TECHNICAL REPORT
R-254
February 1998

Finding Minimal D-separators

Jin Tian
Cognitive Systems Laboratory
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095 USA
jtian@cs.ucla.edu

Azaria Paz
Department of Computer Science
Technion IIT
Haifa 32000
Israel
paz@cs.Technion.AC.IL

Judea Pearl
Cognitive Systems Laboratory
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095 USA
judea@cs.ucla.edu

Abstract

We address the problem of finding a *minimal separator* in a directed acyclic graph (DAG), namely, finding a set Z of nodes that d-separates a given pair of nodes, such that no proper subset of Z d-separates that pair. We analyze several versions of this problem and offer polynomial algorithms for each. These include: finding a minimal separator from a restricted set of nodes, finding a minimum-cost separator, and testing whether a given separator is minimal. We confirm the intuition that any separator which cannot be reduced by a single node must be minimal.

1 Introduction

The problem of finding a small set of variables that renders other sets of variables independent occurs often in probabilistic reasoning systems. The most acute manifestation of this problem occurs in the design of observational studies. Whenever we evaluate the effect of one factor (X) on another (Y), we need to adjust for possible variations in some other set of variables, Z, sometimes called *covariates*, *concomitants*, or *confounders*, which may influence both X and Y and thus bias our assessment. Adjustment amounts to partitioning the population into groups that are homogeneous relative to Z, assessing the effect of X on Y in each homogeneous group, and, finally, averaging the results. Since adjustments require physical measurement of the variables in Z, and since such measurements are often expensive if not impossible, it is natural to seek sets Z that are as small as possible, and still capable of removing the bias caused by confounding. It can be shown (e.g., [Pearl 94, Pearl 95]) that the criterion of bias removal requires that Z d-separates node X from node Y in a (causal)Bayesian network, after removing all arrows from X. Thus, the problem of finding small separating sets translates, in this application, to finding sets of low measurement cost.

The problem also surfaces in model testing and inference. An effective way of testing graphical models for fitness with data is to test whether the data upholds the conditional independencies that are entailed by the graph [Pearl and Verma 91, Spirtes *et al.* 93]. The reliability of such tests deteriorates when the conditioning sets are too large, because large separating sets involve many degrees of freedom and require larger sample size. Finally, in inference problems, small sized separating sets amount to reduced computational costs, especially in conditioning-based algorithms [Pearl 85, Darwiche 95] where the variables in the separating sets are instantiated one at a time.

This paper addresses the mathematical problem of identifying a minimal d-separating set Z in a given directed acyclic graph, such that no proper subset of Z is a separating set. We discuss several versions of this problem and offer polynomial algorithms for each. Section 2 addresses the problem of *testing* whether a given separating set is minimal. The theorems proved in this section confirm the one-at-a-time intuition, namely, if we find a separating set that cannot be reduced by one node (without compromising separation), then that set must be minimal; no reduction by pairs and triples need be

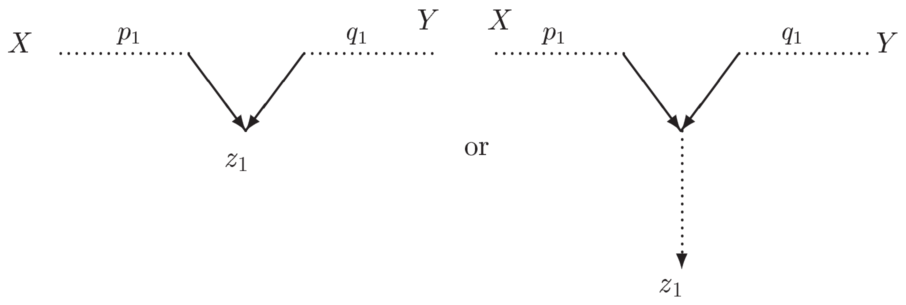

Figure 1: The path activated by z_1.

investigated. Section 3 harnesses the testing algorithms of Section 2 to the solution of our main problem: finding a minimal d-separator. Section 4 extends these results to the case where the separator is restricted to a given set of nodes, while Section 5 finds separators of minimum cardinality and minimum cost.

