Two-Server Network Disconnection Problem*

Byung-Cheon Choi and Sung-Pil Hong**

Department of Industrial Engineering, College of Engineering,
Seoul National University, San 56-1, Shillim-9-Dong, Seoul, 151-742, Korea
sphong@snu.ac.kr

Abstract. Consider a set of users and servers connected by a network. Each server provides a unique service which is of certain benefit to each user. Now comes an attacker, who wishes to destroy a set of edges of the network in the fashion that maximizes his net gain, namely, the total disconnected benefit of users minus the total edge-destruction cost. We first discuss that the problem is polynomially solvable in the single-server case. In the multiple-server case, we will show, the problem is, however, NP-hard. In particular, when there are only two servers, the network disconnection problem becomes intractable. Then a $\frac{3}{2}$-approximation algorithm is developed for the two-server case.

1 Introduction

Consider a network of servers and their users in which each server provides a unique service to the users. (Each server also can be the user of a service other than her own.) Each user takes a benefit through the connection provided by the network to each server. Now comes an attacker who wishes to destroy a set of edges in the manner that optimizes his own objective, although defined to various circumstances, that explicitly accounts for the disconnected benefits of users resulted from destruction.

Such a model, to the author's best knowledge, was proposed first by Martel et al. (8). They considered the single-server problem in which the total disconnected benefit is maximized under an edge-destruction budget constraint. The problem, as they showed, is NP-hard. They also proposed an exact method enumerating maximum disconnected node sets among minimum cost cuts separating node pairs using cut submodularity.

In this paper, we consider the problem of maximizing net gain, namely, the total disconnected benefit of users minus the edge-destroying cost. The problem is polynomially solvable when there is a single server. It is, however, NP-hard in general. We will provide the proofs. In particular, if there are two servers, the problem becomes intractable. Also, we will present a $\frac{3}{2}$-approximation algorithm for the two-server case.

* The research was partially supported by KOSEF research fund R01-2005-000-10271-0.

** Corresponding author.

M. Gavrilova et al. (Eds.): ICCSA 2006, LNCS 3982, pp. 785–792, 2006.

In Section 2, we review the previous models on separating nodes of a graph. Section 3 provides a mathematical model of the problem. It also discusses the polynomiality of the single-server case and NP-hardness for the k-server case with $k \geq 2$. A $\frac{3}{2}$-approximation algorithm is presented in Section 4. Finally, we summarize the results and point out an open problem in Section 5.

2 Node Separation Problems: A Literature Review

In this section we review combinatorial optimization models on the separation of nodes of an undirected graph. There are various such models. (See, e.g. (1; 5).) Among them, the following three models probably have been studied most intensively.

Problem 1. **k-cut problem:** Given an undirected $G = (N, E)$ with nonnegative edge weights, find a minimum weight set of edges E' such that the removal of E' from E separates graph into exactly k nonempty components.

This problem is NP-hard for general $k \geq 3$ and approximable within factor $2 - \frac{2}{k}$ within the optimum. Goldscmidt and Hochbaum (7) showed that the problem is polynomially solvable for fixed k.

Problem 2. **k-terminal cut problem, Multiway cut problem:** Given an undirected $G = (N, E)$ with nonnegative edge weights and a set of k specified nodes, or *terminals*, find a minimum weight set of edges E' such that the removal of E' from E disconnects each terminal from all the others.

k-terminal cut problem is Max-SNP-hard even for $k = 3$ (4). Therefore, there is some $\epsilon > 0$ such that $(1 + \epsilon)$-approximation is NP-hard. Naor and Zosin(9) considered the directed version of k multi-terminal cut problem, and presented two 2-approximation algorithms. The current best approximation guarantee is $\frac{3}{2} - \frac{1}{k}$ by Călinescu et al. (3).

	k-cut problem	k-terminal cut problem	multicut problem
$k = 2$	P[a]	P[b]	P(12)
$k \geq 3$	P(2; 7)	Max-SNP-hard[cd](4)	Max-SNP-hard[e]
arbitrary	NP-hard[f](7)	Max-SNP-hard[gh](4; 3)	Max-SNP-hard[i]

[a] P means "polynomially solvable"
[b] $Max - SNP$-hard if there is a constraint on the size of the separated component
[c] polynomially solvable if the graph is planar
[d] NP-hard even if all edge weights are equal to 1
[e] NP-hard by the reduction from 3-terminal cut problem
[f] There is a $(2 - \frac{2}{k})$-approximation algorithm
[g] There is a $(\frac{3}{2} - \frac{1}{k})$-approximation algorithm
[h] NP-hard even if the graph is planar and all edge weights are equal to 1
[i] approximable within $O(\log k)$.

