International Journal of Management Science
Vol 11, No 1, May 2005

An Algorithm for the Graph Disconnection Problem[*]

Young-Soo Myung[**]
Department of Business Administration,
Dankook University, Cheonan, Chungnam 330-714, Korea

Hyun-joon Kim
Department of Management Information Systems,
University of Ulsan, Ulsan, Korea

(Received Sep. 2003 ; Revised Apr. 2004/Aug. 2004 ; Accepted Jan. 2005)

ABSTRACT

We consider the graph disconnection problem, which is to find a set of edges such that the total cost of destroying the edges is no more than a given budget and the weight of nodes disconnected from a designated source by destroying the edges is maximized. The problem is known to be NP-hard. We present an integer programming formulation for the problem and develop an algorithm that includes a preprocessing procedure for reducing the problem size, a heuristic for providing a lower bound, and a cutting plane algorithm for obtaining an upper bound. Computational results for evaluating the performance of the proposed algorithm are also presented.

Keywords: Integer programming, Graph disconnection, Cutting plane algorithm

1. INTRODUCTION

The graph disconnection problem (GDP) is defined as follows: We are given an undirected graph $G=(V,E)$ with a designated source node, which we will refer to as node 1, a cost function c which assigns a destruction cost to each arc, a weight function w which assigns a gain to each node, and a budget b that we can spend for destroying edges. The objective of the problem is to find a set of edges such

[*] This work was supported by Korea Research Foundation Grant (KRF-2002-041-B00144).
[**] Email: myung@dankook.ac.kr

that the total cost of destroying the edges is no more than a given budget and the sum of weights on the nodes disconnected from a source node is maximized.

Martel et al. [6] have introduced the GDP and have shown that the problem with unit costs and weights is NP-hard. They have also presented an algorithm which enumerates all cuts whose cost less than or equal to the budget. Of course, their algorithm is an exponential time algorithm. Although no research on the GDP other than the Martel et al.'s work has been found, there exist similar ones on network attack. Cunningham [3] and Gusfield [5] have considered a problem of minimizing the ratio of the edge destruction cost to the number of disconnected components. They have presented strongly polynomial time algorithms for this problem.

Martel et al.'s study on the GDP is motivated by a need to evaluate the vulnerability of a network to attacks. The more vulnerable to attacks a network is, the less survivable it is. Survivability of a network is one of the most important issues in designing present-day communication networks. As fiber optic technology rapidly permeates communication networks, the high capacity of fiber links makes communication networks become to have relatively sparse network structures in comparison to the high redundancy of older systems. Such sparse networks, though cost-effective, are vulnerable to serious service disruptions following the failure of key components. Cosares et al. [2] noted that survivable networks are generally more expensive than those with less robust designs, and thus it is essential to quantify the trade-offs between cost and survivability. Wu [7] and Grötschel, Monma, and Stoer [4] also identified network survivability as one of the most significant issues to be considered when designing communication networks.

In this paper, we develop a practical algorithm to solve the GDP. As the problem is NP-hard, we are focusing on producing a feasible solution of good quality and strong upper bounds for the problem. Even in the case that we want to exactly solve the problem, we can use those procedures to implement a branch and bound method. This paper is organized as follows. The next section introduces the notation we use and describes some properties of the GDP. In Section 3, we propose a heuristic to produce lower bounds of the problem and the preprocessing procedure that enables to reduce the problem size. In Section 4, we present a mathematical programming formulation of the problem and develop a cutting plane algorithm for solving a linear programming (LP) relaxation to obtain upper bounds of the problem. In Section 5, we present a numerical example to illustrate the proposed algorithm. Computational results for evaluating the performance of the proposed algorithm are presented in Section 6.

2. NOTATIONS AND PROPERTIES

In a given undirected graph $G = (V, E)$, V is a set of nodes and E is a set of undirected edges. We allow multiple edges in G but no self-loops. Node 1 represents the designated source node and let $V_1 = V \setminus \{1\}$. For $S \subseteq V$, $\delta(S)$ represents the set of edges in E with one end node in S and the other in $V \setminus S$. For $S \subseteq S' \subseteq V_1$, $\delta(S')$ is called a 1-S cut. In other words, 1-S cut is a set of edges whose removal disconnects the subset of nodes S from node 1. We define a min-cost 1-S cut as a 1-S cut with the minimum edge costs and a maximum min-cost 1-S cut as a min-cost 1-S cut such that the sum of weights for the nodes in S part is the largest. Later, we will show that for every $S \subseteq V_1$, the maximum min-cost 1-S cut exists uniquely. Consider an example presented in Figure 1, where the edge costs are shown above the corresponding edges. We assume that the node weight of each node is 1 and our budget for destroying edges is 3. Let's define the following five subsets of V_1: $S_1 = \{4\}$, $S_2 = \{2, 4\}$, $S_3 = \{4, 5\}$, $S_4 = \{2, 4, 5\}$, $S_5 = \{2, 3, 4, 5\}$. Then by definition, all of $\delta(S_1)$, $\delta(S_2)$, $\delta(S_3)$, $\delta(S_4)$, $\delta(S_5)$ are 1-$\{4\}$ cuts and the first four cuts are min-cost 1-$\{4\}$ cuts. Among them $\delta(S_4)$ is the maximum min-cost 1-$\{4\}$ cut.