2 Testing Separators For Minimality

Problem 1 (test for minimal separation) *Given two nodes X and Y in a dag D and a set Z that d-separates X from Y, test if Z is minimal i.e., no proper subset of Z d-separates X from Y.*

First we will find a criterion for a d-separating set to be minimal.

Definition 1 (activated paths) *Consider two nodes X and Y in a dag, a set Z that d-separates X from Y, and a node z_1. We say that a path p from X to Y is* activated *by z_1, if p consists of a path p_1 from X to z_1 and a path q_1 from Y to z_1 such that p_1 meets q_1 at z_1 or one of its ancestors in converging arrows, p_1 and q_1 are not blocked by Z, and none of z_1's descendants is in Z(Figure 1).*

From the property of d-separation, it is clear that $Z \cup \{z_1\}$ is a non-separator if and only if there exists a path activated by z_1.

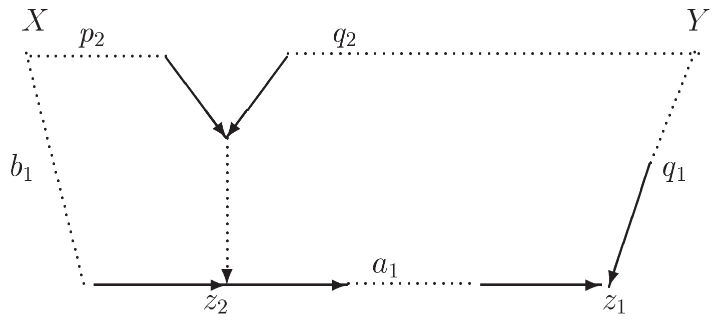

Figure 2: The path activated by z_1 is blocked by z_2.

Lemma 1 *If Z is a separator, both $Z \cup \{z_1\}$ and $Z \cup \{z_2\}$ are not separators, and a path activated by z_1 is blocked by z_2, then z_1 is a descendant of z_2.*

Proof: Assume path (X, p_1, z_1, q_1, Y) is activated by z_1 and path (X, p_2, z_2, q_2, Y) is activated by z_2, as in Definition 1. Assume the path p_1 is blocked by z_2, then p_1 must pass through z_2 and not in converging ar rows, as shown in Figure 2 where $p_1 = (X, b_1, z_2, a_1, z_1)$. Consider the path (X, b_1, z_2, q_2, Y). Both b_1 and q_2 are not blocked by Z, but Z is a separator, so b_1 and q_2 meet at z_2 in converging arrows. So the path a_1 goes away from z_2. a_1 can only go in one direction from z_2 to z_1. For assume z' is the head-to-head node nearest to z_2. None of z_2's descendants is in Z, so z' and none of its descendants are in Z, but this makes Z block the path a_1, hence the path p_1. Q.E.D.

Theorem 1 *Given two nodes X and Y in a dag and a set Z that d-separates X from Y, if the set $Z \cup \{z_1, \ldots, z_n\}$ is also a separator, where $z_1, \ldots, z_n$ are single nodes which are not in Z and $n \geq 2$, then either $Z \cup \{z_1\}$, or $Z \cup \{z_2\}$, $\ldots$, or $Z \cup \{z_n\}$ must be a separator.*

Proof: Assume all $Z \cup \{z_i\}$, $i = 1, \ldots, n$ are not separators, then there is a path d_i activated by z_i, for each $i = 1, \ldots, n$. Consider the path d_1. It must be blocked by at least one node from $\{z_2, \ldots, z_n\}$, else it won't be blocked by $Z \cup \{z_1, \ldots, z_n\}$. Name the node which blocks d_1 as z_2, then from Lemma 1 z_1 is the descendant of z_2. If the path d_2 is blocked by z_1 we get

