Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 23, no. 2, pp. 317–344 (2019)
DOI: 10.7155/jgaa.00493

Computation in Causal Graphs

Juli Atherton[1] *Derek Ruths*[2] *Adrian Vetta*[2,3]

[1]Départment de Mathématiques, Université du Québec à Montréal
[2]School of Computer Science, McGill University
[3]Department of Mathematics and Statistics, McGill University

Abstract

The goal of this paper is to introduce some of the important concepts in causal graph theory and examine them from combinatorial and computational perspectives. Of fundamental importance in applications of causal models that use graphs are dependence-separators or, simply, d-separators. A vertex set Z is a d-separator for a pair of disjoint vertex sets (X, Y) if X and Y are independent conditioned on Z. For the case of a single-setpair it is known that d-separators can be found efficiently by elegant network flow techniques. In this paper, we consider d-separators for a collection $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_k, Y_k)\}$ of setpairs. We focus on two classes of combinatorial objects in this multiple-setpair framework: d-separators and d-super-separators. We say that Z is a d-separator for multiple setpairs if, for each i, X_i and Y_i are independent conditioned on Z; we say that Z is a d-super-separator if, for each i, there exists $Z_i \subseteq Z$, such that X_i and Y_i are independent conditioned on Z_i. For the latter object, we give an $O(\log^2 k)$-approximation algorithm for the problem of finding a minimum cost d-super-separator. The focus on approximation algorithms is necessary as we show this problem is NP-complete. For the former object, we show it is hard to determine whether a d-separator exists, even when there are just five setpairs. This problem remains hard even if each setpair consists of singleton vertices, provided the number of setpairs is large. On the positive side, we show that if there are a fixed number of singleton setpairs then a d-separator for multiple setpairs can be found in polynomial time, if one exists.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
October 2017	February 2018	April 2018	May 2018	April 2019
		Published:		
		April 2019		
	Article type:		Communicated by:	
	Regular Paper		G. Liotta	

E-mail addresses: juli.atherton@gmail.com (Juli Atherton) druths@ruthsresearch.org (Derek Ruths) adrian.vetta@mcgill.ca (Adrian Vetta)

1 Introduction

The use of causal graphs is having a profound impact in statistics, medicine, artificial intelligence, philosophy, and the natural and social sciences (see, for example, [11], [8], [22], [21], [25], [24] and [10]). Despite encompassing two topics close to theoreticians' hearts, namely networks and computation, and having huge practical application, causality theory has received little attention in the theory community. The goal of this paper, therefore, is to introduce some of the important concepts in causality and examine them from combinatorial and computational perspectives.

A causal graph is a directed, acyclic graph $G = (V, A)$ whose vertices $V_1, \dots, V_n$ represent random variables. Associated with the graph is a joint probability distribution P. This probability distribution is consistent with the graph in that it satisfies the *Markov condition*: each random variable V_i is independent of its non-descendants conditional on its parents. Together, the graph and the probability distribution can be used to model and analyze an extremely wide range of systems, as illustrated by the applications we describe in Section 3.

Associated with a causal graph are the combinatorial objects dependence-separators or, simply, d-separators.[1] Consequently, d-separators will be the focus of this paper. A vertex set Z is a *d-separator* for a pair of disjoint vertex sets (X, Y) if X and Y are independent conditioned on Z. Informally, a d-separator resembles the classical graph theoretic notions of vertex cuts/separators. That is, we wish to select (condition on) vertices in the graph into order to disconnect the paths (carrying a dependency) between X and Y. However, there is an important difference: vertices may have different types in that they may disconnect *or* reconnect a path! In addition, the type of a vertex may vary depending upon the path in question. Furthermore, and most unusually, d-separation is a global property not a local property in that a vertex may reconnect a path it does not lie upon, even in cases where the path is far from the vertex in the graph. This may sound complicated but, as we will see in Section 2, d-separators have a very clean and natural graph theoretic definition. There we will also illustrate this concept with some simple examples. We emphasize here, though, that the d-separation property is purely combinatorial: it depends *only* upon the causal graph (consistent with the joint probability distribution) and not on more technical aspects of the probability distribution itself. For this reason, the exact probability distribution will not concern us for the rest of the paper. Moreover, the focus of study for this paper are d-separators for a collection $\{(X_1, Y_1), (X_2, Y_2), \dots, (X_k, Y_k)\}$ of setpairs. We consider two classes of combinatorial objects in this multiple-setpair framework: d-separators and d-super-separators. We say that Z is a *d-separator* for multiple setpairs if, for each i, X_i and Y_i are independent conditioned on Z; we say that Z is a *d-super-separator* if, for each i, there exists $Z_i \subseteq Z$, such that X_i and Y_i are independent conditioned on Z_i. For multiple setpairs, d-separators are combinatorially more

[1]We remark that, in the literature, the d in d-separator is sometimes described as standing for "directed". We will follow the more specific and more natural nomenclature, namely "dependence".

interesting objects but, as we discuss in Section 3, d-super-separators are more important in applications.

1.1 Overview of the Paper

In Section 2 we explain how dependencies can be transmitted along paths in causal graphs. This gives the combinatorial motivation underlying the concept of d-separation. Then in Section 3 we discuss important applications of d-separators. In Section 4 we quickly review what is known about d-separations for a single setpair. In particular, we describe an elegant network flow formulation that allows a minimum cardinality d-separator to be found in polynomial time. We also derive a combinatorial characterization for when d-separators exist - this leads to a linear time algorithm to find a (non-minimum) d-separator. Our main results are given in Section 5 and Section 6.

In Section 5, we consider d-super-separators for multiple setpairs. We show, on the negative side, that the problem of finding a minimum cost d-super-separator is NP-complete. On the positive side we present a factor $O(\log^2 k)$ approximation algorithm for the problem.

Finally, we consider d-separators for multiple setpairs. In Section 6.1, we show that, in very sharp contrast to the single setpair case, for multiple setpairs not only is it hard to find a minimum cardinality d-separator, but it is NP-hard to determine whether any d-separator even *exists* in a causal graph. Thus, it is very unlikely that any approximation algorithm exists for this problem. This result holds even when there are just five setpairs. In Section 6.2, we consider the special case when each set in a setpair consists of a single vertex (or at most a fixed number of vertices). We prove then that it is still hard to determine whether any d-separator exists if the number of setpairs is large, but that there is a polynomial time algorithm if the number of setpairs is fixed.

