International Journal of Management Science
Vol 16, No 3, November 2010

An Efficient Algorithm for Finding the *k*-edge Survivability in Ring Networks*

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

(Received: July 7, 2010 / Revised: October 13, 2010 / Accepted: October 13, 2010)

ABSTRACT

Given an undirected network with a set of source-sink pairs, we are assumed to get a benefit if a pair of source and sink nodes are connected. The *k*-edge survivability of a network is defined as the total benefit secured after arbitrarily selected *k* edges are destroyed. The problem of computing *k*-edge survivability is known to be NP-hard and has applications of evaluating the survivability or vulnerability of a network. In this paper, we consider the k-edge survivability problem restricted to ring networks and develop an algorithm to solve it in $O(n^3|K|)$ time where n is the number of nodes and K is the set of source-sink pairs.

Keywords: Multicommodity Integer Flows, Cycles, Polynomial Time Algorithm

1. Introduction

Given an undirected graph $G = (V, E)$ and a set of m source-sink pairs (s_i, t_i), $i = 1, \cdots m$, we are assumed to get a positive benefit $d(i)$, if the source node s_i and the sink node t_i are connected. We say that two nodes are connected if there exists at least one path between them. Let $K = \{1, \cdots, m\}$ be the index set of source-sink pairs and we assume that $s_i < t_i$, for each $I \in K$. The *k*-edge survivability of a network is defined as the total benefit secured after an arbitrarily selected *k* edges are removed. The *k*-edge survivability has been used to evaluate the survivability of various different network topologies of communication networks [4, 7, 10, 14, 15]. For the relation between the *k*-

* This work was supported by the Korea Research Foundation Grant funded by the Korean Government (KRF-2009-B00178).
** Corresponding author, E- mail: myung@dankook.ac.kr

edge survivability model and each of various similar models dealing with the network survivability, refer to [8].

The k-edge survivability problem (k-SUV) is a problem of computing the k-edge survivability of a given network. For a set of edges $F \subseteq E$, let us define conk(F) as the set of source-sink pairs whose source and sink nodes are still connected after removing the edges in F. For expositional brevity, we use the following notation for summation. If x is defined on the elements of a subset $E' \subseteq E$ and a subset of $K' \subseteq K$, we denote $\sum_{e \in E'} x(e)$ and $\sum_{i \in K} x(i)$ by $x(E')$ and $x(K')$, respectively. For example, $d(K')$ denotes $\sum_{i \in K} d(i)$ for $K' \subseteq K$. Then (k-SUV) can be described as follows:

$$\begin{aligned} &\min_{F \subseteq E} \quad d(conk(F)) \\ &s.t. \quad |F| \leq k \end{aligned} \tag{1}$$

Myung *et al*. [10] have shown that (k-SUV) is NP-hard and Myung and Kim [7] have also shown that unless P = NP, there exists no α-approximation algorithm for the problem, for any $\alpha > 2/3$.

The purpose of this paper is to deal with (k-SUV) restricted to a ring network. An ring network is defined on an undirected cycle $G = (V, E)$ with a node set $V = \{1, 2, \cdots, n\}$ and an edge set $E = \{(1, 2), (2, 3), \cdots, (n\text{-}1, n), (n, 1)\}$. Ring networks have been widely used to model the various real world problems. For example, ring networks have been used to model a Synchronous Optical Network (SONET) [1, 13] and a VLSI circuit [2, 12]. For this reason, various combinatorial optimization problems restricted to ring networks have been studied [5, 9, 11].

Our study is motivated by the expectation that the simple structure of a ring network may enable us to solve (k-SUV) in polynomial time, despite the NP-hardness of the problem on an arbitrary network. For example, if $|K|$ is small and/or source and sink nodes are located in a particular way, we may easily find a set of k (or less than k) edges whose removal disconnects all the source-sink pairs. However, (k-SUV) in general is at least as hard as the minimum multicut problem in that the latter problem is polynomially (k-SUV), and the minimum multicut problem in a ring network does not seem to be trivially solved. Although tree network also has simple structure, we can show that (k-SUV) in an undirected tree is NP-hard using the fact that the minimum multicut problem in a tree is NP-hard. Garg *et al*. [3] have shown that the problem of finding the minimum number of edges whose removal would disconnect

the source and sink nodes of every source-sink pair is NP-hard. The decision version of this problem is polynomially transformed into a decision version of (*k*-SUV) in the same tree.