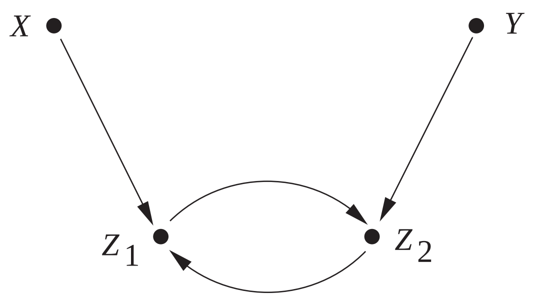

Figure 3: Corollary 2 does not hold in cyclic graphs.

a cyclic graph. Therefore, we can name the node which blocks d_2 as z_3, and z_2 is the descendant of z_3. In general if the path d_i is blocked by one node from $\{z_1, \dots, z_{i-1}\}$ we immediately get a cyclic graph. To avoid this we name the node blocking d_i as z_{i+1}. Finally d_n must be blocked by one node from $\{z_1, \dots, z_{n-1}\}$, we must get a cyclic graph. Q.E.D.

Corollary 1 *If Z and $Z \cup Z_n$ are separators, where $Z_n = \{z_1, \dots, z_n\}$, then there exist a series of $n-1$ sets which are separators: $Z \cup Z_i$, $i = 1, \dots, n-1$, with*

$$Z_1 \subset \cdots \subset Z_{n-1} \subset Z_n,$$

such that each Z_i contains exactly i nodes.

Corollary 2 *If no single node can be removed from a d-separating set Z without destroying d-separability, then Z is minimal.*

Remark: This “one-at-a-time” property is obviously true in undirected graphs; a separating set Z is minimal iff no single node can be removed without destroying separability. But it is not at all obvious in dags, where separation is *nonmono tonic*; reducing the size of a nonseparating set may turn it into a separator. Figure 3 shows that Corollary 2 does not hold in cyclic graphs. The set $Z = \{Z_1, Z_2\}$ d-separates X from Y, and removing either Z_1 or Z_2 destroys d-separability. Yet Z is not minimal, because the empty set d-separates X from Y. Evidently, Corollary 2 rests critically on the assumption of acyclicity.

Corollary 2 provides a method of testing if a d-separating set is minimal, which can be further simplified through the following theorem.

Theorem 2 *If a set Z d-separates X and Y, we can get a smaller d-separator Z' by removing from Z all nodes that are not ancestors of X or Y.*

Proof: Assume Z' is not a separator, then there exists an active path p between X and Y. Because Z is a separator, the path p must be blocked by one node a in $Z - Z'$ which is not an ancestor of X or Y and which is not a head-to-head node on p. Each path going away from a along p must meet a head-to-head node b since a is not an ancestor of X or Y. By the same argument, none of b's descendants are ancestors of X or Y. But this means the path p is also blocked given Z'. Q.E.D.

Let $An(A)$ denote the smallest ancestral set containing set A, that is, $An(A) = A \cup (\cup_{u \in A}\{\text{all ancestors of } u\})$. Theorem 2 says that if Z is a minimal d-separating set between X and Y, then $Z \subseteq An(X \cup Y)$. The following theorem further states that it is sufficient to consider only a subgraph of the dag D, namely $D_{An(X \cup Y)}$, which is composed of nodes from $An(X \cup Y)$ only.

Theorem 3 *Given two nodes X and Y in a dag D, if a set $Z \subseteq An(X \cup Y)$, then Z d-separates X and Y in D if and only if Z d-separates X and Y in $D_{An(X \cup Y)}$.*

Proof: All paths in $D_{An(X \cup Y)}$ are in D. Thus if Z is a separator in D, it is a separator in $D_{An(X \cup Y)}$. If Z d-separates X and Y in $D_{An(X \cup Y)}$, to show that Z also d-separates X and Y in D, we need to prove that all paths from X to Y passing through some non-ancestor nodes are blocked by Z. Consider a path p, if p passes a non-ancestor node head-to-head, then p is blocked by Z. If p doesn't pass any head-to-head non-ancestors, then all nodes on p are ancestors of X or Y. Q.E.D.