1.2 Graph Theoretic Notation

Here we present some definitions we require from graph theory. We are given a directed causal graph $G = (V, A)$ and a collection of pairwise disjoint vertex sets, $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_k, Y_k)\}$. We refer to $T = \bigcup_i (X_i \cup Y_i)$ as *terminal vertices*. A *path* P consists of a collection of vertices $P = \{v_0, v_1, v_2, \ldots, v_k\}$ where for each j, $0 \leq j \leq k-1$, we have $(v_j, v_{j+1}) \in A$ or $(v_{j+1}, v_j) \in A$. The path is *simple* if all of the v_i are distinct. It is *directed* if all the arcs in P have the same orientation, that is either $(v_j, v_{j+1}) \in A$ for all $0 \leq j \leq k-1$ or $(v_{j+1}, v_j) \in A$ for all $0 \leq j \leq k-1$. We call v_0 and v_k the *endpoints* of P and $v_1, v_2, \ldots, v_{k-1}$ the *internal vertices* of P. We say that P is a *terminal path* if $v_0 \in X_i$ and $v_k \in Y_i$, for some i, or vice versa.

For each arc $a = (u, v) \in A$ we say that u is the *tail* and v is the *head* of a. The *out-degree* of a vertex v is the number of arcs that v is the tail for; the *in-degree* is the number of arcs it is the head for. For a vertex v on a path P, we will be primarily interested in the in-degree and out-degree of v *with respect to the path*, that is, counting only the arcs from P.

2 D(ependence)-Separation

Take a causal graph G and vertex sets X and Y. Now suppose we want to find a set Z such that X and Y are conditionally independent given Z. Thus, conditioning on Z breaks any existing dependence between X and Y and does not create any new ones. We denote this by $X \perp\!\!\!\perp Y | Z$, and dub Z a *dependence-separator* or, more simply, a *d-separator*. *D*-separators were introduced by Pearl [20]. In this section we give a standard explanation motivating these objects; for more details on this discussion see the books of Pearl [22] and Sprites et al. [25].

2.1 Confounders and Colliders

Before giving a formal definition of *d*-separator, we present a few examples that will help illuminate the concepts involved. Consider first the graph shown in Figure 1a). In the late 1940s, it was noted there was a strong association between the sales of ice cream X and the incidence of polio Y. Indeed this lead to campaigns against the consumption of ice cream (and fizzy drinks). The association is not causal, however, since both increase in the summer months. Thus the random variables X and Y are dependent, but they become independent if we condition on the season Z.

The variable season is a special case of a confounder. In graph theory terms, a vertex z is a *confounder* for x and y if there is a directed path from z to x *and* a directed path from z to y.

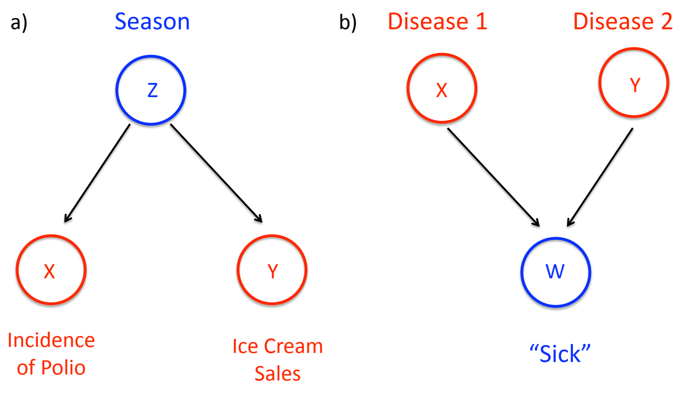

Figure 1: A Confounder and a Collider.

On the other hand, consider the example shown in Figure 1b). We have two diseases that occur independently. However, they will appear (negatively) dependent if we sample data from hospital patients only; given that a person is sick, if Disease 1 is not the cause then Disease 2 must be, and vice versa. To see this, observe that two independent binary random variables, X and Y

become negatively dependent if we condition on $W = X \vee Y$.[2] This is known as *Berkson's Paradox* [3] in the medical literature.

Here the variable sick is a collider between x and y. Formally, in graph theory terms, vertex w is a *collider* on a simple $x - y$ path P if it has in-degree exactly two with respect to the path. If $w \in P$ is not a collider on P then we call it a *non-collider* on P.

2.2 Dependence along Paths

Note that the collider property is defined with respect to a specific path P. Thus, a vertex can be both a collider on one path and a non-collider on a different path. Colliders are of interest to us as they determine whether or not a path P between two vertices x and y carries a dependence along it. This follows from our discussion in Section 2.1. To see this, take a vertex w with neighbours a and b on an $x - y$ path P. If w is a child of both a and b then it is a collider on the path. As seen, the parents of a collider are independent (with respect to that path) and so w *blocks* any dependence propagating along a path. On the other hand, if w is a non-collider on P then it is either a parent of both a and b, or it is the parent of a and the child of b or vice versa. In the former case, a and b are dependent along the path via the confounder w. In the later case a and b are dependent as a is a direct ancestor or b (or vice versa). Repeating this argument at every vertex along the path, we obtain:

Observation 1 *An $x - y$ path induces a dependence between its end-points if and only if it contains no collider.* □

We remark, in view of what follows, that the term dependence in Observation 1 should be viewed as a dependence assuming that the conditioning set Z is empty. Now, to understand what this observation implies let us consider the structure of $x - y$ paths. Note that the arcs of any path can be partitioned into a set of alternating directed sub-paths called *segments*. Hence, adjacent segments meet alternately at colliders and confounders on the path. We then have the following simple observation.

Observation 2 *An $x - y$ path induces a dependence if and only if it consists of one segment or contains two segments incident at a confounder.*

Proof: By Observation 1, we just need to show that an $x - y$ path P contains no collider if and only if it consists of one segment or contains two segments incident at a confounder. If the path P has one segment then it is either a directed path from x to y or a directed path from x to y. In neither case does P contain a collider. If P contains two segments incident at a confounder c then it consists of a directed path from c to x and a directed path from c to y (we call such a path a *hill path*[3] with *summit* c); thus, it contains no collider. On the

[2]For example, suppose $P(X) = \frac{1}{2} = P(Y)$, and so $P(X \vee Y) = \frac{3}{4}$. Then $P(X|(Y \wedge (X \vee Y)) = P(X|Y) = \frac{1}{2} < P(X|X \vee Y) = \frac{2}{3}$.

[3]Hill paths are called *treks* in [25].

other hand, if P contains two segments that are not incident at a confounder then they must be incident at a collider. If P contains three or more segments then clearly it contains at least one collider. □

Observation 2 immediately tells us that if we want to make X and Y conditionally independent then we must break the dependencies induced by every directed path and every hill path between x and y. That is, we must condition on some vertex in all such paths. However, as the example in Figure 1b) shows, this is not sufficient: conditioning on a vertex may cause dependencies to be induced along other paths that contain that vertex as a collider! More specifically, colliders block a path but if we conditioned upon them all then we will *unblock* the path. Thus we now need to block such paths as well. The question then is how do we ensure that the set Z we condition on does indeed block every $x-y$ path? This question is answered by the concept of d-separation.

2.3 D-Separators.

We are finally ready to give a graph theoretic definition of a d-separator [20].

A set Z **d-separates** x and y if for every $x-y$ path P we have
(i) Some non-collider on P is in Z.
or
(ii) There is a collider c and neither c nor any descendent of c is in Z.