We also use the notation for the summation of edge costs and node weights such that $c(E') = \sum_{e \in E'} c(e)$ for any $E' \subseteq E$ and $w(S) = \sum_{i \in S} w(i)$ for any $S \subseteq V_1$. We assume positive demand and cost. We will call a subset of nodes $S \subseteq V_1$ a *separable node set,* if the value of a min-cost 1-S cut is less than or equal to our budget Therefore, the nodes in a separable node set can be disconnected from node 1 while keeping our budget restriction. Then, the GDP can be described as the problem of selecting a separable node set with maximum weights, that we call *an optimal separable node set.* In the example in Figure 1, $\{2, 4, 5\}$ is a separable node set but $\{3, 4, 5\}$ is not.

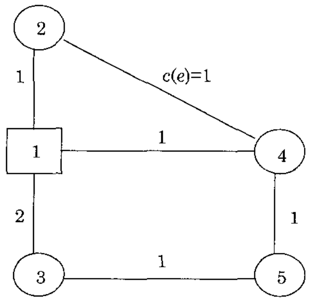

Figure 1. An example network

The following observations on 1-S cuts will be useful when we develop an algorithm for the GDP.

Lemma 1 Suppose that S_1, $S_2 \subseteq V_1$. If $\delta(S_1)$ and $\delta(S_2)$ are min-cost 1-S cuts, then so is $\delta(S_1 \cup S_2)$.

Proof. It is well known that cut cost function $c(\delta(S))$ satisfies the following submodularity condition: $c(\delta(S_1)) + c(\delta(S_2)) \geq c(\delta(S_1 \cap S_2)) + c(\delta(S_1 \cup S_2))$. From the fact that $\delta(S_1)$ and $\delta(S_2)$ are min-cost 1-S cuts, both $\delta(S_1 \cap S_2)$ and $\delta(S_1 \cup S_2)$ should be min-cost 1-S cuts. □

Corollary 2 For every $S \subseteq V_1$, the maximum min-cost 1-S cut exists uniquely.

Corollary 3 For $S_1 \subseteq S_2 \subseteq S \subseteq V_1$, if $\delta(S)$ is the maximum min-cost 1-S_1 cut, then it is also the maximum min-cost 1-S_2 cut.

Corollary 4 If $S \subseteq V_1$ is the optimal separable node set for the GDP, then $\delta(S)$ is the maximum min-cost 1-S cut.

The maximum min-cost 1-S cut can be derived by solving a maximum flow problem with node 1 as a single source and the nodes in S as multiple sinks. Note that the nodes in node 1 part of the maximum min-cost 1-S cut can be identified by finding nodes reachable from node 1 in the residual graph for the corresponding maximum flow problem. For the maximum flow algorithms and the definition of the residual network, refer to Ahuja et al. [1].

3. HEURISTIC AND PREPROCESSING PROCEDURE

In this section, we develop a heuristic of producing a feasible solution and a preprocessing procedure that reduces the problem size. Our heuristic is an add-type heuristic that iteratively selects a node to construct a separable node set. Based on Corollary 4, we always maintain the selected subset of nodes at each iteration, say S, such that $\delta(S)$ is the maximum min-cost 1-S cut. For this purpose, when we include a node $i \in V_1 \backslash S$ into the current set S, we include all nodes constituting the $S \cup \{i\}$ part of the maximum min-cost 1-$(S \cup \{i\})$ cut.

Now we explain how our heuristic selects a node to be included into the current set S at each iteration. Note that a node $i \in V_1 \backslash S$ can be included into the current set S only when the min-cost 1-$(S \cup \{i\})$ cut value is no more than our budget. The quality of the obtained solution depends on how to select a node among several candidate ones. We have tested three different strategies. The first strategy, called Algorithm LB1, is to select a node such that the ratio of the increased node weights to the increased edge costs is maximized. The second strategy, called Algorithm LB2, is to select a node with the maximum weight first. The last one, called Algorithm LB3, uses the result of an LP relaxation which will be introduced in the next section. LB3 selects a node whose corresponding variable in an LP relaxation has the most positive value. We will report the computing experiments of the three different strategies in the section on computational results.

In order to determine whether a node can be included without violating the budget constraint, we need to calculate maximum min-cost 1-($S \cup \{i\}$) cuts for all $i \in V_1 \setminus S$, that is, we need to solve $|V_1 \setminus S|$ maximum flow problems. In our heuristic, however, to avoid computational burden, we use $\delta(S \cup S_i)$ instead of the maximum min-cost 1-($S \cup \{i\}$) cut where $\delta(S_i)$ is the maximum min-cost 1-$\{i\}$ cut, for each $i \in V_1 \setminus S$. Note that if the sum of costs for the edges in $\delta(S \cup S_i)$ is no more than our budget, so is the min-cost 1-($S \cup \{i\}$) cut value. When calculating a ratio with respect to a node i in Algorithm LB1, we also use $\delta(S \cup S_i)$ as the maximum min-cost 1-($S \cup \{i\}$) cut.