2. Algorithm

In this section, we develop an algorithm to solve (*k*-SUV) in a ring network. To avoid trivial cases, we assume that $k \geq 2$. Notice that if $F' \subseteq F$ for any two subsets F and F' of E, then conk(F) $\subseteq$ conk(F'). Therefore, we can always find an optimal solution of (*k*-SUV) that exactly includes k edges. In a ring network, exactly two paths exist between the source and sink nodes in each source-sink pair. Given a ring network $G = (V, E)$, let $G_e = (V, E \setminus \{e\})$ for each $e \in E$, denote a path network that is constructed by deleting e from the given ring network. Then if one path between s_i and t_i for $i \in K$, goes through an edge $e \in E$, the other path forms a subpath of G_e. Suppose that $F \subseteq E$ is an optimal solution with $|F| = k$ and $e \in F$. Based on our previous observation, $F \setminus \{e\}$ must be an optimal solution for ((*k*-1)-SUV) in the path network G_e. Our approach to solving (*k*-SUV) on a ring network is to compare the optimal solutions of ((*k*-1)-SUV) on $G = (V, E \setminus \{e\})$ for each $e \in E$.

From now on, we focus on an algorithm for solving (*k*-SUV) in a path network and if we do not specify $G = (V, E)$, we assume it as a path network with a node set $V = \{1, 2, \cdots, n\}$ and an edge set $E = \{(1, 2), (2, 3), \cdots, (n\text{-}1, n)\}$. We also assume that two edges $e = (i, i+1)$ and $e' = (j, j+1)$ are ordered such that $e < e'$ if $i < j$. As we mentioned, there exists a unique path in G between s_i and t_i for each source-sink pair (s_i, t_i), $i \in K$ and we let E_i denote the set of edges contained in the unique path between s_i and t_i. The problem (*k*-SUV) on a path network $G = (V, E)$ can be formulated as an integer programming problem. We define variables $x(e)$ on the edge set E such that $x(e) = 1$ if edge e is selected to be removed, and $x(e) = 0$ otherwise. We also define variables $y(i)$ for each source-sink pair (s_i, t_i), $i \in K$, such that $y(i) = 1$ if s_i and t_i are connected, and $y(i) = 0$ otherwise. Then (*k*-SUV) in a path network can be represented as the following 0-1 integer programming problem.

$$(P) \qquad \min \quad \sum_{i \in K} d(i) y(i)$$

$$s.t. \quad x(E_i) + y(i) \geq 1, \qquad i \in K \tag{2}$$

$$x(E) \le k, \tag{3}$$

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

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

The constraints (2) ensure that the source and sink nodes are connected if no edge in the path between the two nodes is removed.

Consider the following Lagrangian relaxation of (P) that dualizes (3) using the nonnegative multiplier u.

$$(P(u)) \qquad z(u) = \min \sum_{i \in K} d(i)y(i) + \sum_{e \in E} ux(e) - uk$$

$$s.t. \quad x(E_i) + y(i) \ge 1, \qquad i \in K \tag{2}$$

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

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

It is well known that $z(u)$ for any $u \ge 0$ is a lower bound on the optimal value of (P) and if (x, y) is an optimal solution of $(P(u))$ for some $u \ge 0$ and $x(E) = k$, then it is also an optimal solution of (P). We will show that such (x, y) and u always exist for (P) and develop a combinatorial algorithm to find them in $O(n^2|K|)$ time.

First, we develop an algorithm for solving $(P(u))$. A slightly more generalized version of $(P(u))$ has been dealt with by Myung [6]. Our algorithm is almost the same as their algorithm but we describe the algorithm for the completeness of the paper. Our algorithm first obtains a dual solution for the linear programming (LP) relaxation of $(P(u))$, then we construct a (integer) feasible solution of $(P(u))$ based on the obtained dual solution. We will show that our algorithm generates an optimal solution. When considering the LP relaxation of $(P(u))$, we omit the constant term uk from the objective function and replace (4) and (5) by $x(e) \ge 0$ and $y(i) \ge 0$, respectively. Then the dual of this LP relaxation is as follows.