Let's try to understand why this is the appropriate concept to induce independence. As discussed, selecting a non-collider will block dependence propagation along the path. In contrast, selecting a collider c causes a dependence between the parents of c. But this is also true if we select any descendent of c: conditioning on a descendent of c creates a dependence between the parents of c! Thus, for P to be blocked, if no non-collider of P is selected we need to ensure that there is some collider on P that is not selected nor are any of its descendants. The definition follows.

Thus, if x is d-separated from y by a set Z then x and y are independent conditioned on Z. In general, we will be more interested in making two vertex sets X and Y conditionally independent rather than just two singleton vertices.

A set Z d-separates a setpair (X, Y)
if and only if
Z d-separates every $x-y$ path P, for all $x \in X, y \in Y$.

If Z is a d-separator for X and Y then we write $X \perp\!\!\!\perp Y|Z$. For the case of multiple setpairs $(X_1, Y_1), (X_2, Y_2), \ldots, (X_k, Y_k)$, not just one setpair, we have the following generalization:

A set Z d-separates multiple setpairs $(X_1, Y_1), \ldots, (X_k, Y_k)$
if and only if
Z d-separates each setpair (X_i, Y_i), for all $1 \leq i \leq k$.

This naturally leads us to the following questions. Given a causal graph G: Does a d-separator exist? If so, can we efficiently find one? If so, can we efficiently find a small d-separator? We will address these questions in Section 4 for a single setpair and in Section 6 for multiple setpairs. Before doing so, however, let's discuss how d-separators relate to other types of vertex-separator studied in the graph theory and applications literature. Familiar vertex separators include, of course, the standard vertex cuts studied by Menger [17]. Furthermore, additional properties are often imposed upon the separators. There are numerous famous examples: star cutsets [4]; clique cutsets [5, 29]; stable cutsets [27]; connected separators [19, 18]. Of particular importance in applications are balanced separators [16] when the components formed by the separator have bounded cardinalities, for example in planar graphs [15, 6].

There are, however, important differences between d-separators and these other types of separator. Firstly, the property of d-separation is *non-monotonic*: if Z is a d-separator it need not be the case that $Z \cup \{z\}$ is too. Adding z may unblock a path. A vertex v can simultaneously be a non-collider on one path and a collider (or the descendent of a collider) on another path. So it can simultaneously block and unblock paths; in contrast, in standard separator problems vertices can only block paths.

Secondly, whether or not a path is blocked may depend upon vertices that do not lie on the path - namely, descendants of colliders on the path. Thus, d-separation is a *global* property. In contrast, for standard separator problems, separation is a *local* property: whether or not a path is blocked can be determined by consideration of that path alone. As we will see, these two properties make d-separators rather tricky to deal with.

Luckily, d-super-separators are slightly easier to deal with as the property of d-super-separation is monotonic. Formally, we define a d-super-separator as:

A set Z d-super-separates multiple setpairs $(X_1, Y_1), \ldots, (X_k, Y_k)$
if and only if
For all $1 \leq i \leq k$, $\exists Z_i \subseteq Z$ such that Z_i d-separates each setpair (X_i, Y_i).

Clearly, if $Z_i \subseteq Z$ then $Z_i \subseteq Z \cup \{z\}$, for all $z \notin Z$. Monotonicity is extremely helpful in designing algorithms with strong performance guarantees. We show this via our approximation algorithm for the minimum cost d-super-separator problem in Section 5. We remark that for multiple setpairs, since d-separators are combinatorially more complex objects than d-super-separators, we present our theoretical results on d-separators after those on d-super-separators.

3 Applications of D-Separators

In this section, since it may not be immediately clear how useful d-separators and d-super-separators are, we give brief descriptions of several very important applications.

• **Model Testing.** How to test the validity of any model is a fundamental issue. Often in the social and health sciences, for example, this task is extremely difficult. However, d-separators do offer a simple way to refute a hypothesized causal model. This is because d-separation characterizes exactly the conditional independence relations that follow from applying the Markov condition to a directed acyclic graph[4]. Consequently, we can use experimental data on the probability distribution to evaluate the validity of our causal graph. Simply find a d-separator in the graph and then test, using the data, whether the proposed separator does indeed induce conditional independence amongst the corresponding variables. Furthermore, the cheapest way to increase the reliability of these tests is to collect data related to a d-separator for multiple setpairs.

• **Prediction and the Evaluation of Interventions.** Causal graphs allow us to measure the implications of constraining some variables on other variables. For example, we might wish to estimate the effect on lung cancer rates of a new smoking policy, or estimate the impact on inflation of a policy of quantitative easing. To do this we need to be able to formulate the impact of an intervention in terms of observational data only. At first thought this may appear impossible since (i) the graph may contain unobserved or immeasurable variables and (ii) the very fact that some variables are to be constrained may make inference dubious using only data collected when those variables were unconstrained. Remarkably, however, these two problems can often be overcome: prediction may be achieved via algebraic simplifications obtained by the examination of d-separators in (subgraphs of) the causal graph! See [22] for detailed descriptions of this approach.

These applications rely upon successful stratification upon variables in a d-separator Z. To achieve this it suffices to collect sufficient data only for these variables. If the cost of data collection for a variable $v \in V$ is c_v then the total cost will be $\sum_{v \in Z} c_v$. For large databases that will be queried more than once we are in the case of multiple setpairs $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_k, Y_k)\}$. There a cost for variable v is incurred if and only if it is needed for stratification with respect to at least one setpair, say Z_i. Thus Z need only be a d-super-separator not a d-separator − the information we have on variable $v \in Z$ does not hurt us if we need to stratify on Z_j where $v \notin Z_j$. Our aim therefore is to find a minimum cost d-super-separator.

• **Design of Observational Studies.** Consider an observational medical study where we desire a subset of the variables (upon which we perform stratification) that allow the true treatment effect to be determined in a population of treated and untreated subjects. The wrong choice of variables will lead to spurious associations between treatment and response. Furthermore, some of the variables

[4] More accurately, any conditional independence relation that is not implied by the Markov condition is unstable. That is, it ceases to hold given a perturbation to the probability distribution; see [25] or [22] for specific details.

may be unmeasurable (or very costly to measure). Given the causal graph, a correct set of variables to choose corresponds to a d-separator in an auxiliary graph - such a subset is said to satisfy the *backdoor criteria*; see [22]. More broadly, given the expense of medical studies we may wish to design a study that can address more than one question. Thus we should design the trial to collect data with respect to a minimum cost d-super-separator.

• **Graph Bases.** A basis encodes all the statistical claims in a linear Markovian model [22]. A graph basis is a *collection* of d-separators for multiple setpairs that d-separate every non-adjacent pair of vertices in the causal graph and correspond to a basis. A graph basis therefore induces a d-super-separator. The simplest basis is the collection of parent sets for each vertex, but Pearl and Meshkat [23] argue that "it may be possible to select a more economical basis". Consequently, we again need to focus upon low cost d-super-separators.