As we already mentioned, when we include a node $i \in V_1 \setminus S$ into the current set S, we include all nodes constituting the $S \cup \{i\}$ part of the maximum min-cost 1-($S \cup \{i\}$) cut. For this purpose, however, we don't use $\delta(S \cup S_i)$ but exactly calculate the maximum min-cost 1-($S \cup \{i\}$) cut. Initially, we compute $\delta(S_i)$ for each $i \in V_1$. By using that information, we develop a preprocessing procedure that reduces the problem size. Our preprocessing procedure is based on the following observation.

Remark 1. If $c(\delta(S_i))>b$ for any $i \in V_1$, the cost of every cut disconnecting 1 and i is more than our budget. In this case, any subset of V_1 containing node i can not be an separable node set, i.e., can not be a feasible solution. Therefore, in order to find an optimal separable node set we can limit our search to the subsets of $V_1 \setminus \{i\}$

Remark 1 implies that if $c(\delta(S_i))>b$ for any $i \in V_1$, we can find an optimal solution from the reduced graph obtained from the original graph by contracting nodes 1 and i into node 1. When the two nodes 1 and i are contracted, edges incident to node i are replaced by the edges incident to node 1 and self-loops formed by edges connecting 1 and i are removed. Notice that for every $S \subseteq V_1 \setminus \{i\}$, $\delta(S)$ in the original graph is the same as that in the reduced graph.

Our heuristic of constructing a separable node set and a preprocessing procedure is formally described as follows.

Algorithm LB1 (LB2, LB3)

```
Input: $G = (V, E)$, $\{c_e\}$, $\{w_i\}$, $b$.
Output: A separable node set $S$.
  for each $i \in V_1$ do
     construct the maximum min-cost 1-$\{i\}$ cut, $\delta(S_i)$
     if $c(\delta(S_i)) > b$, contract nodes 1 and $i$ [preprocessing]
  $S \leftarrow \emptyset$
  while $I = \{i \in V_1 \setminus S \mid c(\delta(S \cup S_i) \le b) \ne \emptyset$ do
     select a node $i \in I$ with the maximum ratio of $w(S_i \setminus S)$ to $c(\delta(S \cup S_i))$-$c(\delta(S))$
                       (with the maximum $w(S_i \setminus S)$ in Algorithm LB2)
                       (with the most positive fractional solution in Algorithm LB3)
     set $S$ such that $\delta(S)$ is the maximum min-cost 1-$(S \cup \{i\})$ cut.
```

4. CALCULATING UPPER BOUNDS

In this section, we describe the procedure for computing upper bounds of the problem. For this purpose, we first present an integer programming formulation of the problem and obtain an upper bound by solving an LP relaxation of the proposed integer programming model.

4.1 Integer programming formulation

In this subsection, we formulate the problem as an integer programming problem. We define edge variables $x(e)$ on edge set E such that $x(e)$=1 if an edge e is destroyed and $x(e)$=0, otherwise. We also define node variables $y(i)$ on node set V_1 such that $y(i)$=1 if a node i is disconnected from node 1 and $y(i)$=0, otherwise. Let $\boldsymbol{P}(i)$ for each $i \in V_1$ be the set of paths between node 1 and i. For any path $P \in \boldsymbol{P}(i)$, we let $E(P)$ denote the set of edges in path P. Then GDP can be represented as the following 0-1 integer programming problem.

$$(IP) \quad \max \quad \sum_{i \in V_1} w(i)y(i) \tag{1}$$

$$s.t. \quad \sum_{e \in E} c(e)x(e) \le b, \tag{2}$$

$$\sum_{e \in E(P)} x(e) \ge y(i), \quad \forall P \in \mathrm{P}(i), i \in V_1 \tag{3}$$

$$x(e) \in \{0,1\}, \qquad \forall e \in E \tag{4}$$

$$y(i) \in \{0,1\}, \qquad \forall i \in V_1 \tag{5}$$

The constraint (2) reflects our budget restriction and the constraints (3) ensure that node i is disconnected from node 1 only when at least one edge is destroyed from each path connecting the two nodes.

We will solve an LP relaxation of (IP) to obtain upper bounds. To obtain tight bounds we also add a family of valid inequalities. Those inequalities are redundant to (IP) but provide better bounds when added to the LP relaxation of (IP). We call $L \subseteq V_1$ a *strong node set*, if no pair of nodes $\{i, j\} \in L$ is a separable node set. In other words, more than one node in a strong node set can not be disconnected from node 1 simultaneously. Then the following fact trivially holds.

Lemma 5 If $L \subseteq V_1$ is a strong node set, then the inequality

$$\sum_{i \in L} y(i) \leq 1 \tag{6}$$

is valid for (IP).

To find a strong node set, we construct an auxiliary graph from an instance of the GDP. The auxiliary graph $H = (V_1, A)$ contains each node in V_1 and has an edge between nodes i and j if $\{i, j\}$ is a separable node set. A stable set of the graph $H = (V_1, A)$ is a set of nodes any two of which are nonadjacent. Then, a strong node set corresponds to a stable set of the auxiliary graph $H = (V_1, A)$.