$$(D(u)) \qquad \max \sum_{i \in K} f(i)$$

$$s.t. \quad \sum_{i: e \in E_i} f(i) \le u, \qquad e \in E \tag{6}$$

$$f(i) \le d(i), \qquad i \in K \tag{7}$$

$$f(i) \ge 0, \qquad i \in K \tag{8}$$

If we think $f(i)$ as the flow from s_i to t_i, $(D(u))$ may be viewed as the maximum multicommodity flow problem on a path network where the capacity of each edge is u and the flow of commodity i (between s_i and t_i) cannot exceed $d(i)$. Our algorithm simply routes as much flow as possible and we will show that this simple strategy generates optimal flows when we set the flow of each commodity in such an order that a commodity with the smaller indexed sink node comes first. Next, we determine a set of edges to be selected. Let F denote such a set. Given a dual solution f, we say that an edge e is *saturated* if $\sum_{i:e\in E_i} f(i) = u$. We initially include all saturated edges in F, then we perform a delete step to remove redundant edges. After finishing the delete step, we set (x, y) as follows: $x(e) = 1$ if $e \in F$ and $x(e) = 0$, otherwise; $y(i) = \max\{1-x(E_i), 0\}$ for each $i \in K$. We formally describe the algorithm for solving $(P(u))$ as follows:

Algorithm I

(Step 1) [Initialization] $f \leftarrow \mathbf{0}$ and $F \leftarrow \varnothing$.

(Step 2) [Flow routing] For each $i \in K$ in nondecreasing order of the indexes of the sink node, increase $f(i)$ until either $f(i)$ reaches $d(i)$ or at least one edge in E_i is saturated. Include all newly saturated edges into F.

(Step 3) [Edge selection] For each edge $e \in F$ in decreasing order, do the following operation: Unless there exists at least one source-sink pair i with $f(i) < d(i)$ such that e is a unique saturated edge in the path between the source and sink nodes, delete e from F.

(Step 4) [Set (x, y)] Set $x(e) \leftarrow 1$ if $e \in F$, and set $x(e) \leftarrow 0$ otherwise. Set $y(i) \leftarrow \max\{1-x(E_i), 0\}$ for each $i \in K$.

Now we prove the correctness of Algorithm I. As (x, y) is determined by $F \subseteq E$, we will use (x, y) and F alternatively to describe a solution of $(P(u))$.

Theorem 1: Algorithm I optimally solves $(P(u))$ in $O(n|K|)$ time.

Proof:

Let f and (x, y) be the optimal solution obtained through Algorithm I. By the nature of Steps 2 and 4, f and (x, y) are feasible to $(D(u))$ and $(P(u))$, respectively. Therefore, each solution is optimal to the corresponding problem if both solutions satisfy

the following complementary slackness conditions between the LP relaxation of ($P(u)$) and ($D(u)$), i.e.:

(C1) $(x(E_i)+y(i)-1)f(i) = 0$, for each $i \in K$.
(C2) $(u - \sum_{i:e\in E_i} f(i))x(e) = 0$, for each $e \in F$.
(C3) $(d(i)-f(i))y(i) = 0$, for each $i \in K$.

Since F contains only the saturated edges, (C2) trivially holds. To prove (C3), we show that if $f(i) < d(i)$, then $y(i) = 0$. Suppose that $f(i) < d(i)$. Then there must have been at least one saturated edge in E_i after completing the iteration for i in Step 2, and at least one such edge should remain in F through the delete process of Step 3. From Step 4, it means that $x(E_i) \geq 1$ and $y(i) = 0$. Finally, we prove that the solution satisfies (C1). We need to show that if $f(i) > 0$, then $x(E_i)+y(i) = 1$. As $x(E_i)+y(i) \geq 1$, it is sufficient to show that it never happens that $f(i) > 0$ and $x(E_i)+y(i) > 1$. By Step 4, $x(E_i)+y(i) > 1$ means $x(E_i) > 1$, i.e., $|E_i \cap F| > 1$. Let e_1 and e_2 be two edges in $E_i \cap F$ with $e_1 < e_2$. As these two edges are not removed during Step 3, there must be two distinct source-sink pairs, i_1 and i_2 in K such that $f(i_1) < d(i_1)$, $E_{i_1} \cap F = \{e_1\}$, $f(i_2) < d(i_2)$, $E_{i_2} \cap F = \{e_2\}$. As $t_{i_1} < t_i$, the flow of i_1 must be routed before i in Step 2 and there must be a saturated edge $e' \in E_{i_1}$ such that $e' \notin E_i$ because $f(i_1) < d(i_1)$, $f(i) < d(i)$ and $f(i) > 0$. As $e' < e_1$, Step 3 checks e_1 before e'. This contradicts the fact that e_1 is not removed during Step 3. So f and (x, y) satisfy (C3). Finally, each of the steps in Algorithm I can be carried out within $O(n|K|)$. □

