The Complexity of Finding Most Vital Arcs and Nodes

Amotz Bar-Noy
IBM Research Division
Thomas J. Watson Research Center
Yorktown Heights, NY 10598

Samir Khuller *
Dept. of Computer Science
University of Maryland
College Park, MD 20742

Baruch Schieber
IBM Research Division
Thomas J. Watson Research Center
Yorktown Heights, NY 10598

November 29, 1995

Abstract

Let $G(V, E)$ be a graph (either directed or undirected) with a non-negative length $\ell(e)$ associated with each arc e in E. For two specified nodes s and t in V, the k most vital arcs (or nodes) are those k arcs (nodes) whose removal maximizes the increase in the length of the shortest path from s to t. We prove that finding the k most vital arcs (or nodes) is NP-hard, even when all arcs have unit length. We also correct some errors in an earlier paper by Malik, Mittal and Gupta [ORL 8:223-227, 1989].

Keywords: networks, graphs, NP-Complete, vital arcs.

1. Introduction

The Most Vital Arcs Problem (MVAP) is defined as follows.

Input: A graph $G = (V, E)$ (either directed or undirected) with a non-negative length $\ell(e)$ associated with each arc e in E, two specified nodes s and t in V, and a positive integer k.

*Research supported by NSF Research Initiation Award CCR-9307462 and an NSF CAREER Award CCR-9501355.

Output: A set of k arcs whose removal maximizes the increase in the length of the shortest path from s to t. These arcs are the k *most vital* arcs (with respect to s and t).

In the corresponding Most Vital Arcs *Decision* Problem (Decision-MVAP) the input includes a threshold h, and the output is "yes" if there are k arcs whose removal makes the length of the shortest path from s to t at least h.

Similarly, the Most Vital Nodes Problem (MVNP), and the corresponding decision problem (Decision-MVNP) are defined by replacing arcs by nodes.

Our main result is proving that both decision problems are *strongly* NP-Complete for either directed or undirected graphs. (A problem is *strongly* NP-Complete if it is NP-Complete even when all the integers involved are restricted to be at most polynomially large in the size of the input.) For this, we prove that both problems are NP-Complete, even when all arcs have unit length. Clearly, this implies that MVAP and MVNP are *strongly* NP-Hard.

The MVAP and MVNP were defined and motivated by Corley and Sha [2]. They gave some preliminary results including a polynomial time algorithm for the single most vital arc problem (i.e., the case $k = 1$). Ball, Golden, and Vohra [1] considered a generalization of the MVAP in which associated with each arc e is a *cost* $c(e)$ of removing it, and the goal is to find the set of arcs with total cost not exceeding a given budget whose removal maximizes the increase in the length of the shortest path from s to t. (In our setting $c(e) = 1$ for all arcs $e \in E$, and the budget is k.) They proved that this general problem is NP-Hard.

Malik, Mittal, and Gupta [4] described an exponential time algorithm for MVAP with arbitrary k. However, this algorithm seems to be fallacious as shown in Section 3. They also proposed an efficient algorithm for the single most vital arc problem for undirected graphs. However the proof of this algorithm has an error. In Section 4, we give a correct proof of their algorithm.

2. The NP-Completeness proof

In this section we prove that the Decision-MVAP and Decision-MVNP are NP-Complete even when all arcs have unit length. First, we give the proof for the Most Vital Arcs Decision Problem in undirected graphs. We then extend the proof to directed graphs and for the Decision-MVNP.

The proof is by reducing the Node Cover Decision Problem (Decision-NCP) to the Decision-MVAP for undirected graphs with unit length arcs. The Decision-NCP is defined as follows (see, e.g., [3]).

Input: An undirected graph $G = (V, E)$, and a positive integer c.

Question: Is there a node cover for G of size c or less, i.e., is there a subset $U \subseteq V$ with $|U| \leq c$ such that for each arc $(u, v) \in E$ at least one of u and v belongs to U?

Suppose that we are given an instance for the Decision-NCP, consisting of a graph $G = (V, E)$ and a parameter c. We show a polynomial time reduction to an instance of the Decision-MVAP, such that the answer for the instance of the Decision-MVAP is "yes" if and only if the answer for the corresponding instance of Decision-NCP is also "yes".