4 D-Separators for a Single Setpair

Given a graph $G = (V, A)$ and a single setpair (X, Y), can we find a set Z that d-separates X and Y? To answer this question, we first present a very simple characterization theorem concerning the existence of d-separators in the case of a single setpair (X, Y). Applying the characterization theorem will give a linear time algorithm to obtain a d-separator if one exists. In this section, we also briefly discuss a result of Lauritzen et al. [12] that shows how to solve the harder problem of finding a *minimum cardinality* d-separator in polynomial time using network flow techniques. Their result is intriguing in that it shows the d-separation problem for a single setpair can be formulated as a minimum s-t cut problem in an auxiliary undirected graph $\mathcal{G}$ - this is despite the apparent additional complexities associated with d-separators previously discussed.

Let's begin with the characterization theorem. First observe that to obtain a d-separator it suffices to block only those $x - y$ paths that contain only non-terminals as internal vertices. To see this, suppose a path P between x and y contains an internal terminal vertex, without loss of generality x'. Then P has as a subpath a terminal path P' between x' and y. But if P' is blocked by Z then clearly P is as well. So throughout this section, the terminal paths we consider will be assumed to contain internally only non-terminal vertices.

We now need a few more definitions. A *terminal arc* is an arc (x, y) or (y, x) for some $x \in X$ and $y \in Y$, that is, a terminal path with no internal vertices. A vertex v is called a *panoramic collider* if it has a parent $x \in X$ and a parent $y \in Y$. A *hod* is a subgraph connecting three terminal vertices, $h_1 \in X, h_2 \in Y$ and $h_3 \in X \cup Y$, in the following fashion. First, there is a path between h_1 and h_2 that consists only of a panoramic collider Υ; we call this path the *box* of the hod. Second, there is a path from Υ to h_3; we call this path the *handle* of the hod. Possibly the handle is empty and $\Upsilon = h_3$. An example of a hod is shown in Figure 2.

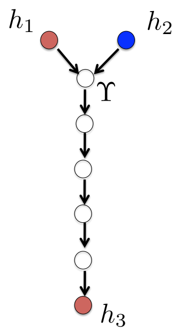

Figure 2: A Hod.

Theorem 1 *A single setpair (X, Y) is d-separable if and only if G contains neither a hod nor a terminal arc.*

Proof: If G contains a terminal arc then clearly X and Y are not d-separable. Suppose G contains a hod. Some vertex on the handle must be selected in any d-separator Z. But then the box path of the hod is unblocked and there is now no way to block it. To see this note that the box path has no non-colliders and exactly one collider c; moreover, all vertices in the handle are descendants of c and at least one of them is in Z.

On the other hand assume that G contains no hod and no terminal arc. Consider any panoramic collider c. Observe that c cannot be the ancestor of any terminal, otherwise we have a hod. Let W be the set of panoramic colliders and all their descendants. Any terminal path that contains a vertex of W must contain a collider within W. Consequently, for any proposed d-separator Z, we see that $Z \setminus W$ blocks every terminal path that intersects W. Hence, it suffices to find a set $Z \subseteq V \setminus W$ that blocks every terminal path in $G \setminus W$.

This we can achieve as follows. For any directed terminal path P in $G \setminus W$, select its oldest non-terminal vertex to be in Z; such a vertex exists as P is not a terminal arc. For any hill path P in $G \setminus W$ select its summit vertex s to be in Z; this is a feasible choice as all internal vertices of P are non-terminals so s is a non-terminal. Let $\hat{Z}$ consist of the vertices in Z and all of their non-terminal ancestors. We claim that $\hat{Z}$ is a d-separator.

Thus, it suffices to show that every terminal path in $G \setminus W$ that contains a collider is blocked by $\hat{Z}$. Take such a path P and consider one of its colliders c. Assume that c has parents v_1 and v_2 in P. Now c is a non-panoramic collider as $c \notin W$, so at least one of its parents, say v_1, is not a terminal. If neither c nor any of its descendants are in $\hat{Z}$ then P is blocked. So we may assume that c is in $\hat{Z}$. But v_1 is an ancestor of c and, thus, is also in $\hat{Z}$ by definition. But v_1 is a non-collider on P, so the path is blocked. $\Box$

Thus we have a characterization theorem for d-separability. If X and Y

consist of single vertices then the characterization theorem has a particularly simple form.

Corollary 1 *If X and Y are singleton-sets then G contains a d-separator if and only if there are no terminal arcs.*

Proof: If X and Y are both singletons then there are no hods as a hod requires three terminal vertices. The result is then immediate from Theorem 1. □

Corollary 1 indicates why for graph bases we are interested in non-adjacent vertices. Furthermore, Theorem 1 allows us to test in linear time (in the number of edges) whether or not G contains a d-separator for X and Y, and if so to find such a separator.

Theorem 2 *For a single setpair (X, Y), there is a linear time algorithm to obtain a d-separator if one exists.*

Proof: First we check every arc to see if it is a terminal arc, that is it connects a vertex in X to a vertex in Y. If there is such an arc then there can be no d-separator by Theorem 1. Next we need to test whether there is a hod in G. Examining all the parents of a vertex we can test if it is a panoramic collider. We can find all panoramic colliders $\mathcal{P}$ in linear time as we need examine each arc only once. Given the panoramic colliders we need to see if any of them have directed paths to a terminal vertex. To do this we just need to find the reachability set from $\mathcal{P}$ and this can also easily be done in linear time. If this reachability set contains a terminal then we have found a hod and so, by Theorem 1, there is no d-separator.

So assume we find no hod or terminal arc. Then there is a d-separator. Indeed, by the proof of Theorem 1, we may simply choose $\hat{Z}$, that is, all non-terminal ancestors of the oldest vertex in a hill path or a directed terminal path. First find the set of vertices that are ancestors of terminals in Y (respectively X). If a non-terminal vertex is ancestor of both a vertex in X and a vertex in Y then it is (an ancestor of) a the oldest vertex in a hill path. If it is an ancestor of only vertices in Y (respectively X) then test if it is a child of a terminals in X (respectively Y); if so, it is the oldest vertex in a terminal path. □

Observe that the d-separator $\hat{Z}$ may be extremely large. As stated, though, it may be desirable to obtain a minimum cardinality d-separator. This we can do by the techniques of Lauritzen et al [12]. To describe their result, let V^+ consist of those vertices in G that have a directed path to some vertex in $X \cup Y$. Let G^+ be the graph induced by V^+. Note that the sets of *terminals* X and Y are trivially subsets of V^+. We then need the following definition:

A *d-cut* is a set $S \subseteq V^+$ such that
(i) S is a vertex-cut between X and Y in G^+. That is, the components of $G^+ \setminus S$ can be partitioned into parts $\mathcal{C}_X$ and $\mathcal{C}_Y$ containing X and Y, respectively.
(ii) For $v \in S$, either $\delta^-(v) = \{u : (u, v) \in A\} \subseteq \mathcal{C}_X \cup S$, or $\delta^-(v) \subseteq \mathcal{C}_Y \cup S$.