4.2 Cutting plane algorithm

We obtain an upper bound for the problem by solving an LP relaxation of (IP) with additional inequalities (6) where integrality conditions (4) and (5) are replaced by $0 \leq x(e) \leq 1$, for each $e \in E$ and $0 \leq y(i) \leq 1$, for each $i \in V_1$, respectively. Since we have enormous numbers of inequalities (3) and (6), we adopt a cutting plane algorithm to solve the LP relaxation of (IP). This algorithm initially solves an LP that consists of (2), $0 \leq x(e) \leq 1$, for each $e \in E$ and $0 \leq y(i) \leq 1$, for each $i \in V_1$. If we obtain a fractional solution, we find an inequality either (3) or (6) that cuts off the LP solution, and add this inequality to the current problem. We continue this procedure until we find an integer solution or can not identify either (3) or (6) violated by the current fractional solution.

To facilitate the algorithm, we need the so-called separation procedures, one for (3) and the other for (6), each of which finds, if any, an inequality violated by the current fractional solution. For the inequalities (3), we use the following well-known separation procedure. We calculate the length of the shortest path between node 1 and node i with weights $x(e)$ for all edges $e \in E$; then we check for violations of the inequalities (3) associated with a node $i \in V_1$ by comparing the shortest path length between 1 and i with $y(i)$.

As for the inequalities (6), it doesn't seem to be easy to develop an separation procedure since if such procedure exists, it would solve the problem of finding a maximum weight stable set that is NP-hard. So, we develop a separation heuristic that may not completely find a violated inequality but finds many of them in reasonable time. We initially construct the auxiliary graph $H=(V_1, A)$. Our heuristic selects nodes with positive values of variables $y(i)$ as many as possible while keeping the selected set as a strong node set; i.e., a stable set with respect to $H=(V_1, A)$. After selecting a strong node set, we check whether the inequality (6) associated with the selected set is violated by (x, y). Different sequences of selecting nodes may produce different strong node sets. However, we don't try to enumerate all the possible stable sets of H. Our strategy for selecting such sets is as follows: We always start a new search from a node not in any strong set previously selected and if no such node exists, we stop selecting a strong node set.

5. NUMERICAL EXAMPLE

In this section, we illustrate our algorithm using an example presented in Figure 2(a). The edge costs and the node weights are shown above the corresponding edges and nodes, respectively. Our budget for destroying edges is 20. We first illustrate Algorithm LB1. In the first step, we find the maximum min-cost 1-$\{i\}$ cut $\delta(S_i)$ for each $i \in V_1$. Then, $S_2 = \{2, 3\}$, $S_3 = \{3\}$, $S_4 = \{4\}$, $S_5 = \{5\}$, $S_6 = \{5,6\}$. Since $c(\delta(S_6)) > 20$, we contract nodes 1 and 6 that results in the graph shown in Figure 2(b). In the next step, we construct a feasible solution. Initially, $S = \varnothing$ and $I = \{2, 3, 4, 5\}$. We first select S_2, since it gives the maximum ratio of $w(S_i \setminus S)$ to $c(\delta(S \cup S_i)) - c(\delta(S))$. Then $I = \{4\}$. We select S_4 and obtain a final solution $S = \{2, 3, 4\}$ whose objective value is 55.

We now describe the process of obtaining an upper bound by solving an LP relaxation. We initially solve an LP that consists of (2), $0 \le x(e) \le 1$, for each $e \in E$ and $0 \le y(i) \le 1$, for each $i \in V_1$ and obtain a trivial solution in which $x(e) = 0$, for each $e \in E$ and $y(i) = 1$, for each $i \in V_1$. Then the separation procedure for (3) generates the following inequalities in the form of (3): $x(12) \ge y(2)$; $x(12) + x(23) \ge y(3)$;

$x(14) \geq y(4)$; $x(15) \geq y(5)$. After adding those inequalities to the current LP, we solve the new LP and obtain a fractional solution such that $x(12) = 1$, $x(14) = 1$, $x(15) = 0.538$, $x(23) = 0$, $x(25) = 0$, $x(34) = 0$; $y(2) = y(3) = y(4) = 1$, $y(5) = 0.538$. We repeat the separation procedure for (3) until we find no inequality (3) violated by the obtained fractional solution. Then we perform the separation procedure for (6). In this example, we generate two such inequalities, $y(2) + y(5) \leq 1$ and $y(3) + y(5) \leq 1$ The LP with those inequalities finally provides an integer solution that is optimal.

