158.91 19.2</td>
<td>149.05 37.1</td>
<td>121.63 58.6</td>
</tr>
</tbody>
</table>

advantages of improved formulations can be observed for instances with the density up to 9.5% and disappear for even slightly denser networks.

6. Conclusion

This paper considered the problem of identifying the most important nodes of an undirected graph, whose removal leads to the largest defragmentation of the network, measured by a connectivity measure of a graph. Examples of such measures include the total number of connected pairs of nodes, the size of the largest connected component, or the inverse number of the number of connected components. Several optimization problem formulations were considered to identify such nodes, different in the number of constraints they require. One of the known formulations requires an exponential number of constraints, another needs $O(nm)$ constraints, which can be reduced to $O(n^3)$, yet another one allows to formulate the problem with an $O(n^2)$ number of constraints. The focus of this paper was to study which of the $O(nm)$ constraints can be omitted due to redundancy. Moreover, the equivalence result has been established,
Table 6: Computational results of solving problems minimizing the number of connected pairs of nodes connectivity measure, $F_1$, are presented. US Airlines 332 and 494 bus networks are both connected, therefore reported objective values represent fractions in percent of the total number of originally connected node pairs, calculated as $\frac{n_m}{n-1}$. The Erdos 472 network is disconnected, so the objectives reported are the fractions in percent of the number of connected node pairs in the original graph, equal to 91,806.

Table 7: Computational results of solving problems minimizing the maximum size of connected component, $F_2$, are presented. The sign ‘−’ denotes that a problem instance could not be solved using specific solution method within 50,000 seconds of time.

stating that two formulations, one with exponential number of constraints and another with $O(nm)$ constraints, are equivalent from the quality of linear programming relaxation point of view. Computational experiments demonstrate that the $O(nm)$ problem formulation allows us to decrease solution time by a factor of 10 or more in some cases, compared to the most recent benchmark.

7. References

References

### Table 8: Computational results of solving problems maximizing the number of connected components connectivity measure, $F_3$, are presented. The sign “−” denotes that a problem instance could not be solved using specific solution method within 50,000 seconds of time.

| $m$ | $\text{den}$ | $\text{|(2.1)|}$ | $\text{|(2.15)|}$ | $\text{|(3.8)|}$ | $\text{B}$ | $\text{P1}$ | $\text{P2}$ | $\text{P2+lazy(2)}$ | $\text{P2+lazy(3)}$ | $\text{CPU (sec.)}$ |
|-----|-------------|-------------|-------------|-------------|------------|------------|------------|-----------------|-----------------|-----------------|
| 155 | 2.5        | 11,750      | 8,741       | 8,665       | 428.2      | 294.9      | 782        | 100.2          | 72.5            |
| 184 | 3.0        | 13,408      | 9,949       | 9,900       | 1,057.9    | 961.8      | 91.5       | 164.2          | 282.3           |
| 208 | 3.5        | 15,478      | 11,684      | 11,684      | 1,025.9    | 858.5      | 139.4      | 131.1          | 114.3           |
| 238 | 4.0        | 17,142      | 12,558      | 12,558      | 782.4      | 514.9      | 89.3       | 140.7          | 179.5           |
| 263 | 4.5        | 19,844      | 15,380      | 15,355      | 969.3      | 760.4      | 208.9      | 322.5          | 124.5           |
| 293 | 5.0        | 21,197      | 17,464      | 17,464      | 2,012.8    | 1,978.1    | 386.6      | 240.1          | 365.9           |
| 330 | 5.5        | 21,618      | 18,114      | 18,114      | 920.5      | 804.7      | 162.8      | 135.1          | 162.9           |
| 355 | 6.0        | 24,203      | 20,726      | 20,726      | 12,973.9   | 1,290.5    | 866.8      | 655.1          |                 |
| 388 | 6.5        | 27,023      | 22,764      | 22,764      | 9,114.9    | 6,945.0    | 1,251.6    | 679.1          | 573.3           |
| 409 | 7.0        | 27,961      | 23,346      | 23,346      | 7,385.5    | 7,872.4    | 847.5      | 666.8          | 481.9           |
| 426 | 7.5        | 28,392      | 24,070      | 24,070      | 5,231.4    | 5,129.1    | 807.2      | 642.1          | 409.8           |
| 463 | 8.0        | 30,808      | 26,153      | 26,153      | 4,971.2    | 4,237.8    | 694.2      | 485.3          | 481.2           |
| 488 | 8.5        | 33,996      | 27,766      | 27,766      | 4,198.3    | 5,426.7    | 481.4      | 260.4          | 425.4           |
| 515 | 9.0        | 34,098      | 29,552      | 29,552      | 4,221.9    | 4,112.4    | 602.4      | 422.4          | 773.7           |
| 545 | 9.5        | 35,468      | 31,715      | 31,715      | 10,833.2   | 2,345.5    | 770.0      | 426.4          | 1,207.7         |
| 571 | 10.0       | 36,888      | 32,823      | 32,823      | –          | –          | –          | –              |                 |

Table 9: Computational results comparing different problem formulations using a set of uniform random graphs with $n = 80$ nodes. $m$ denotes the random number of edges in the graph instance and $\text{den}$ stands for the density of the graph. $\text{|(2.1)|}$, $\text{|(2.15)|}$ and $\text{|(3.8)|}$ represent the sizes of corresponding constraint pools. All instances were solved with 15,000 seconds of time limit. The table demonstrates that computational advantages of the enhanced formulations hold for sparse networks with densities up to 9.5%, as instances with density 10.0% could not be solved by either of the approaches within the time limit.


Dinh, T. N., Thai, M. T., 2011. Precise Structural Vulnerability Assessment via Mathematical