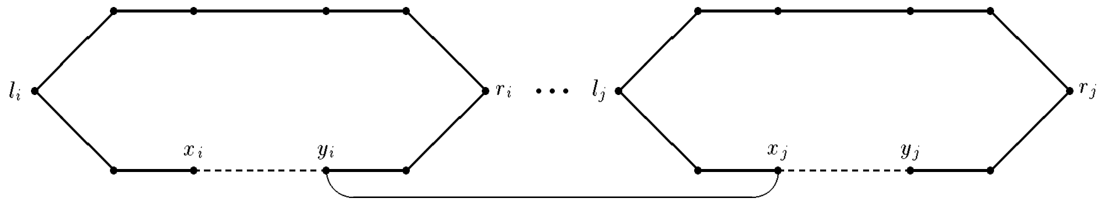

Figure 1: The reduction.

The input to the Decision-MVAP consists of: a graph $G' = (V', E')$, two specified nodes s and t, a positive integer k, and a threshold h. We start with the definition of the graph $G' = (V', E')$. Let $|V| = n$ and assume that the nodes in V are labelled $1, 2, \ldots, n$. The graph G' consists of one "gadget" per each node $i \in V$. Each gadget consists of two parallel paths of length five with common endpoints. See Figure 1 for the gadgets corresponding to nodes i and j in V. The right endpoint of gadget $i < n$, is the same as the left endpoint of gadget $i + 1$. We refer to the two paths in each gadget as the *upper* portion and the *lower* portion of the gadget.

We identify four nodes in each gadget i: l_i – the left endpoint of gadget i, r_i – the right endpoint of gadget i, x_i and y_i – the left and right endpoints of the third arc in the lower portion of gadget i (See Figure 1.).

For each arc $(i, j) \in E$, $i < j$, we add to G' a path of length $5(j - i) - 2$ from node y_i to node x_j. (See Figure 1.) We refer to this path as the *shortcut* (i, j).

We assume that the graph consists of two types of arcs: *removable* arcs and *fixed* arcs. The fixed arcs are those arcs that cannot be removed from the graph in any solution to the Decision-MVAP. Later, we show how to modify the graph so that this assumption is relaxed and all arcs are removable. The only removable arc in each gadget i is arc (x_i, y_i) in the lower portion (the dashed arcs in Figure 1).

This completes the definition of $G' = (V', E')$. Note that the size of G' is at most quadratic in the size of G.

The rest of the input to the Decision-MVAP is defined as follows. Node $s \in V'$ is the left endpoint of gadget 1 (l_1), and node $t \in V'$ is the right endpoint of the gadget n (r_n). The parameter k is set to be c, and the threshold h is set to be $5n$.

We prove the following property about graph G'.

Lemma 2.1: *Consider any subgraph of G' obtained by removing some of the removable arcs, and let P be a shortest path from s to t in the sub-graph. Then all the arcs in P are traversed from left to right.*

Proof: Let (i, j), $i < j$, be a shortcut in G'. The following observations are implied by the fact that shortcut (i, j) is of length $5(j - i) - 2$.

Observation 1: The simple path from r_i to x_j that uses only the upper portions of gadgets $i+1, \ldots, j-1$ and then traverses the lower portion of gadget j to x_j is of length $5(j-i)-3$. The length of any path between these two nodes that uses shortcut (i, j) is at least $5(j - i)$.

Observation 2: The simple path from y_i to l_j that traverses the lower portion of gadget i to r_i and then uses only the upper portions of gadgets $i+1, \ldots, j-1$ is of length $5(j-i)-3$. The length of any path between these two nodes that uses shortcut (i, j) is at least $5(j - i)$.

To obtain a contradiction assume that P contains an arc that is "backtracked"; i.e., traversed from right to left. Consider such an arc e in P that is not followed by another backtracked arc. (Such an arc must exist.) Since P is simple, the left endpoint of e must be the left endpoint of some other arc. Hence, there are only two cases: either arc e ends at a node y_i and there exists some shortcut (i, j), or arc e ends at a node l_j, for some gadgets i and j.

Case 1: The arc e ends at y_i. In this case, P goes either from r_i to y_i and then to x_j, using shortcut (i, j), or it goes from x_j to y_i and then to r_i. A contradiction, since by Observation 1 this is not the shortest path between r_i and x_j.