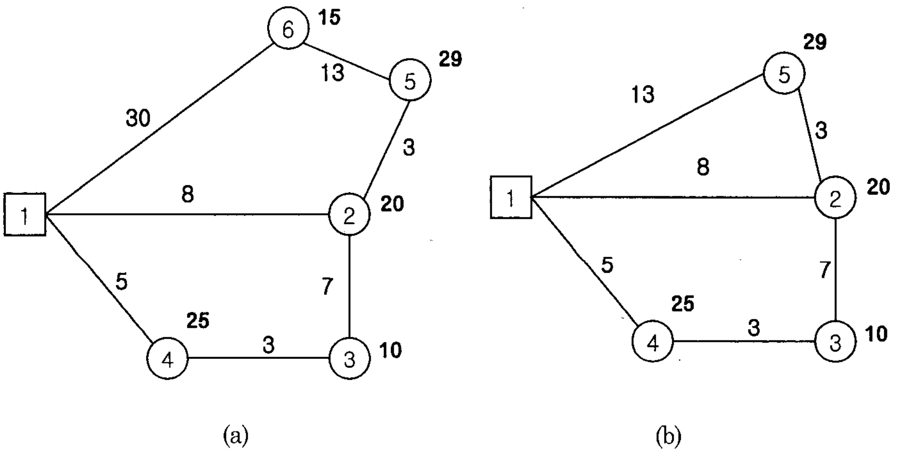

Figure 2. (a) Original network (b) Network after preprocessing

6. COMPUTATIONAL RESULTS

The proposed algorithm for calculating lower and upper bounds of the problem was coded in the language C and test runs were performed on a PC with 150MHz Pentium CPU. We used CPLEX callable library to solve an LP relaxation. We performed computational experiments using the randomly generated problems. To generated the test problems, we first drew a 100×100 rectangle on which node sites were randomly located.

The computational results on the randomly generated problems are summarized in Tables 1-3. Tables 1-3 show the results for 1080 randomly generated test problems, which were divided into 108 groups of 10 instances. Each group is classified by the size of the underlying graph and the level of budget measured by the ratio of the budget to the sum of edge costs. Each table shows the size of reduced networks to analyze the effect of preprocessing. The effect of preprocessing is

Table 1. Computational results for the problems with 30 nodes

Original Network			After Preprocessing		Lower Bounds			Upper Bounds		Number of cuts		GAP	Time (second)		
|V|	|E|	b/c(E)	|V|	|E|	LB1	LB2	LB3	UB1	UB2	(3)	(6)		PRE	LB	UB
30	100	0.40	3.2	2.6	3.5	3.4	3.5	4.2	3.8	2.50	0.60	0.021	0.04	0.00	0.12
30	100	0.45	4.3	4.1	5.0	5.1	5.2	5.7	5.2	4.10	0.80	0.000	0.04	0.00	0.07
30	100	0.50	5.3	5.4	5.9	6.1	6.2	7.1	6.4	5.50	0.90	0.031	0.03	0.01	0.08
30	100	0.55	5.9	6.3	6.5	6.1	6.7	7.8	6.9	6.80	0.90	0.016	0.05	0.00	0.06
30	100	0.60	7.0	8.2	6.9	5.9	7.1	8.3	7.4	8.90	0.80	0.019	0.04	0.01	0.06
30	100	0.65	8.3	11.8	7.9	7.2	8.4	9.0	8.9	11.30	0.20	0.029	0.05	0.00	0.06
30	100	0.70	9.5	14.7	7.9	7.0	8.4	9.8	9.2	15.00	0.80	0.061	0.04	0.00	0.08
30	100	0.75	11.3	21.9	8.1	7.3	8.6	10.7	9.9	31.40	0.90	0.114	0.03	0.01	0.11
30	100	0.80	12.2	26.2	9.4	8.0	9.1	12.0	10.8	43.70	0.90	0.112	0.04	0.00	0.13
30	100	0.85	14.7	36.0	18.9	17.1	18.7	21.1	19.8	54.30	1.40	0.074	0.05	0.00	0.15