Combining Corollary 2, Theorem 2 and Theorem 3 yields the following algorithm for Problem 1.

Algorithm 1 (test for minimal separation)

1. *If Z contains any node which is not in $An(X \cup Y)$, then Z is not minimal, stop.*
2. *Construct $D_{An(X \cup Y)}$.*
3. *Choose any node u from Z.*

4. *Test if $Z - \{u\}$ is a separator in $D_{An(X \cup Y)}$, using the algorithm in [Geiger et al 89].*

5. *If $Z - \{u\}$ is a separator then Z is not minimal, stop.*
   If $Z - \{u\}$ is not a separator then choose a different node from Z, denote it by u and goto step 4. If all nodes in Z have been checked, Z is minimal.

Analysis: Let $|Z|$ stand for the number of nodes in Z, $|E|$ the number of edges in D, $|V_{An}|$ the number of nodes in $An(X \cup Y)$, and $|E_{An}|$ the number of edges in $D_{An(X \cup Y)}$. Step 4 requires $O(|E_{An}|)$ time[Geiger et al 89], and it may be executed up to $|Z|$ times. Thus the overall complexity of the algorithm is $O(|Z| \cdot |E_{An}|)$.

In view of the extensive literature on undirected graphs, it is often advantageous to translate d-separation problems to separation problems in undirected graphs. The following theorem permits this translation.

Theorem 4 *Given two nodes X and Y in a dag D, and a set $Z \subseteq An(X \cup Y)$. Z d-separates X from Y in D if and only if Z separates X from Y in $(D_{An(X \cup Y)})^m$, the moral graph of $D_{An(X \cup Y)}$.*

This follows from a theorem of [Lauritzen et al 90]: A set S d-separates X from Y in a dag D if and only if S separates X from Y in $(D_{An(X \cup Y \cup S)})^m$. The condition of Theorem 4 ensures $An(X \cup Y) = An(X \cup Y \cup Z)$ and so $(D_{An(X \cup Y)})^m \equiv (D_{An(X \cup Y \cup Z)})^m$, for all $Z \subseteq An(X \cup Y)$. The moral graph is obtained from the original graph by connecting any two nodes which have a common child and then dropping directions.

Remark 1: Theorem 4 and Theorem 2 transfer all minimal separation problems from dags to undirected graphs.[1]

Remark 2: Theorem 3 can be easily proved from Theorem 4. Since the same undirected graph $(D_{An(X \cup Y)})^m$ is constructed from D and $D_{An(X \cup Y)}$.

Remark 3: Some properties found in undirected graphs may be extended back to dags through Theorem 4. For example, it is obviously true in undirected graphs that a separation set Z is minimal if no single node can be removed without destroying the separability. Extending this back to dags through Theorem 4, Corollary 2 is obtained, which is not obvious in dags.

[1]Another transformation from dags to undirected graphs was suggested by [Becker and Geiger 94] to facilitate finding small loop cutsets in dags.

In undirected graphs we have efficient methods of testing whether a separation set is minimal[Gafni 97], which are based on the following criteria.

Theorem 5 *Given two nodes X and Y in an undirected graph, a separating set Z between X and Y is minimal if and only if for each node u in Z, there is a path from X to Y which passes through u and does not pass through any other node in Z.*

Proof: If for each node u in Z, there is a path from X to Y which passes through u and does not pass through any other node in Z, it is obvious that no node can be removed from Z without destroying separability. On the other hand, if t here exists a node v such that every path from X to Y which passes through v will pass through some other node in Z, then v can be removed from Z without destroying separability and Z is not minimal. Q.E.D.