Case 2: The arc e ends at l_j. Suppose that P arrives at l_j from x_j. Since P is simple it must continue to r_j using the upper portion of gadget j. If the path arrives at x_j from y_i for some shortcut (i, j), then we get a contradiction, since by Observation 2 this is not the shortest path between y_i and l_j. Otherwise, the path arrives at x_j from y_j. A contradiction, since the path from y_j to r_j using the lower portion of gadget j is shorter. Suppose that P arrives at l_j from r_j using the upper portion of gadget j. Since P is simple it must continue to x_j. If it backtracks to y_i, for some shortcut (i, j), then we get a contradiction, since by Observation 2 this is not the shortest path between y_i and l_j. Otherwise, the path continues to y_j. A contradiction since the path from r_j to y_j using the lower portion of gadget j is shorter. $\square$

We return to the reduction.

Lemma 2.2: *There exists a node cover for G of size at most $c = k$ if and only if there are (at most) k arcs in G' whose removal increases the length of the shortest path from s to t to $5n$.*

Proof: The *if* direction: Suppose that there exists a node cover for G of size at most $c = k$. For each node in the cover we remove the removable arc in the corresponding gadget. We claim that the shortest path from s to t in G' after the removal of these arcs is $5n$. Note that the path consisting of the upper portions of all the gadgets is of length $5n$. We show that this is indeed the shortest path. Assume to the contrary that this is not the case and there is a shortest path P of length less than $5n$.

Note that P must contain at least one of the shortcuts. Consider a shortcut (i, j), for $i < j$. Since either i or j are in the node cover for G, either the removable arc in gadget i or the removable arc in gadget j (or both) were removed. Consequently, if P uses shortcut (i, j) it must either backtrack the shortcut, or go from right to left either from x_j to l_j or from r_i to y_i. A contradiction to Lemma 2.1.

The *only if* direction: Suppose that there are k arcs in G' whose removal increases the shortest path from s to t to $5n$. Each of these arcs is the removable arc in some gadget. Consider the set of nodes $U \subseteq G$ corresponding to these gadgets. We claim that the set U is a node cover. Assume to the contrary that this is not the case, and let (i, j) be an edge such that neither i nor j is in U. This implies that neither the removable arc in gadget i nor the removable arc in gadget j were removed from G'. But then the path $s - l_i - y_i - x_j - r_j - t$, that uses only the upper portions of all gadgets but gadgets i and j and uses the shortcut (i, j) is of length $5n - 1$; a contradiction. □

To make all the arcs removable we replace each unremovable arc by $k + 1$ parallel arcs. This guarantees that after the removal of at most k arcs, at least one of these arcs will not be removed. If parallel arcs are forbidden, we can replace each unremovable arc by a path of length two, adding a linear number of new nodes (in this case the parameters k and h have to be changed accordingly).

To handle directed graphs we orient all the arcs in G' from left to right. In this case, Lemma 2.1 becomes trivial.

Finally, the NP-Completeness proof for the Decision-MVNP is similar. We define an analogue graph in which instead of removable and unremovable arcs we have removable and unremovable nodes. Again, the transformation to a graph in which all nodes are removable is done by duplicating all unremovable nodes $k + 1$ times.

Since clearly Decision-MVAP and Decision-MVNP belong to NP, we proved the following theorem:

Theorem 2.3: *The Decision-MVAP and Decision-MVNP are NP-Complete even when all arcs have unit length.*

3. The counter example

In this section we describe a counter example to the algorithm for the MVAP proposed in [4]. Given a graph $G = (V, E)$, two specified nodes s and t, and a positive integer k, this algorithm enumerates all the paths from s to t in an increasing order of length. Denote these paths by $P_1, P_2, \ldots$. The algorithm constructs a graph $G' = (V, E')$ in stages. Initially, $E' = \emptyset$, then in stage i, it sets $E' = E' \cup \{\text{the arcs in } P_i\}$. It terminates at the first time the minimum $s - t$ cut (i.e., the minimum number of arcs whose removal separates s from t) is more than k. The most vital arcs are the arcs in the last minimum $s - t$ cut whose size is at most k.