30	100	0.90	16.0	41.9	19.1	17.1	18.6	22.3	20.6	81.80	1.60	0.115	0.03	0.01	0.20
30	100	0.95	16.7	44.7	19.2	17.3	18.3	23.3	21.3	99.00	1.60	0.158	0.05	0.00	0.25
30	200	0.40	1.1	0.1	0.5	0.5	0.5	0.5	0.5	0.10	0.00	0.000	0.09	0.00	0.06
30	200	0.45	1.3	0.3	1.2	1.2	1.2	1.2	1.2	0.30	0.00	0.000	0.11	0.00	0.05
30	200	0.50	1.4	0.4	1.4	1.4	1.4	1.4	1.4	0.40	0.00	0.000	0.06	0.00	0.05
30	200	0.55	1.6	0.6	1.5	1.5	1.5	1.6	1.5	0.60	0.10	0.000	0.09	0.00	0.06
30	200	0.60	2.3	1.6	2.4	2.4	2.4	2.4	2.4	1.70	0.20	0.000	0.06	0.00	0.06
30	200	0.65	3.1	2.8	3.3	3.4	3.4	3.7	3.4	2.90	0.40	0.000	0.08	0.01	0.07
30	200	0.70	3.6	3.6	4.1	4.2	4.2	4.7	4.2	3.60	0.50	0.000	0.07	0.00	0.06
30	200	0.75	4.5	4.9	4.5	4.7	4.7	5.3	4.7	4.70	0.70	0.000	0.09	0.00	0.05
30	200	0.80	5.6	8.4	4.8	5.0	5.0	5.9	5.0	8.90	0.80	0.000	0.08	0.00	0.07
30	200	0.85	6.6	12.0	4.8	5.1	5.1	6.5	5.1	14.10	0.90	0.000	0.07	0.00	0.07
30	200	0.90	8.0	17.2	5.0	5.2	5.2	6.8	5.2	25.20	1.00	0.000	0.08	0.00	0.08
30	200	0.95	10.0	29.2	5.1	5.4	5.4	7.8	5.4	41.20	1.00	0.000	0.09	0.00	0.12
30	300	0.40	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.000	0.14	0.00	0.06
30	300	0.45	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.000	0.13	0.00	0.05
30	300	0.50	1.2	0.2	0.3	0.3	0.3	0.3	0.3	0.20	0.10	0.000	0.13	0.00	0.06
30	300	0.55	1.8	0.9	1.5	1.5	1.5	1.5	1.5	0.90	0.20	0.000	0.12	0.00	0.06
30	300	0.60	2.3	1.8	1.7	1.8	1.8	1.9	1.8	1.70	0.30	0.000	0.12	0.00	0.06
30	300	0.65	3.4	4.9	2.9	2.9	2.9	3.2	2.9	4.40	0.40	0.000	0.12	0.00	0.06
30	300	0.70	4.9	9.5	4.0	4.0	4.0	4.4	4.0	8.80	0.50	0.000	0.12	0.01	0.07
30	300	0.75	6.4	16.3	4.7	4.7	4.7	5.3	4.7	13.20	0.70	0.000	0.12	0.00	0.08
30	300	0.80	8.9	32.5	5.3	5.5	5.5	6.4	5.5	41.10	0.80	0.000	0.12	0.00	0.12
30	300	0.85	12.4	66.9	5.3	5.6	5.6	7.6	5.6	102.90	0.90	0.000	0.12	0.00	0.27
30	300	0.90	14.7	88.1	5.8	6.1	6.1	9.4	6.1	128.10	0.90	0.000	0.12	0.01	0.31
30	300	0.95	17.1	118.0	5.8	6.1	6.1	13.0	6.1	208.50	1.10	0.000	0.13	0.00	0.85