This theorem leads to the following algorithm for Problem 1. The idea is that if Z is minimal, then all nodes in Z can be reached using Breadth First Search(BFS) that starts from both X and Y without passing any other nodes in Z

Algorithm 2 (test for minimal separation)

1. *If Z contains any node which is not in $An(X \cup Y)$, then Z is not minimal, stop.*
2. *Construct $D_{An(X \cup Y)}$.*
3. *Construct $(D_{An(X \cup Y)})^m$.*
4. *Starting from X, make Breadth First Search. Whenever a node in Z is met, mark it if it is not already marked, and do not continue along that path. When BFS stops, if not all nodes in Z are marked, Z is not minimal, stop. Remove all markings.*
5. *Starting from Y, run BFS. Whenever a node in Z is met, mark it if it is not already marked, and do not continue along that path. When BFS stops, if not all nodes in Z are marked, Z is not minimal. If all nodes in Z are marked, Z is minimal.*

$\lvert Z\rvert$	$\lvert E_{An}\rvert$	$\lvert E^m_{An}\rvert$	Algorithm 1 $O(\lvert Z\rvert \cdot \lvert E_{An}\rvert)$	Algorithm 2 $O(\lvert E^m_{An}\rvert)$
bounded by constant	$O(\lvert V_{An}\rvert^2)$	$O(\lvert V_{An}\rvert^2)$	$O(\lvert V_{An}\rvert^2)$	$O(\lvert V_{An}\rvert^2)$
	$O(\lvert V_{An}\rvert)$	$O(\lvert V_{An}\rvert)$	$O(\lvert V_{An}\rvert)$	$O(\lvert V_{An}\rvert)$
		$O(\lvert V_{An}\rvert^2)$	$O(\lvert V_{An}\rvert)$	$O(\lvert V_{An}\rvert^2)$
$O(\lvert V_{An}\rvert)$	$O(\lvert V_{An}\rvert^2)$	$O(\lvert V_{An}\rvert^2)$	$O(\lvert V_{An}\rvert^3)$	$O(\lvert V_{An}\rvert^2)$
	$O(\lvert V_{An}\rvert)$	$O(\lvert V_{An}\rvert)$	$O(\lvert V_{An}\rvert^2)$	$O(\lvert V_{An}\rvert)$
		$O(\lvert V_{An}\rvert^2)$	$O(\lvert V_{An}\rvert^2)$	$O(\lvert V_{An}\rvert^2)$

Table 1: Comparison of Algorithm 1 with Algorithm 2.

Analysis: Let $|E^m_{An}|$ stand for the number of edges in $(D_{An(X\cup Y)})^m$. Step 3-5 each requires $O(|E^m_{An}|)$ time. Thus the complexity of the algorithm is $O(|E^m_{An}|)$.

The comparison of Algorithm 1 with Algorithm 2 under different situations is given by Table 1.

3 Finding a Minimal Separator

Problem 2 (minimal separation) *Given two nodes X and Y in a dag D, find a minimal d-separating set between X and Y, namely, find a set Z such that Z, and no proper subset of Z, d-separates X from Y.*

A variant of Algorithm 2 solves this problem.

Algorithm 3 (minimal separation)

1. *Construct $D_{An(X\cup Y)}$.*

2. *Construct $(D_{An(X\cup Y)})^m$.*

3. *Find a separating set Z' in $(D_{An(X\cup Y)})^m$.*

4. *Starting from X, run BFS. Whenever a node in Z' is met, mark it if it is not already marked, and do not continue along that path. When*

BFS stops, let Z'' be the set of nodes which are marked. Remove all markings.

5. *Starting from Y, run BFS. Whenever a node in Z'' is met, mark it if it is not already marked, and do not continue along that path. When BFS stops, let Z be the set of nodes which are marked.*

Return(Z).

Analysis: Step 2-5 each requires $O(|E^m_{An}|)$ time. Thus the overall complexity of the algorithm is $O(|E^m_{An}|)$.