An example of a d-cut is shown in Figure 3, where the vertices in X are shaded circles and vertices in X are shaded squares.

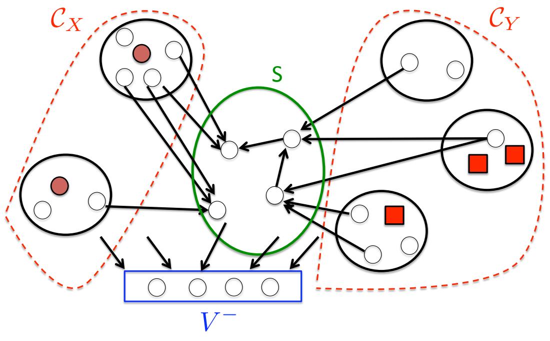

Figure 3: A d-cut.

Now d-cuts and d-separators are closely related via the following theorem, whose proof we include for completeness.

Theorem 3 *[12] For a single setpair, a set $Z \subseteq V^+$ is a d-separator if and only if it is a d-cut.*

Proof: Let $Z \subseteq V^+$ be a d-cut. Take any path P between some terminals $x \in X$ and $y \in Y$. Suppose first that P is contained in G^+. Then, by (i), Z cuts X and Y in G^+. Hence, the path P must intersect Z. Let $P \cap Z = \{z_1, \ldots, z_k\}$, numbered according to their order along P. If any z_i is a non-collider on P then Z separates the path. So assume that every z_i is a collider on P. This implies that z_i and z_{i+1} cannot be adjacent along P, for any $1 \leq i < k$. Therefore the two parents of z_1 on P are both in $\mathcal{C}_X \subset \mathcal{C}_X \cup S$, by (ii) and the fact that P begins in $\mathcal{C}_X$. But then the two parents of z_2 on P are also both in $\mathcal{C}_X$, etc. Consequently, the path can never can never enter $\mathcal{C}_Y$, a contradiction. So Z does d-separate the path P.

Next, suppose that P uses a vertex v in V^-. Observe that v has no directed path to any vertex in G^+ (otherwise it has a directed path to $X \cup Y$). It follows that any subpath of P within V^- contains a collider. So, as $Z \cap V^- = \emptyset$, we see that Z again d-separates P.

On the other hand, let $Z \subseteq V^+$ be a d-separator. First we need to show (i), that Z is a vertex-cut between X and Y in G^+. If not, there is a path P between $x \in X$ and $y \in Y$ in $G^+ - Z$. As Z is a d-separator, P must contain a collider c for which none of the descendants of c are in Z. We assume P to be a terminal path in $G^+ - Z$ with the fewest colliders having this property. As c is in G^+ it has a directed path Q to $X \cup Y$. Without loss of generality, let Q go to $y' \in Y$. Then no vertex of Q is in Z. Let P' be the subpath of P from x to c. Then $P' \cup Q$ is a path (from $x \in X$ to $y' \in Y$) in $G^+ - Z$.

If $P' \cup Q$ contains no collider then it is a hill path. But then $Z \cap P' \neq Q$, as Z is a d-separator, a contradiction. Thus, $P' \cup Q$ must contain a collider none of whose descendants are in Z. However, it has one less collider with this property than P, a contradiction. Thus (i) holds.

So $G^+ - Z$ consists of the components $S_1, \ldots, S_k$ where none of the components contains both X and Y terminals. We next need to show (ii). Towards this goal, we build an auxiliary graph H as follows. Let the vertices of H be $\{S_1, \ldots, S_k\}$. There is an edge (S_i, S_j) in H if and only if in G there are three vertices $v_i \in S_i, v_j \in S_j, z \in Z$ where $\{v_i, v_j\} \subseteq \delta^-(z)$.

Let $\mathcal{C}_X$ consist of those S_i that lie in components of H that contain some vertex of X. We claim that $\mathcal{C}_X$ contains no vertex of Y. If so we are done as then $\mathcal{C}_Y = V^+ - \mathcal{C}_X - Z$ is non-empty, and, by construction, for any $v \in Z$ we must have $\delta^-(v) \subseteq \mathcal{C}_X \cup Z$ or $\delta^-(v) \subseteq \mathcal{C}_Y \cup Z$.

So let's prove the claim. If $\mathcal{C}_X$ contains $y \in Y$, we can build a path P between from some $x \in X$ to y that goes through the components $S_1, \ldots, S_t$, say, where there is a $z_i \in Z$ with in-neighbours $u_i \in S_i$ and $v_{i+1} \in S_{i+1}$. Let P be such a terminal path containing a minimal number of colliders outside of Z. Consider the subpath P_i of P in S_i from v_i to u_i. Suppose that P_i contains a collider c. There is a directed path Q from c to $X \cup Y$, say to $x' \in X$. If Q goes through Z then c contains a descendant in Z. If Q does not go through Z then it is a directed path entirely contained within S_i. Now replacing $P[x, c]$ by $Q[x', c]$ gives a terminal path with fewer colliders than P outside Z. Repeating this argument gives that every collider on P is in Z or has a decendant in Z. This contradicts the fact that Z is a d-separator. □

So, to find a minimum cardinality d-separator, we simply need to obtain a minimum cardinality d-cut in G^+. To do this we create an undirected auxiliary graph $\mathcal{G}$ called the *moral graph* [13]. This is simply the undirected supergraph of G^+ obtained by adding a edge between each pair of vertices that share a common non-terminal child in G. We then find a minimum $X - Y$ cut in $\mathcal{G}$. This finds an optimal d-cut which, in turn, produces a minimum cardinality d-separator. It is straightforward to verify (albeit perhaps surprising) that this approach works; see Tian et al. [26] for details on this method.

5 D-Super Separation for Multiple Setpairs

In this section, we consider the problem of finding a minimum cost subset of vertices Z that contains a d-separator for each of the setpairs $(X_1, Y_1), \ldots, (X_k, Y_k)$; we call this the *minimum cost d-super-separator problem.* Observe that there is a trivial k-approximation algorithm[5] for this d-super-separator problem. Simply find a minimum cost d-separator Z_i^* for each setpair (X_i, Y_i) as described in Section 4, and then output $Z = \cup_{i=1}^k Z_i^*$. Clearly Z is a d-super-separator whose

[5]An α-approximation algorithm is a polynomial time algorithm that is guaranteed, on any problem instance, to output a feasible solution whose objective value is within an α multiplicative factor of the optimal solution value.

cost is at most a factor k larger than that of the optimal solution. Here we show that a much better approximation algorithm exists. Specifically, we give an $O(\log^2 k)$ approximation guarantee. The focus upon approximation algorithms is necessary; we show in Theorem 5 that the minimum cost d-super-separator problem is NP-complete.