The second part of our algorithm is to find $u \geq 0$ such that the optimal solution (x,y) of ($P(u)$) satisfies $x(E) = k$. We first show that such a solution always exists.

Theorem 2: There exists $u \geq 0$ such that the optimal solution F of ($P(u)$) satisfies $|F| = k$.

Proof:

It is well known that $z(u)$ is a piecewise linear concave function with respect to u. It is easy to see that $z(u)$ has at most $|E|$ line segments and the algorithm generates a unique solution F for each line segment. As we assumed positive d, E is an optimal solution of ($P(0)$) with $z(0) = 0$. Suppose that there exists an optimal solution F of ($P(0)$)

such that $|F| \le k$. Since adding an arbitrary edge to F never increases $z(0)$, we can always find an optimal F of $(P(0))$ such that $|F| = k$. Suppose that every optimal solution F of $(P(0))$ satisfies $|F| > k$. Since $z(u)$ is piecewise linear concave and $(P(\infty))$ will have an optimal solution F with $F \le k$, there exists some $u > 0$ such that either an optimal solution F of $(P(u))$ satisfies $|F| = 0$ or two solutions F^1 and F^2 with $|F^1| < k < |F^2|$ are optimal to $(P(u))$. It is sufficient to show that even in the latter case, we can find an optimal solution F of $(P(u))$ satisfying $|F| = k$. We will show that we can construct such a solution using F^1 and F^2. Suppose that $F^1 = \{e_1, \cdots, e_p\}$ and $F^2 = \{\hat{e}_1, \cdots, \hat{e}_t\}$ are optimal to $(P(u))$ and $p < k < t$. We claim that if F^2 contains more than one edge that are less than or equal to e_1, we can delete such edges except the largest one without increasing the objective value. For example, suppose that $\hat{e}_1 < \cdots < \hat{e}_{s-1} < \hat{e}_s \le e_1 < \hat{e}_{s+1}$, then for any subset $E' \subseteq \{\hat{e}_1, \cdots, \hat{e}_{s-1}\}$, $F^2 \setminus E'$ must be also optimal to $(P(u))$. This is because otherwise, adding E' to F^1 would decrease the objective value contradicting the optimality of F^1. In a same way, we can show that if F^2 contains more than one edge that are greater than or equal to e_p, we can delete all such edges except the smallest one without increasing the objective value.

So, it is sufficient to consider the case where $\hat{e}_1 \le e_1 < \hat{e}_2$ and $\hat{e}_{t-1} < e_p \le \hat{e}_t$. Since $p+2 \le t$, there must exist $\hat{e}_j \in F^2$ such that $e_i \le \hat{e}_j < \hat{e}_{j+1} \le e_{i+1}$ and $|\{\hat{e}_1, \cdots, \hat{e}_j, e_{i+1}, \cdots, e_p\}| = k$. Let $F^{11} = \{e_1, \cdots, e_i\}$, $F^{12} = \{e_{i+1}, \cdots, e_p\}$, $F^{21} = \{\hat{e}_1, \cdots, \hat{e}_j\}$, and $F^{22} = \{\hat{e}_{j+1}, \cdots, \hat{e}_t\}$. Notice that $F^1 = F^{11} \cup F^{12}$, $F^2 = F^{21} \cup F^{22}$, and $|F^{21} \cup F^{12}| = k$. We prove that $F^{21} \cup F^{12}$ is also optimal to $(P(u))$. Let $v(F)$ denote the objective value of $(P(u))$ with respect to F. Since $d(\text{conk}(F^{21} \cup F^{12})) + d(\text{conk}(F^{11} \cup F^{22})) \le d(\text{conk}(F^1)) + d(\text{conk}(F^2))$, the following relation holds.