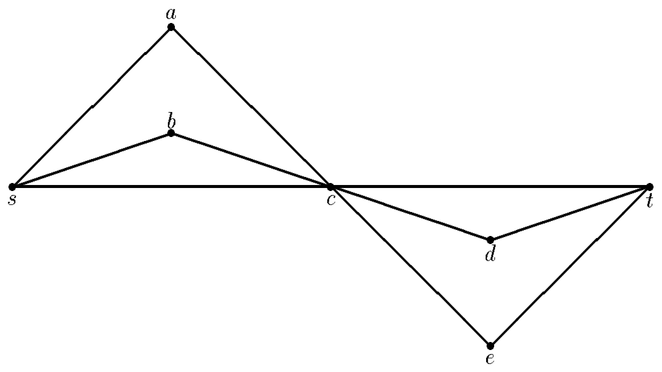

Figure 2: The counter example to the flow algorithm.

The graph $G = (V, E)$ that serves as a counter example to this algorithm is depicted in Figure 2. In this graph: $P_1 = \langle s, c, t \rangle$, $P_2 = \langle s, a, c, t \rangle$, $P_3 = \langle s, b, c, t \rangle$, $P_4 = \langle s, c, d, t \rangle$, and $P_5 = \langle s, c, e, t \rangle$. (The order between P_2, P_3, P_4, and P_5 is arbitrary.) Note that $P_1 \cup \cdots \cup P_5 = E$. Suppose that we invoke the proposed algorithm to solve the MVAP with the graph $G = (V, E)$ and $k = 2$. The algorithm terminates when P_5 is added to E' since at that time the cut consists of three arcs. It outputs the arcs (c, d) and (c, t) (or (d, t) and (c, t)) as the two most vital edges. However, this is not true because after removing (c, d) and (c, t) (or (d, t) and (c, t)) the shortest path from s to t is $\langle s, c, e, t \rangle$ of length three. Whereas, after removing the edges (s, c) and (c, t) the length of the shortest $s - t$ path is four.

4. Correct proof for the Algorithm

The proof of correctness given in [4] relies on the following erroneous claim:

> Let T be a tree of shortest paths from s to all the nodes and let P be the shortest $s - t$ path in T. If some arc $(i, j) \in P$ is removed from T, dividing the node set V into V_s and $\overline{V_s}$ such that $s \in V_s$ and $t \in \overline{V_s}$, then there exist shortest paths from all other nodes in $\overline{V_s}$ to t that do not include nodes in V_s.

The following figure depicts a counter example to this claim.

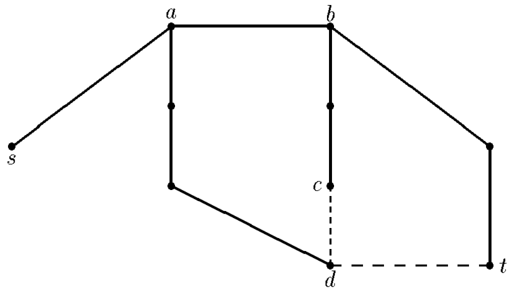

Figure 3: Counter-example to the Claim

The tree of shortest paths from s includes all the arcs except arcs (c, d) and (d, t) (the dashed arcs in the picture). If we delete the arc (a, b) then $c \in \overline{V_s}$. However the shortest path from c to t is via d which belongs to V_s.

Fortunately, for the correctness of the algorithm, the following weaker claim suffices:

> Let T be a tree of shortest paths from s to all the nodes and let P be the shortest $s - t$ path in T. If some arc $(i, j) \in P$ is removed from T, dividing the node set V into V_s and $\overline{V_s}$ such that $s \in V_s$ and $t \in \overline{V_s}$, then there exist shortest paths from all other nodes in $\overline{V_s}$ to t that do not use the arc (i, j).

We briefly outline the proof: consider any node $v \in \overline{V_s}$. Let $P(v, t)$ be a shortest path from v to t. We claim that i cannot be on this path. Assume that i is on this path. Notice that there exist shortest paths from i to v and from i to t that are paths in T. Since $t, v \in \overline{V_s}$, it follows that these paths use the arc (i, j). Hence $P(v, t)$ is not simple, which is a contradiction to the assumption that it is a shortest path.

References

[1] M.O. Ball, B.L. Golden, and R.V. Vohra. Finding most vital arcs in a network. *Operations Research Letters*, 8:73–76, April 1989.

[2] H.W. Corley and D.Y Sha. Most vital links and nodes in weighted networks. *Operations Research Letters*, 1:157–160, September 1982.

[3] M.R. Garey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman and Co., New York, NY, 1979.

[4] K. Malik, A.K. Mittal, and S.K. Gupta. The k most vital arcs in the shortest path problem. *Operations Research Letters*, 8:223–227, April 1989.