As $G = (V, A)$ is acyclic there is an acyclic ordering of the vertices. When constructing the ordering, given a comparison between two incomparable vertices, we may assume that a terminal vertex is placed before a non-terminal vertex. For clarity of presentation, here we consider the case of singleton setpairs, that is $X_i = \{x_i\}$ and $y_i = \{y_i\}$ for all i; the proof of the general case is similar. Without loss of generality, for each i, we label the terminals $\{x_i, y_i\}$ such that x_i appears before y_i in the acyclic ordering. Furthermore, we label the setpairs so that y_i appears before y_{i+1} in the ordering.

Let V_i consist of y_i and all of the vertices before it in the ordering (including non-terminal vertices). This vertex set underlies the undirected moral graphs for commodity i. Let the moral graphs be $\mathcal{G}_i = (V_i, E_i)$. By the ordering, we have $V_1 \subset V_2 \subset \cdots \subset V_k$ and thus $E_1 \subset E_2 \subset \cdots \subset E_k$. Recall, to find an optimal d-separator for commodity i we need to find a minimum cut in the moral graph $\mathcal{G}_i$ [12]. But how can we find a low cost d-super-separator? Our approach is to model this problem as an integer program and then relax it to a linear program (LP). We solve the linear program and round the resultant fractional solution to an integral solution using the ball-growing method of Garg et al. [7]. Our approximation guarantee will follow as we show that this rounding can be achieved at a reasonable cost.

We remark that there are several technical difficulties in applying the ball-growing method to the d-super-separator problem. First the method finds edge cuts in undirected graphs whereas we have a directed graph and are looking for vertex separators. This is not a significant barrier. As we have seen, the consideration of moral graphs allows us to focus upon undirected graphs. Furthermore, a careful modification to the technique allows us to search for vertex separators. Much more problematic, though, is that the ball-growing method only applies to a single graph. But we have many graphs as we are working with the set of moral graphs $\{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$. Consequently, the key technical contribution of this section involves showing how to exploit the special structure inherent in these moral graphs to extend the ball-growing method to multiple graphs.

So now let's formulate the linear program relaxation. Let $\mathcal{P}_i$ be the set of paths from x_i to y_i in the moral graph $\mathcal{G}_i$. Then an integral solution must select at least one vertex from every path in $\mathcal{P}_i$. In our LP relaxation, we may fractionally select vertices so that in total at least one unit worth of vertices are chosen from amongst the interior vertices $I(P)$ of any path $P \in \mathcal{P}_i$. Thus, given a cost c_v for each vertex v, we obtain the following linear program:

$$\begin{aligned} \min \sum_{v \in V} c_v \cdot d_v & \\ \sum_{v \in I(P)} d_v \geq 1 & \qquad \forall P \in \mathcal{P}_i \quad \forall i \\ d_v \geq 0 & \qquad \forall v \in V \\ d_{x_i}, d_{y_i} = 0 & \qquad \forall i \end{aligned}$$

This linear program has an exponential number of constraints but it can be solved in polynomial time via a separation oracle using the ellipsoid method [9] or via a compact network flow formulation.

Take an optimal solution $\mathbf{d}$ to the linear program. Then the optimum LP value $V^* = \sum_{v \in V} c_v \cdot d_v$ gives a lower bound on the cost of the optimal d-superseparator. Furthermore, we can view $\mathbf{d}$ as inducing a distance function in the moral graphs. Specifically, for any path $P = \{v_0, v_1, \ldots, v_r\}$ in $\mathcal{G}_i$, let $l_i(P) = \sum_{v \in I(P)} d_v$ be the *length* of the path P. Then the distance $d_i(v_0, v_r)$ of v_r from v_0 is the minimum of $l_i(P)$ over all paths P between v_0 and v_r in $\mathcal{G}_i$. Thus $d_i(x_i, y_i) \geq 1$ by imposition.

Given this distance function the rounding algorithm proceeds in phases as follows. It will become clear that it is extremely important that the phases of the algorithm run in reverse acyclic order (the method will fail otherwise). Consequently, for the first phase we set $i = k$. Let B_{x_k} be the set of vertices (*ball*) of distance less than r_k (to be defined) from x_k in $H_k = \mathcal{G}_k$. We say that a vertex v is on the boundary $\mathtt{b}(B_{x_k})$ if in H_k we have

$$d(x_k, v) \leq r_k < d(x_k, v) + d_v \tag{1}$$

Now set $q_v = \frac{r_k - d(x_k, v)}{d_v}$. By Inequality (1), it follows that $0 \leq q_v < 1$. Thus, we will view a boundary vertex $v \in \mathtt{b}(B_{x_k})$ as being a q_v *fraction* in B_{x_i}. Observe that each vertex v contributes an amount $c_v \cdot d_v$ to the linear program value; we call this amount the *area* of vertex v. The *volume* of B_{x_i} is then

$$vol(B_{x_i}) = \sum_{v \in B_{x_i}} q_v \cdot c_v \cdot d_v$$

We define B_{y_k} and its volume in an analogous fashion. Then amongst B_{x_k} and B_{y_k}, we define B_k to be the one of smaller volume.

Similarly, we need to define a ball B_i for each $i < k$. We do this recursively. To do this set $i := i - 1$ and consider $H_i = \mathcal{G}_i \setminus \cup_{j=i+1}^{k} \mathtt{b}(B_j)$. If x_i and y_i are already disconnected in H_i then let $B_i = \emptyset$. Otherwise, define B_{x_k} and B_{y_k} to be the set of vertices of distance less than r_i from x_i and y_i, respectively, in H_i. Again we then let B_i be the ball with smaller volume amongst B_{x_i} and B_{y_i}. This process is repeated until $i = 0$.

The final output is $W = \cup_{i=1}^{k} \mathtt{b}(B_i)$. Now we show W is a d-super-separator, then we will prove it has low cost.

Claim 1 *The set of vertices W forms a d-super separator for $(x_1, y_1), \ldots, (x_k, y_k)$.*

Proof: Take any pair (x_i, y_i). Observe that $V_i \cap \cup_{j=i}^{k} \mathtt{b}(B_j) \subseteq W = \cup_{j=1}^{k} \mathtt{b}(B_j)$ is a separator between x_i and y_i in the moral graph $\mathcal{G}_i$. This must be the case, since if x_i and y_i are not already separated in $\mathcal{G}_i$ by $V_i \cap \cup_{j=i+1}^{k} \mathtt{b}(B_j)$ then, by construction, they will be separated when we add $\mathtt{b}(B_i)$. □

Thus W is a feasible solution to our problem. The cost of this solution is

$$c(W) = \sum_{i=1}^{k} \sum_{v \in \mathtt{b}(B_i)} c_v \cdot d_v$$

Moreover, it follows from the ball-growing method [7] (see also [28, 30]) that, for each i we can efficiently obtain a radius $r_i \leq \frac{1}{2}$ such that the corresponding ball B_i satisfies the property:

$$\sum_{v \in \mathtt{b}(B_i)} c_v \cdot d_v \ \leq \ 2 \ln(k+1) \cdot \left(vol(B_i) + \frac{1}{k} V^* \right) \tag{2}$$

Consequently, to quantify the performance guarantee of the algorithm we must bound the sum of the volumes of the B_i. Key to doing so, is the following claim that ensures the balls shrink sufficiently in each iteration.