Problem 3. **Multicut problem:** Given an undirected $G = (N, E)$ with nonnegative edge weights and k pairs of nodes, find a minimum weight set of edges E' such that the removal of E' from E separates all pairs of nodes.

This problem is a generalization of the k-terminal cut problem. Hence, it also becomes Max-SNP-hard when $k = 3$. Currently, this problem is approximable within the factor of $O(\log k)$ (6).

These works can be summarized as above text table.

In the NP-hardness proof of the k-server network disconnection problem, we reduce the $(k+1)$-terminal problem to the k-server network disconnection problem.

3 The *k*-Server Network Disconnection Problem

3.1 Problem Formulation

Given an undirected graph $G = (N, E)$ with $N = \{1, 2, \ldots, n\}$, the server set $S = \{s_1, \cdots, s_k\} \subseteq N$, the costs $c_{ij} \geq 0$, $(i, j) \in E$, the nonnegative vectors $d_i = (d_i^1, \cdots, d_i^k)^T$, $i \in N$, find a set $F \subseteq E$ that maximizes the total disconnected benefits of nodes minus the edge-destruction cost, $\sum_{(i,j)\in F} c_{ij}$.

Let N_l be the set of nodes remaining connected from server l after the destruction of F. Then, it is easy to see that for any $i \neq j$, the two sets, N_i and N_j are either identical or mutually exclusive: $N_i = N_j$ or $N_i \cap N_j = \emptyset$. Hence, if we denote by N_0 the nodes disconnected from all the servers, then the set $\mathcal{N} = \{N_0, N_1, \ldots, N_k\}$ is a partition of N. If $\mathcal{N}$ has p distinct sets, we will call $\mathcal{N}$ a *p-partition.* Our problem can be restated as follows:

Problem 1. ***k*-server network disconnection problem:** Find a partition $\mathcal{N} = \{N_0, N_1, \ldots, N_k\}$ with $s_j \in N_j$, $j = 1, 2, \ldots, n$ that maximizes

$$z(\mathcal{N}) = \sum_{l:N_l\in\mathcal{N}} \sum_{i\in N_l} \sum_{j\notin N_l} d_i^j - \sum_{\substack{(i,j)\in E: i\in N_p,\ j\in N_q \\ \text{such that } N_p\neq N_q}} c_{ij}. \tag{1}$$

3.2 Polynomiality of the Single-Server Case

We need to find $N_1 \subseteq N$ with $s_1 \in N_1$ so that $\mathcal{N} = \{N \setminus N_1, N_1\}$ maximizes (1). Define a binary variable x as follows:

$$x_j = \begin{cases} 1, \text{ if } j \in N_1, \\ 0, \text{ otherwise.} \end{cases}$$

For notational convenience we assume $s_1 = 1$, $c_{ij} = c_{ji}$ for all $i, j \in N$ and $c_{ij} = 0$ for $(i, j) \notin E$. Then we can formulate the single-server problem as a 0-1 quadratic program as follows.

$$\begin{gathered} \max z(x) = \textstyle\sum_{j=1}^n d_j^1(1 - x_j) - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n c_{ij}(x_i - x_j)^2 \\ \text{sub. to } \ x_1 = 1,\ x_j \in \{0, 1\},\ j \in N \setminus \{1\}. \end{gathered} \tag{2}$$

Using $x_j = x_j^2$, it is easy to rewrite the objective of (2) as a quadratic function of $\hat{x} = (x_2, x_3, \ldots, x_n)$: $z(\hat{x}) = \hat{x}^T Q \hat{x} + \sum_{j=2}^n (d_j^1 - c_{1j})$, where $Q = (q_{ij})_{\substack{i=2,\ldots,n \\ j=2,\ldots,n}}$ is given as follows:

$$q_{ij} = \begin{cases} -d_i^1 + c_{i1} - \sum_{k=2}^{n} c_{ik}, & \text{if } i = j, \\ c_{ij}, & \text{otherwise.} \end{cases} \tag{3}$$