$$\begin{aligned} v(F^{21} \cup F^{12}) + v(F^{11} \cup F^{22}) &= d(\text{conk}(F^{21} \cup F^{12})) + d(\text{conk}(F^{11} \cup F^{22})) \\ &\quad + u(|F^{21} \cup F^{12}| + |F^{11} \cup F^{22}|) - 2uk \\ &\le d(\text{conk}(F^1)) + d(\text{conk}(F^2)) + u(|F^1| + |F^2|) - 2uk \\ &= v(F^1) + v(F^2) \end{aligned}$$

Since F^1 and F^2 are optimal to $(P(u))$, so are $F^{21} \cup F^{12}$ and $F^{11} \cup F^{22}$. □

From Theorem 1 and 2, we can see that the LP relaxation of (P) on a path net-

work gives an integer solution. One may suspect that the LP relaxation of (P) on the original ring network also gives an integer solution but the following example indicates that this is not true for a ring network. Consider a ring network with 4 nodes with $K = \{(1, 3), (2, 4)\}$. Assume that $d(1) = d(2) = 1$ and $k = 1$, then the optimal objective values of (P) and its LP relaxation are 2 and 1, respectively, because no source-sink pair can be disconnected in (P) while $y(1) = y(2) = 0.5$ in the optimal solution of the LP.

The proof of Theorem 2 shows how to find an optimal solution of (P). Our algorithm solves ($P(u)$) at each break point of $z(u)$ until it finds an optimal solution F with $|F| = k$ or a pair of optimal solutions F^1 and F^2 with $|F^1| < k < |F^2|$ for some break point $u \geq 0$. In the latter case, we can construct an optimal solution including k edges using F^1 and F^2 as we did in the proof of Theorem 2. So, we have proved the following result.

Theorem 3: (k-SUV) in a ring network can be solved in $O(n^3|K|)$.

References

[1] 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.

[2] Frank, A., T. Nishizeki, N. Saito, H. Suzuki, and E. Tardos, "Algorithms for routing around a rectangle," *Discrete Applied Mathematics* 40 (1992), 363-378.

[3] Garg, N., V. V. Vazirani, and M. Yannakakis, "Primal-dual approximation algorithms for integral flow and milticut in trees," *Algorithmica* 18 (1997), 3-20.

[4] Kolar, D. J. and T. Wu, "A study of survivability versus cost for several fiber network architectures," *Proceedings of IEEE Int'l Conference on Communications* (1988), 61-66.

[5] Myung, Y.-S., "Multicommodity flows in cycle graphs," *Discrete Applied Mathematics* 154 (2006), 1615-1621.

[6] Myung, Y.-S., "The server disconnection problems on a a ring network," *Journal of KIIE* 35 (2009), 87-91.

[7] Myung, Y.-S. and H.-J. Kim, "A Cutting Plane Algorithm for Computing k-edge

Survivability of a Network," *European Journal of Operational Research* 156 (2004), 579-589.

[8] Myung, Y.-S. and H.-J. Kim, "Network disconnection problems in a centralized network," *Naval Research Logistics* 54 (2007), 710-719.

[9] Myung, Y.-S., H.-G. Kim, and D.-W. Tcha, "Optimal load balancing on SONET bidirectional rings," *Operations Research* 45 (1997), 148-152.

[10] Myung, Y.-S., H.-J. Kim, and D.-W. Tcha, "Design of communication networks with survivability constraints," *Management Science* 45 (1999), 238-252.

[11] Schrijver, A., P. Seymour and P. Winkler, The ring loading problem, SIAM J. Discrete Math., 11 (1998), 1-14.

[12] Suzuki, H., A. Ishiguro, and T. Nishizeki, "Variable-priority queue and doughnut routing," *Journal of Algorithms* 13 (1992), 606-635.

[13] Wu, T., Fiber network survivability, Artech House, Inc. Boston, 1992.

[14] Wu, T., R. H. Cardwell, and W. E. Woodall, "Decreasing survivable fiber network cost using optical switches," Proceeding of IEEE Int'l Conference on Communications, (1988), 93-97.

[15] Wu, T., D. J. Kolar, and R. H. Cardwell, "Survivable network architectures for broad-band fiber optic networks: model and performance comparison," *Journal of Lightwave Technology* 6 (1988), 1698-1709.