Table 2. Computational results for the problems with 50 nodes

Original Network			After Preprocess-ing		Lower Bounds			Upper Bounds		Number of cuts		GAP	Time (second)		
|V|	|E|	b/c(E)	|V|	|E|	LB1	LB2	LB3	UB1	UB2	(3)	(6)		PRE	LB	UB
50	200	0.40	5.3	4.9	4.4	4.4	4.6	5.3	4.9	4.60	0.70	0.042	0.13	0.00	0.11
50	200	0.45	6.4	6.5	4.7	4.5	4.9	5.9	5.0	6.00	1.20	0.026	0.16	0.00	0.06
50	200	0.50	7.6	8.4	4.9	4.7	5.2	6.4	5.3	7.70	1.20	0.020	0.16	0.00	0.07
50	200	0.55	9.0	12.0	5.0	4.4	5.2	6.8	5.6	12.90	1.10	0.051	0.16	0.01	0.06
50	200	0.60	10.9	16.1	5.7	5.0	5.9	7.2	6.4	14.70	1.00	0.066	0.17	0.01	0.08
50	200	0.65	12.7	21.7	5.9	5.2	6.0	7.7	6.9	26.00	0.80	0.105	0.17	0.00	0.09
50	200	0.70	14.3	27.5	6.7	5.3	6.5	8.1	7.6	34.50	0.80	0.101	0.16	0.01	0.14
50	200	0.75	16.2	34.5	7.7	6.3	7.7	8.7	8.4	51.80	0.80	0.089	0.16	0.00	0.19
50	200	0.80	18.1	41.2	7.8	6.6	7.7	9.1	8.7	79.40	1.20	0.114	0.17	0.01	0.27
50	200	0.85	19.8	47.2	8.1	7.4	7.9	9.5	9.2	135.90	0.90	0.123	0.15	0.02	0.53
50	200	0.90	22.6	56.4	8.4	7.0	8.1	10.3	9.6	185.70	0.80	0.114	0.17	0.02	1.89
50	200	0.95	24.4	63.2	8.7	6.6	8.1	11.8	10.2	198.70	1.00	0.160	0.16	0.01	3.23
50	350	0.40	1.7	0.7	1.0	1.1	1.1	1.1	1.1	0.70	0.10	0.000	0.27	0.00	0.06
50	350	0.45	2.2	1.3	1.4	1.5	1.5	1.7	1.5	1.30	0.50	0.000	0.28	0.00	0.06
50	350	0.50	2.7	2.2	1.4	1.5	1.5	2.0	1.5	2.00	0.50	0.000	0.27	0.00	0.07
50	350	0.55	4.0	4.1	1.8	2.2	2.2	2.6	2.2	4.00	0.50	0.000	0.27	0.01	0.07
50	350	0.60	4.9	5.4	2.1	2.3	2.3	3.0	2.3	5.40	0.80	0.000	0.27	0.00	0.07
50	350	0.65	6.2	8.3	2.3	2.3	2.4	3.2	2.4	7.20	0.90	0.000	0.27	0.00	0.07
50	350	0.70	7.9	12.9	2.9	2.7	3.0	3.7	3.1	13.10	0.90	0.001	0.27	0.00	0.07
50	350	0.75	10.1	21.2	3.1	2.8	3.2	4.2	3.2	21.40	1.00	0.002	0.29	0.00	0.09
50	350	0.80	12.7	36.4	3.4	3.2	3.6	4.4	3.7	59.30	1.30	0.011	0.27	0.01	0.27
50	350	0.85	15.4	50.1	3.6	3.4	3.8	4.6	4.0	75.70	1.20	0.016	0.25	0.01	0.37
50	350	0.90	17.7	65.8	4.5	3.9	4.5	5.3	4.8	118.20	1.00	0.042	0.27	0.00	0.75
50	350	0.95	20.2	79.9	4.6	4.1	4.7	5.9	5.2	209.30	1.20	0.066	0.27	0.01	4.19
50	500	0.40	1.4	0.4	0.9	0.9	0.9	0.9	0.9	0.40	0.10	0.000	0.41	0.00	0.09
50	500	0.45	1.8	0.9	0.9	0.9	0.9	1.0	0.9	0.90	0.10	0.000	0.43	0.00	0.06
50	500	0.50	2.3	2.2	1.1	1.1	1.1	1.2	1.1	2.40	0.20	0.000	0.43	0.00	0.07
50	500	0.55	3.0	3.9	1.1	1.1	1.1	1.3	1.1	3.80	0.40	0.000	0.43	0.00	0.07
50	500	0.60	3.4	5.5	1.4	1.4	1.4	1.7	1.4	4.80	0.40	0.000	0.42	0.00	0.08
50	500	0.65	4.8	9.8	1.9	1.9	1.9	2.3	1.9	10.40	0.40	0.000	0.42	0.00	0.09
50	500	0.70	7.1	21.5	2.2	2.3	2.3	2.8	2.3	47.80	0.70	0.000	0.44	0.00	0.22
50	500	0.75	9.8	37.2	3.2	3.0	3.3	3.8	3.3	86.10	0.90	0.000	0.43	0.01	0.52
50	500	0.80	12.7	55.4	3.3	3.3	3.5	4.3	3.6	113.00	0.90	0.020	0.42	0.01	1.51
50	500	0.85	15.7	75.6	3.5	3.5	3.7	5.1	3.9	163.50	1.00	0.036	0.42	0.02	4.14
50	500	0.90	18.9	107.9	3.5	3.6	3.8	7.7	4.2	230.00	1.20	0.066	0.42	0.02	7.27
50	500	0.95	22.1	129.9	4.1	3.9	4.2	13.9	5.4	255.20	1.30	0.176	0.42	0.03	1.84