Lemma 1. *An unconstrained 0-1 quadratic maximization with nonnegative off-diagonal elements is polynomially solvable.*

Proof. See Picard and Ratliff (10). □

Theorem 1. *The single-server network disconnection problem is polynomially solvable.*

Proof. Since $c_{ij} \geq 0$, $(i, j) \in E$, the polynomiality of the quadratic program formulation (2), follows from (3) and Lemma 1. □

3.3 *NP*-Hardness of the *k*-Server Case

Unlike the single-server case, our problem is NP-hard in general. When $k = 2$, in particular, it becomes NP-hard. To see this, consider the decision version of the 3-terminal cut problem, Problem 2.

Problem 2. **Decision version of 3-terminal cut problem(3DMC)** Given a terminal set $T = \{t_1, t_2, t_3\}$, a weight $w_{ij} \geq 0$, $(i, j) \in E$ and a constant W, is there a partition $\mathcal{N} = (N_1, N_2, N_3)$ such that $t_j \in N_j$, $j = 1, 2, 3$, and

$$w(\mathcal{N}) = \sum_{\substack{(i,j)\in E \mid i\in N_p,\ j\in N_q \\ \text{such that } N_p \neq N_q}} w_{ij} \leq W?$$

Theorem 2. *The 2-server network disconnection problem is NP-hard.*

Proof. Given any instance of 3DMC, we can construct an instance of the 2-server problem as follows : On the same graph $G = (N, E)$, designate the first two terminals of 3DMC as the two server nodes, $s_1 = t_1$ and $s_2 = t_2$ while t_3 is a non-server node in the 2-server instance. Now, define the benefit vector for the node of the 2-server instance: For a constant $M > W$, set $d_{s_1} = (d_{s_1}^1, d_{s_1}^2) = (0, M)$, $d_{s_2} = (M, 0)$, and $d_{t_3} = (M, M)$. The other nodes are assigned to the benefit vector, $(0, 0)$. Also set $Z = 4M - W$. Finally, the edge cost is defined as $c_{ij} = w_{ij}$, $(i, j) \in E$. We claim that the answer to 3DMC is 'yes' if and only if the 2-server instance has a partition whose value is no less than Z.

Suppose there is a partition $\hat{\mathcal{N}} = \{\hat{N}_1, \hat{N}_2, \hat{N}_3\}$ of 3DMC satisfying $w(\hat{\mathcal{N}}) \leq W$. Then, it is easy to see that $\mathcal{N} = \{\hat{N}_3, \hat{N}_1, \hat{N}_2\}$ is a solution of the 2-server instance: $s_j \in \hat{N}_j$, $j = 1, 2$, and

$$z(\hat{\mathcal{N}}) = \sum_{l=1}^{3} \sum_{i\in\hat{N}_l} \sum_{j\notin\hat{N}_l} d_i^j - \sum_{\substack{(i,j)\in E: i\in\hat{N}_p,\ j\in\hat{N}_q \\ \text{such that } \hat{N}_p \neq \hat{N}_q}} w_{ij} \geq 4M - W = Z.$$

On the other hand, assume $\tilde{\mathcal{N}} = \{\tilde{N}_0, \tilde{N}_1, \tilde{N}_2\}$ is a solution of the 2-server instance such that $z(\tilde{\mathcal{N}}) \geq Z$. To show that $\tilde{\mathcal{N}}$ is a 3-terminal cut, it suffices to

see that $\tilde{N}_j$, $j = 0, 1, 2$ are all pair-wise distinct. But, if any two of the sets are identical, then $z(\tilde{x}) \leq 3M$, a contradiction to the assumption. Furthermore, this also implies

$$z(\tilde{\mathcal{N}}) = 4M - w(\tilde{x}) \geq Z = 4M - W.$$

Since $4M - w(\tilde{\mathcal{N}}) \geq Z = 4M - W$, we get $w(\tilde{\mathcal{N}}) \leq W$. □

It is not hard to see that the proof can be extended to show that the $(k+1)$-terminal cut problem is a special case of the k-server network disconnection problem.