In Algorithm 3 we transform a dag to an undirected graph, then find a minimal separating set in the undirected graph. The method used in the undirected graph can not be extended to use directly in a dag. However, there is a less efficient method for finding a minimal separating set in undirected graphs which can be extended to dags. In undirected graphs, given a separating set Z and a node u in Z, if $Z-\{u\}$ is not a separator, then u must belong to every subset of Z which is a separator. Thus to find a minimal separating subset of Z, we check Z node-by-node: if $Z-\{u\}$ is not a separator, keep u; else remove u from Z. From Theorem 4, this method can be used directly in dags. To this purpose we need to find a d-separating set in $D_{An(X\cup Y)}$ first, which can be easily found: if X and Y are not adjacent then they are d-separated by their parent set $Pa(X\cup Y)$[Verma and Pearl 91], the union of the sets of parents of X and Y(less X and Y).

Algorithm 4 (minimal separation)

1. *Construct $D_{An(X\cup Y)}$.*
2. *Set Z to be $Pa(X\cup Y)$.*
3. *Choose one node u from Z.*
4. *Test if $Z-\{u\}$ is a separator in $D_{An(X\cup Y)}$, using the algorithm in [Geiger et al 89].*
5. *If $Z-\{u\}$ is a separator, set $Z=Z-\{u\}$.*
   Choose a different node from Z, denote it by u and goto step 4. If all nodes in Z have been checked, stop.

Return(Z).

Analysis: Step 4 requires $O(|E_{An}|)$ time, and it is executed $|V_{Pa}|$ times, where $|V_{Pa}|$ stands for the number of nodes in $Pa(X \cup Y)$. Thus the overall complexity of the above algorithm is $O(|V_{Pa}| \cdot |E_{An}|)$.

The comparison of Algorithm 3 ($O(|E^m_{An}|)$) with Algorithm 4 ($O(|V_{Pa}| \cdot |E_{An}|)$) is given by Table 1 with $|Z|$ substituted by $|V_{Pa}|$.

4 Restricted Separation

Problem 3 (restricted separation) *Given two nodes X and Y in a dag D and a set S of nodes not containing X and Y, find a subset Z of S which d-separates X and Y.*

We already know that if we find a restricted separator over S in $(D_{An(X\cup Y)})^m$ then it is a d-separator in D. On the other hand, if there exists a restricted d-separator over S in D then there must exist a restricted separator over S in $(D_{An(X\cup Y)})^m$ by removing all nodes which are not in $An(X \cup Y)$ from it. Again, this problem is transferred to the corresponding problem in an undirected graph. But in an undirected graph, if any subset of a set S' is a separator, then S' itself is a separator. These observations yield the following theorem.

Theorem 6 *Given two nodes X and Y in a dag D and a set S of nodes not containing X and Y, there exists some subset of S which d-separates X and Y only if the set $S' = S \cap An(X \cup Y)$ d-separates X and Y.*

So Problem 3 is solved by testing if S' d-separates X and Y. This requires $O(|E|)$ time using the algorithm in [Geiger et al 89].

Problem 4 (restricted minimal separation) *Given two nodes X and Y in a dag D and a set S of nodes not containing X and Y, find a subset Z of S which is a minimal d-separating set between X and Y.*

This problem is readily solved using Algorithm 3 or Algorithm 4. To use Algorithm 3, we let Z' be $S' = S \cap An(X \cup Y)$ in step 3. The time complexity is $O(|E^m_{An}|)$. To use Algorithm 4, we let Z be S' in step 2. The time complexity is $O(|S'| \cdot |E_{An}|)$. The comparison of this two algorithms is given before.

5 Finding a Minimum-cost Separator

Problem 5 (minimum separation) *Given two nodes X and Y in a dag D, find a minimum(least cardinality) d-separating set between X and Y.*

This problem is solved by using network flow techniques in undirected graphs.