Claim 2 *If x_i and y_i lie in the same component U of H_i then $vol(B_{x_i}) + vol(B_{y_i}) \leq \sum_{v \in U} c_v \cdot d_v$.*

Proof: Suppose $v \in B_{x_i} \cap B_{y_i}$ and that v contributes $q_v^{x_i} \cdot c_v \cdot d_v$ and $q_v^{y_i} \cdot c_v \cdot d_v$ to these volumes, respectively. We claim that $q_v^{x_i} + q_v^{y_i} \leq 1$. Suppose not, then

$$\begin{aligned} d_i(x_i, y_i) &\leq d_{H_i}(x_i, v) + d_v + d_{H_i}(v, y_i) \\ &< d_{H_i}(x_i, v) + (q_v^{x_i} + q_v^{y_i}) d_v + d_{H_i}(v, y_i) \\ &= (d_{H_i}(x_i, v) + q_v^{x_i} \cdot d_v) + (q_v^{y_i} \cdot d_v + d_{H_i}(v, y_i)) \\ &= r_i + r_i \\ &\leq 1 \end{aligned}$$

This contradicts the fact that $d_i(x_i, y_i) \geq 1$. Thus $q_v^{x_i} + q_v^{y_i} \leq 1$ and so v contributes at most $c_v \cdot d_v$ to $vol(B_{x_i}) + vol(B_{y_i})$. □

Corollary 2 *If $B_i \subseteq B_j$ for some $j > i$ then $vol(B_i) \leq \frac{1}{2} vol(B_j)$.*

Proof: We may apply Claim 2 where $U = B_j \backslash bd(B_j)$. Since each non-boundary vertex is completely in B_j, we have

$$\begin{aligned} vol(B_j) &= \sum_{v \in B_j} q_v \cdot c_v \cdot d_v \\ &\geq \sum_{v \in B_j \backslash bd(B_j)} c_v \cdot d_v \\ &\geq vol(B_{x_i}) + vol(B_{y_i}) \\ &\geq 2 \cdot vol(B_i) \end{aligned}$$

as desired. □

Applying Corollary 2 we obtain

Lemma 1 $\sum_{i=1}^{k} vol(B_i) \leq V^* \cdot (1 + \log k)$

Proof: Take the regions $B_1, \ldots, B_k$ and designate them small or large according to the amount of volume they encompass. Specifically, set $\mathcal{S} = \{i \in [k] : vol(B_i) \leq \frac{1}{k} \cdot V^*\}$ and $\mathcal{L} = \{i \in [k] : vol(B_i) > \frac{1}{k} \cdot V^*\}$. Thus

$$\sum_{i=1}^{k} vol(B_i) \quad = \quad \sum_{i \in \mathcal{S}} vol(B_i) + \sum_{i \in \mathcal{L}} vol(B_i)$$

But clearly,

$$\sum_{i \in \mathcal{S}} vol(B_i) \;\leq\; |\mathcal{S}| \cdot \frac{V^*}{k} \;\leq\; V^* \tag{3}$$

Observe that the regions $B_1, \ldots, B_k$ form a laminar set. Therefore a vertex may be in many regions. However, by Corollary 2, those regions that containing a specific vertex v must iteratively shrink in volume by a factor at least two. Thus a vertex can be in at most $\log k$ large regions. Hence,

$$\begin{aligned}
\sum_{i \in \mathcal{L}} vol(B_i) \quad &\leq \quad \sum_{i \in \mathcal{L}} \sum_{v \in B_i} c_v d_v \\
&= \quad \sum_{v} \sum_{i \in \mathcal{L} : v \in B_i} c_v d_v \\
&\leq \quad \log k \cdot \sum_{v} c_v d_v \\
&= \quad V^* \cdot \log k
\end{aligned} \tag{4}$$

Combining Inequalities (3) and (4) we get $\sum_{i=1}^{k} vol(B_i) \leq V^* \cdot (1 + \log k)$. □

Putting this all together gives the desired approximation guarantee.

Theorem 4 *There is an $3 \log^2 k$-approximation algorithm for the minimum cost d-super separator problem.*

Proof: We have

$$\begin{aligned}
c(W) \quad &= \quad \sum_{i=1}^{k} \sum_{v \in \mathtt{b}(B_i)} c_v \cdot d_v \\
&\leq \quad \sum_{i} 2 \ln(k+1) \cdot \left(vol(B_i) + \frac{1}{k} V^* \right) \qquad \text{[by Inequality (2)]} \\
&\leq \quad 2 \ln(k+1) \cdot \left(V^* + \sum_{i} vol(B_i) \right) \\
&\leq \quad 2 \ln(k+1) \cdot (2 + \log k) \cdot V^* \qquad \text{[by Lemma 1]}
\end{aligned}$$

Now we may assume that $k \geq 19$. Otherwise, we simply get a factor $k < 3\log^2 k$ guarantee by applying the trivial approximation algorithm. Thus

$$\begin{aligned} c(W) &\leq 2\ln(k+1)\cdot(2+\log k)\cdot V^* \\ &\leq 3\log^2 k \cdot V^* \\ &\leq 3\log^2 k \cdot \text{OPT} \end{aligned}$$

The result follows. □

As remarked, no polynomial time algorithm for the minimum cost d-super-separator problem can exist unless $P = NP$.

Theorem 5 *The minimum cost d-super-separator problem is NP-complete*

Proof: The decision problem is to find a d-super-separator of cost at most C, for a given C. This problem is in NP. Given $\{Z_1, Z_2, \ldots, Z_k\}$ simply check that Z_i is a d-separator for the setpair (X_i, Y_i). This can be done in polynomial time via the techniques shown in Section 4. Then check that the cost of $Z = \cup_i Z_i$ is at most C.

To show this problem is NP-complete we give a reduction from Vertex Cover: given an undirected graph $H = (U, E)$ and an integer k, is there a set $S \subseteq V$ of cardinality at most k such that every edge has an endpoint in S? To do this we create a causal graph $G = (V, A)$ as follows. The graph is an arborescence with root vertex r. For each vertex $v \in U$ we have two vertices $v, \hat{v} \in V$ along with the two arcs $(r, \hat{v})$ and $(\hat{v}, v)$. For each edge $(u, v) \in E$ we have a singleton setpair (X_e, Y_e) where $X_e = \{u\}$ and $Y_e = \{v\}$. Finally, we set the cost of r to be infinite and the cost of the other vertices to be one. The result follows from the fact that an edge (u, v) is covered if and only if we select either $\hat{u}$ or $\hat{v}$ in the d-super-separator. □

6 D-Separation for Multiple Setpairs

We now examine the *Multi-Setpair d-Separator Problem.* Given a collection of setpairs $(X_1, Y_1), \ldots, (X_k, Y_k)$, is there a set Z that d-separates X_t and Y_t, for all $1 \leq t \leq k$?