4 A $\frac{3}{2}$-Approximation of the 2-Server Case

Let $\mathcal{N} = \{N_0, N_1, N_2\}$ be a solution of the 2-server case such that $s_1 \in N_1$ and $s_2 \in N_2$. Define

$$E_0 = \{(i,j) \in E \mid i \in N_1 \text{ and } j \in N_2\}$$
$$E_1 = \{(i,j) \in E \mid i \in N_1 \text{ and } j \in N_0\}, \text{ and}$$
$$E_2 = \{(i,j) \in E \mid i \in N_2 \text{ and } j \in N_0\}.$$

Recall that it may be either $N_1 = N_2$ or $N_0 = \emptyset$ and thus an optimal solution

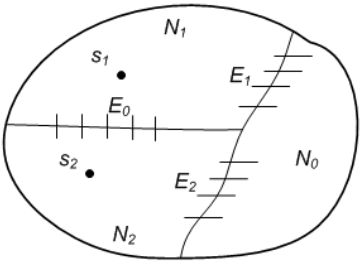

Fig. 1. 2-server solution

may be 1-, 2-, or 3-partition. If it is trivially a 1-partition, then the optimal objective value is 0. The idea is to approximate the optimum with an optimal 2-partition which is, as we will show, polynomially computable.
Algorithm H : Find an optimal 2-partition as an approximation of the optimum.

Lemma 1. *An optimal 2-partition can be computed in polynomial time.*

Proof. There are two cases of an optimal 2-partition: $N_1 = N_2$ or $N_1 \neq N_2$.

Case 1: N_1 and N_2 are distinct
Then we can formulate the 2-server problem as a quadratic program similarly to the single-server case. Define a binary variable x as follows:

$$x_j = \begin{cases} 1, & \text{if } j \in N_1, \\ 0, & \text{if } j \in N_2. \end{cases}$$

As before, we adopt the notation, $s_1 = 1$, $s_2 = 2$, $c_{ij} = c_{ji}$ for all $i, j \in N$ and $c_{ij} = 0$, $(i, j) \notin E$. Then, it is easy to see that the 2-server case is equivalent to

$$\begin{gathered} \max \;\; z(x) = \textstyle\sum_{j=1}^{n} \left(d_j^2 x_j + d_j^1 (1 - x_j)\right) - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} (x_i - x_j)^2 \qquad (4) \\ \text{sub. to } \; x_1 = 1, \; x_2 = 0, \\ x_j \in \{0, 1\}, \; j \in N \setminus \{1, 2\}. \end{gathered}$$

Then we can rewrite (4) in terms of $\hat{x} = (x_3, x_4, \ldots, x_n)^T$: $z(\hat{x}) = \hat{x}^T Q \hat{x} + \sum_{j=1}^{n} d_j^1 + d_1^2 + d_2^1 - 2c_{12}$, where $Q = (q_{ij})_{\substack{i=3,\ldots,n \\ j=3,\ldots,n}}$ is given as follows:

$$q_{ij} = \begin{cases} d_i^1 - d_i^2 + c_{i1} - c_{i2} - \sum_{k=1}^{n} c_{ik}, & \text{if } i = j, \\ c_{ij}, & \text{otherwise.} \end{cases} \qquad (5)$$

Since c_{ij}, $(i, j) \in E$ are nonnegative, $z(\hat{x})$ can be solved in polynomial time from Lemma 1.

Case 2: N_1 and N_2 are identical
In this case, the problem is essentially a single-server problem. Merge s_1 and s_2 into a single node and replace benefit vector (d_i^1, d_i^2) by a single benefit $d_i^1 + d_i^2$. Then, it is easy to see that the 2-server case is equivalent to the single-server case defined on the modified network, implying that the 2-server case can be solved in polynomial time. Case 1 and 2 complete proof. □

Theorem 1. *An optimal 2-partition is factor $\frac{3}{2}$ within the optimum.*

Proof. Let $\mathcal{N}^* = \{N_0^*, N_1^*, N_2^*\}$ and $\mathcal{N}^H = \{N_0^H, N_1^H, N_2^H\}$ be an optimal solution and the solution of *Algorithm H*, respectively.