Table 3. Computational results for the problems with 80 nodes

Original Network			After Preprocessing		Lower Bounds			Upper Bounds		Number of cuts		GAP	Time (second)		
|V|	|E|	b/c(E)	|V|	|E|	LB1	LB2	LB3	UB1	UB2	(3)	(6)		PRE	LB	UB
80	300	0.40	7.5	7.9	2.8	2.4	2.8	3.2	2.8	7.00	1.10	0.018	0.50	0.00	0.06
80	300	0.45	9.9	11.4	3.0	2.7	2.9	3.5	3.2	10.20	0.70	0.067	0.50	0.00	0.06
80	300	0.50	13.3	17.4	3.2	2.3	3.2	3.9	3.6	15.70	0.70	0.099	0.50	0.01	0.08
80	300	0.55	14.8	19.6	3.5	3.0	3.6	4.1	3.9	18.40	0.90	0.082	0.50	0.00	0.10
80	300	0.60	17.7	25.5	3.9	3.2	3.7	4.6	4.3	25.50	0.60	0.103	0.45	0.01	0.11
80	300	0.65	20.6	32.3	4.3	3.8	3.8	4.8	4.6	30.00	0.60	0.069	0.50	0.01	0.14
80	300	0.70	23.2	40.5	4.3	4.2	4.2	5.1	4.9	42.60	0.90	0.072	0.50	0.01	0.19
80	300	0.75	25.6	46.8	4.8	4.3	4.5	5.4	5.3	48.90	0.50	0.077	0.51	0.01	0.23
80	300	0.80	28.5	56.1	5.0	4.3	4.7	5.7	5.6	73.70	0.80	0.104	0.50	0.02	0.32
80	300	0.85	32.2	68.6	5.2	4.4	4.9	6.0	5.9	99.80	0.50	0.117	0.50	0.03	0.50
80	300	0.90	35.4	80.5	5.4	4.8	5.2	6.3	6.3	128.30	0.20	0.138	0.50	0.03	0.75
80	300	0.95	41.0	109.3	14.8	14.0	14.6	15.8	15.8	210.30	0.40	0.160	0.51	0.06	1.32
80	500	0.40	3.0	2.3	1.9	1.9	1.9	2.0	1.9	2.30	0.60	0.000	0.83	0.01	0.07
80	500	0.45	4.8	4.5	2.0	2.0	2.0	2.4	2.0	4.40	1.00	0.000	0.86	0.00	0.06
80	500	0.50	6.6	7.4	2.0	2.0	2.1	2.7	2.1	7.90	1.00	0.024	0.84	0.00	0.08
80	500	0.55	8.3	10.4	2.1	2.2	2.2	2.9	2.3	10.10	1.20	0.031	0.85	0.00	0.07
80	500	0.60	11.5	17.0	2.2	2.4	2.4	3.2	2.5	16.40	1.30	0.044	0.85	0.00	0.08
80	500	0.65	14.4	25.7	2.4	2.5	2.5	3.5	2.7	27.40	1.30	0.013	0.84	0.00	0.12
80	500	0.70	18.5	36.8	2.8	2.9	3.0	3.7	3.0	38.10	1.30	0.009	0.83	0.01	0.16
80	500	0.75	23.6	56.0	3.0	2.7	3.0	4.0	3.3	55.00	1.50	0.066	0.84	0.01	0.29
80	500	0.80	27.7	72.9	3.4	2.8	3.3	4.2	3.8	95.00	1.50	0.093	0.85	0.01	0.45
80	500	0.85	32.0	91.1	3.5	2.8	3.4	4.4	4.1	136.20	1.10	0.115	0.85	0.02	0.68
80	500	0.90	36.3	115.1	4.0	3.0	4.1	4.6	4.4	192.90	0.90	0.068	0.75	0.03	1.19
80	500	0.95	42.2	155.0	4.1	3.0	4.2	4.9	4.7	487.80	1.20	0.090	0.83	0.04	7.61
80	700	0.40	2.4	1.6	0.7	0.7	0.7	0.9	0.7	1.60	0.30	0.000	1.20	0.00	0.09
80	700	0.45	3.8	3.8	1.4	1.4	1.4	1.5	1.4	3.40	0.50	0.000	1.20	0.00	0.08
80	700	0.50	5.2	7.7	1.5	1.5	1.5	1.7	1.5	5.50	0.50	0.003	1.20	0.01	0.06
80	700	0.55	7.1	13.5	1.7	1.7	1.7	2.0	1.8	9.90	0.50	0.020	1.22	0.01	0.09
80	700	0.60	10.7	27.0	1.8	1.9	1.8	2.3	1.9	24.80	0.60	0.022	1.21	0.00	0.18
80	700	0.65	14.1	42.3	2.1	2.2	2.2	2.7	2.3	54.30	0.80	0.029	1.21	0.01	0.34
80	700	0.70	17.9	64.3	2.5	2.4	2.6	3.1	2.6	260.70	1.20	0.001	1.20	0.02	4.83
80	700	0.75	21.9	88.1	2.6	2.5	2.8	3.4	2.8	278.40	1.20	0.005	1.21	0.02	6.00
80	700	0.80	25.5	118.8	2.7	2.3	2.9	3.8	3.0	437.90	1.10	0.020	1.21	0.04	18.62
80	700	0.85	29.3	145.7	2.8	2.4	2.9	4.4	3.1	550.30	1.30	0.043	1.22	0.07	50.80
80	700	0.90	35.3	189.2	2.8	2.4	2.9	5.9	3.5	633.80	1.30	0.150	1.22	0.09	67.49
80	700	0.95	39.4	223.0	3.2	2.7	3.1	7.3	4.3	773.10	1.60	0.189	1.22	0.10	71.11

strong especially when the level of budget is low. Lower and upper bounds, denoted by LB and UB, respectively, were obtained using the procedures presented in Section 3 and 4, respectively. UB1 corresponds to the value of the LP relaxation with the inequalities (3) while UB2 does to that with both the inequalities (3) and (6). LB1, LB2, and LB3 represent the values of the heuristic solutions generated by Algorithm LB1, Algorithm LB2, and Algorithm LB3, respectively.

The number of inequalities (3) and (6) added to the initial LP are also reported. GAP represents the ratio of the difference between the best lower bound and UB2 to the best lower bound. Each figure in all tables represents the average of the corresponding values for 10 problems. As seen in Tables 1-3, our algorithm calculated lower and upper bounds with good quality within a reasonable time even for the large size problems.

REFERENCES

[1] Ahuja, R. K., T. L. Magnanti and J. B. Orlin, *Network Flows*, Prentice Hall, New Jersey, 1993.

[2] Cosares, S., N. D. Deutch, I. Saniee, and O. J. Wasem, "SONET toolkit: A decision support system for designing robust and cost-effective fiber-optic networks," *Interfaces* 25 (1995), 20-40.

[3] Cunningham, W. H., "Optimal attack and reinforcement of a network," *Journal of the ACM* 32 (1985), 549-561.

[4] Grötschel, M., C. L. Monma, and M. Stoer, "Design of survivable networks," Network Models, M.O. Ball et al. (eds.), North-Holland, Amsterdam, 1995, 617-672.

[5] Gusfield, D., "Computing the strength of a graph," *SIAM Journal on Computing* 20 (1991), 639-654.

[6] Martel, C., G. Nuckolls, and D. Sniegowski, "Computing the disconnectivity of a graph," Working paper, UC Davis, 2001.

[7] T. Wu, *Fiber network survivability*, Artech House, Boston, 1992.