Algorithm 5 (minimum separation)

1. *Construct $D_{An(X \cup Y)}$.*
2. *Construct $(D_{An(X \cup Y)})^m$.*
3. *Take X as source node and Y as sink node. Assign unit node capacity to all the remaining nodes.*
4. *Change $(D_{An(X \cup Y)})^m$ to a flow network as described in [Even 79].*
5. *Find a minimum separating set Z between X and Y, which corresponds to a minimum cut, using a maximum flow algorithm.*

Return(Z).

Analysis: The complexity of this algorithm is $O(|V_{An}|^{1/2} \cdot |E^m_{An}|)$ when the Dinic algorithm is used[Even 79].

Algorithm 5 can also be used to find a minimum separating set restricted to a set S: we assign unit node capacity to all nodes in $S' = S \cap An(X \cup Y)$ and infinite capacity to all other nodes in the step 3. When the Dinic algorithm is used, the complexity is $O(|V_{An}|^3)$ now instead of $O(|V_{An}|^{1/2} \cdot |E^m_{An}|)$ because not all nodes have unit capacities[Even 79]. A more efficient algorithm can be achieved through the following theorem.

Theorem 7 *Given a set S of nodes in an undirected graph G, we construct a graph G_S restricted to S as follows: The nodes of G_S are S. The edges of G_S are the edges in G between the nodes in S plus the following edges: Let p and q be two nodes in S. If there is a path in G from p to q with all its nodes except p and q outside of S, then connect p and q by an edge in G_S. Then a set $Z \subseteq S$ separates two nodes X and Y in G_S if and only if Z separates X from Y in G.*

Proof: Immediately; each active path in G_S corresponds to an active path in G, and vice versa.

Algorithm 6 (restricted minimum separation)

1. *Construct* $D_{An(X \cup Y)}$.
2. *Construct* $(D_{An(X \cup Y)})^m$.
3. *Construct the graph* $G_{S'}$ *restricted to* $X \cup Y \cup S'$ *as defined in the Theorem 7, where* $S' = S \cap An(X \cup Y)$.
4. *Take* X *as source node and* Y *as sink node. Assign unit node capacity to all nodes in* S'.
5. *Change* $G_{S'}$ *to a flow network as described in [Even 79].*
6. *Find a minimum separating set* Z *between* X *and* Y*, which corresponds to a minimum cut, using a maximum flow algorithm.*

Return(Z*).*

Analysis: To construct $G_{S'}$, for each node of $X \cup Y \cup S'$ we use Breadth First Search on $(D_{An(X \cup Y)})^m$ to find all its adjacent nodes in $G_{S'}$ in time $O(|E^m_{An}|)$. Altogether constructing $G_{S'}$ requires $O(|S'| \cdot |E^m_{An}|)$ time. Finding a minimum separating set in $G_{S'}$ requires $O(|S'|^{1/2} \cdot |E^{G_{S'}}|)$, where $|E^{G_{S'}}|$ is the number of edges in $G_{S'}$. Thus the complexity of this algorithm is $O(|S'| \cdot |E^m_{An}|)$, which is asymptotically faster than $O(|V_{An}|^3)$ when $(D_{An(X \cup Y)})^m$ is sparse.

Problem 6 (minimum cost separation) *Given two nodes* X *and* Y *in a dag* D*, if each node* u *other than* X *and* Y *is associated with a non-negative number* $c(u)$ *called the cost of* u*, find a d-separating set between* X *and* Y *which has the minimum total costs.*

The minimum cost d-separating set must be a minimal d-separating set. Thus the problem is equivalent to finding a minimal d-separating set which has the minimum cost.

Algorithm 5 can be used to solve this problem. The cost of each node is assigned as the capacity of the node in step 3 of Algorithm 5. The complexity of the algorithm is $O(|V_{An}|^3)$ when the Dinic algorithm is used. The correctness of this algorithm is justified by the well-known *Max-flow min-cut theorem*[Even 79].