$$z(\mathcal{N}^*) = \sum_{l=0}^{2} \sum_{i \in N_l^*} \left(\sum_{j=1}^{2} d_i^j - d_i^l\right) - \sum_{(i,j) \in E_0^* \cup E_1^* \cup E_2^*} c_{ij}$$

$$= \frac{1}{2}\Big(\Big(\sum_{i \in N_0^* \cup N_1^*} d_i^2 + \sum_{i \in N_2^*} d_i^1 - \sum_{(i,j) \in E_0^* \cup E_2^*} c_{ij}\Big) + \Big(\sum_{i \in N_0^* \cup N_2^*} d_i^1 + \sum_{i \in N_1^*} d_i^2 - \sum_{(i,j) \in E_0^* \cup E_1^*} c_{ij}\Big)$$

$$+ \Big(\sum_{i \in N_0^*} (d_i^1 + d_i^2) - \sum_{(i,j) \in E_1^* \cup E_2^*} c_{ij}\Big)\Big)$$

$$= \frac{1}{2}\left(z(N_0^*, N_1^* \cup N_2^*) + z(N_1^*, N_0^* \cup N_2^*) + z(N_2^*, N_0^* \cup N_1^*)\right).$$

Since $z(\mathcal{N}^H) \geq \max\{z(N_0^*; N_1^* \cup N_2^*),\ z(N_1^*; N_0^* \cup N_2^*),\ z(N_2^*; N_0^* \cup N_1^*)\}$,

$$z(\mathcal{N}^*) \leq \frac{3}{2} z(\mathcal{N}^H).$$

This completes the proof. □

Theorem 2. *Algorithm H is a $\frac{3}{2}$-approximation algorithm for the 2-server case.*

Proof. From Lemma 1 and Theorem 1. □

5 Summary and Further Research

We consider the k-server network disconnection problem. We show that the problem can be solved in polynomial time for the single-server case while the problem is NP-hard in general. The problem is NP-hard even for the two-server case. Also, we propose a $\frac{3}{2}$-approximation algorithm for two-server case.

The approximability of the k-server network disconnection problem for general $k \geq 0$ remains open. The quadratic program based approximation algorithm for the 2-server case does not seem to straightforwardly extend to the general case.

Acknowledgment

The authors thank Prof. Young-Soo Myung for the fruitful discussion which leads to a tighter analysis of the algorithm.

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation,* Springer-Verlag, Berlin, 1999.

[2] M. Burlet, and O. Goldschmidt. *A new and improved algorithm for the 3-cut problem.* Operations Research Letters 21(1997) pp. 225-227.

[3] G. Călinescu, H. Karloff, and Y. Rabani. *An improved approximation algorithm for muliway cut,* Proc. 30th ACM Symposium on Theory of Computing, ACM, (1998) pp. 48-52.

[4] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. *The complexity of multiterminal cuts,* SIAM Journal on Computing 23(1994) pp. 864-894.

[5] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness,* W.H. Freeman, New York, 1979.

[6] N. Garg, V.V. Vazirani, and M. Yanakakis. *Approximating max-flow min-(multi)cut theorems and their applications,* SIAM Journal on Computing 25(1996) pp. 235-251.

[7] O. Goldschmidt, and D.S. Hochbaum. *A polynomial time algorithm for the k-cut problem for k fixed,* Mathematics of Operations Research 19(1994) pp. 24-37.

[8] C. Martel, G. Nuckolls, and D. Sniegowski. *Computing the disconnectivity of a graph,* manuscript, UC Davis (2001).

[9] J. Maor, and L. Zosin. *A 2-approximation algorithm for the directed multiway cut problem,* SIAM Journal on Computing 31(2001) pp. 477-482.

[10] J.C. Picard, and H.D. Ratliff. *Minimum cuts and related problems,* Networks 5(1974) pp. 357-370.

[11] H. Saran, and V.V. Vazirani. *Finding k-cuts within twice the optimal,* Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, (1991) pp. 743-751.

[12] M. Yannakakis, P.C. Kanellakis, S.C. Cosmadakis, and C.H. Papadmitriou. *Cutting and partitioning a graph after a fixed pattern, in Automata, Language and Programming,* Lecture Notes and Computer Science 154(1983) pp. 712-722.