Algorithm 6 can be used to find a minimum cost d-separating set restricted to a set S, assigning the cost of each node as the capacity of the node in step 4. Constructing $G_{S'}$ requires $O(|S'| \cdot |E^m_{An}|)$ time. Finding a minimum cost separating set in $G_{S'}$ requires $O(|S'|^3)$ time. Thus the complexity of the algorithm is $O(|S'| \cdot |E^m_{An}| + |S'|^3)$.

6 Discussion

We have shown that problems associated with finding d-separators in directed acyclic graphs are not substantially more complex than those associated with finding separators in undirected graphs. In particular, we have given polynomial algorithms for (1) finding any minimal separator, (2) finding a minimal separator from a restricted set of nodes, (3) finding a minimum-cost separator, and (4) testing whether a given separator is minimal. Additionally, we have shown that any separator which cannot be reduced by a single node must be minimal.

Although the problems addressed in this paper are formulated in terms of separating a pair of singleton nodes X and Y, all our theorems hold for the general case where X and Y are disjoint sets of nodes. To adjust the algorithms for this general case, one can simply add two extra nodes X' and Y' to $(D_{An(X \cup Y)})^m$, such that X' is connected to all nodes in set X and Y' is connected to all nodes in set Y, and seek a minimal separator between X' and Y' restricted to nodes outside X and Y.

References

[Becker and Geiger 94] A. Becker and D. Geiger. Approximation algorithms for the loop cutset problem. In R. Lopez de Mantaras and D. Poole, editors, *Uncertainty in Artificial Intelligence 10*, pages 60–68. Morgan Kaufmann, San Mateo, CA, 1994.

[Darwiche 95] A. Darwiche. Conditioning algorithms for exact and appropriate inference in causal networks. In P. Besnard and S. Hanks, editors, *Uncertainty in Artificial Intelligence 11*, pages 99–107. Morgan Kaufmann, San Francisco , CA, 1995.

[Even 79] S. Even, *Graph Algorithms*, Computer Science Press, 1979.

[Gafni 97] E. Gafni, Personal communication, 1997.

[Geiger et al 89] D. Geiger, T.S. Verma and J. Pearl, *d*-Separation: From Theorems to Algorithms, *Proceedings*, 5th Workshop on Uncertainty in AI, Windsor, Ontario, Canada, August 1989, pp. 118-124.

[Lauritzen et al 90] S.L. Lauritzen, A.P. Dawid, B.N. Larsen and H.G. Leimer, Independence Properties of Directed Markov Fields, *Networks*, vol.20 pp491-505, 1990.

[Pearl 85] J. Pearl. A constraint propagation approach to probabilistic reasoning. In *Proc. of the First Workshop on Uncertainty in AI*, pages 31–42, Los Angeles, CA, August 1985.

[Pearl 94] J. Pearl. A probabilistic calculus of actions. In R. Lopez de Mantaras and D. Poole, editors, *Uncertainty in Artificial Intelligence 10*, pages 454–462. Morgan Kaufmann, San Mateo, CA, 1994.

[Pearl 95] J. Pearl. Causal diagrams for experimental research. *Biometrika* 82, pp. 669–710, December 1995.

[Pearl and Verma 91] J. Pearl and T. Verma. A theory of inferred causation. In J.A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 441–452. Morgan Kaufmann, San Mateo, CA, 1991.

[Spirtes *et al.* 93] P. Spirtes, C. Glymour, and R. Schienes. *Causation, Prediction, and Search.* Springer-Verlag, New York, 1993.

[Verma and Pearl 91] T.S. Verma and J. Pearl, Equivalence and Synthesis of Causal Models, *Uncertainty in Artificial Intelligence 6*, P.P. Bonissone, M. Henrion, L.N. Kanal and J.F. Lemmer (eds), Elsevier Science Publishers, 1991, pp. 255-268.