6.1 A Hardness Result

We will prove in this section that the d-Separator Problem is NP-complete for multiple setpairs. Indeed, it is hard even when there are just $k = 5$ setpairs. This hardness proof is via a reduction from the Planar 3-SAT problem [14].
The Planar 3-SAT Problem: Given n Boolean variables $a_1, a_2, \ldots, a_n$ and m clauses $C_1, \ldots, C_m$, where each clause contains exactly three literals. Furthermore, the bipartite clause-variable incidence graph, $\mathcal{H}$, is planar[6]. Does there

[6] $\mathcal{H}$ contains a vertex for each variable and a vertex for each clause; a clause vertex is adjacent to a variable vertex if and only if the variable appears in that clause.

exist a TRUE-FALSE assignment of the variables such that *every* clause is satisfied?

Theorem 6 *For five or more setpairs, it is NP-hard to determine whether a graph contains a d-separator.*

Proof: Given a planar 3-SAT instance $\mathcal{S}$, we construct a causal graph G and setpairs $\{(\mathcal{X}_1, \mathcal{Y}_1), \ldots, (\mathcal{X}_5, \mathcal{Y}_5)\}$ such that $\mathcal{S}$ has a satisfying assignment if and only if G has a d-separator. The construction of G involves several gadgets. Each gadget will contain a collection of terminal-pairs. However, this will produce a polynomial number of terminal pairs as there will be a polynomial number of gadgets (and possibly several terminal pairs in each gadget). Thus, we cannot allow each terminal-pair to form their own individual setpair. Instead, we will group the terminal-pairs together to form our four setpairs. Observe that each setpair will then induce numerous new terminal pairings that must be considered in addition to the original terminal-pairs in the gadgets.

So let's begin by describing the gadgets. The simplest gadget is a *dummy terminal*. The dummy terminal is a vertex d that represents a path between a terminal-pair, (X_d, Y_d), that consists of a single collider. An example is shown in Figure 4, where we represent the dummy vertex d by a clear square. (Real terminals are represented by shaded squares, and non-terminals by circles). The reader should be warned that, here, we use capital letters to denote X_d and Y_d even though they are just singleton vertices (this is because such a terminal pair will subsequently be used in Corollary 7 to form a singleton setpair).

For each variable a_i, there will be a *variable gadget*, as shown in Figure 5. The gadget contains two terminal pairs, (X_i, Y_i) and $(\hat{X}_i, \hat{Y}_i)$. We may assume that variable a_i appears in n_i clauses, say $\{C_{i,1}, C_{i,2}, \ldots, C_{i,n_i}\}$.

For each clause $C_{i,r}$, where $1 \leq r \leq n_i$, we have a *clause-variable path* $\{X_i, a_{i,r}, d_{i,r}, \bar{a}_{i,r}, Y_i\}$. This is a hill path with summit at $d_{i,r}$. Here $d_{i,r}$ is a dummy terminal and $a_{i,r}, \bar{a}_{i,r}$ are non-terminals. Since this is a hill path at least one of $a_{i,r}, \bar{a}_{i,r}$ must be chosen in a separator, as (dummy) terminals cannot be selected. We would like exactly one to be chosen and for this choice to be consistent for all r, as this would give a natural way for a d-separator to produce a variable assignment. To do this we use the other terminal pair $(\hat{X}_i, \hat{Y}_i)$. Specifically, the gadget contains a *variable path* $\{\hat{X}_i, a_i, \hat{d}_i, \bar{a}_i, \hat{Y}_i\}$, where $\hat{d}_i$ is a dummy terminal and $a_i, \bar{a}_i$ are colliders on the variable path. Thus the variable path is blocked by a separator unless both a_i (or a descendant) and $\bar{a}_i$ (or a descendant) are in the separator.

To complete the description of the variable gadget, we add an arc from a_i to each $a_{i,r}$, and an arc from $\bar{a}_i$ to each $\bar{a}_{i,r}$. It follows that $a_{i,r}$ and $\bar{a}_{i,r}$ cannot both be selected in the d-separator or the variable-path is unblocked. Similarly, $a_{i,r}$ and $\bar{a}_{i,r'}$ cannot both be selected in the d-separator, for any $r \neq r'$. Thus, the separator contains either all the $a_{i,r}$ and none of the $\bar{a}_{i,r}$, or all the $\bar{a}_{i,r}$ and none of the $a_{i,r}$. The former case will correspond to setting a_i to TRUE and the latter case will correspond to setting $\bar{a}_i$ to TRUE. Hence, we obtain a variable assignment mechanism.

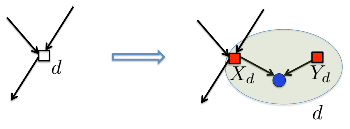

Figure 4. A Dummy Terminal d.

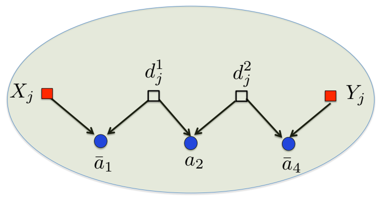

Figure 6. A Clause Gadget: $C_j = a_1 \vee \bar{a}_2 \vee a_4$.

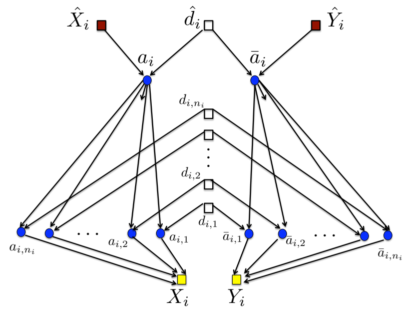

Figure 5. A Variable Assignment Gadget.

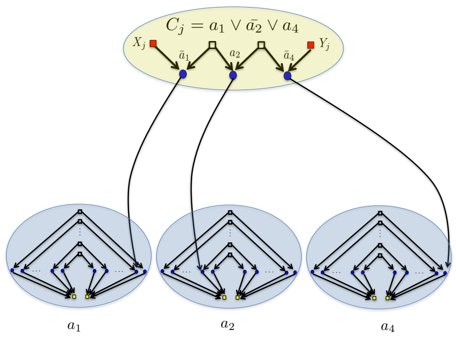

Figure 7. Joining the Gadgets.

So we have a mechanism to produce variable assignments. We need them to satisfy the clauses, so, towards this goal, we have a clause gadget for each clause C_j. This gadget contains one terminal-pair (X_j, Y_j) and a single path $\{X_j, l_{j,1}, d_j^1, l_{j,2}, d_j^2, l_{j,3}, Y_j\}$ between them. Here d_j^1, d_j^2 are dummy terminals and $l_{j,1}, l_{j,2}, l_{j,3}$ are colliders named after the *negations* of the three literals in the clause. For example, the gadget for a clause $C_j = a_1 \vee \bar{a}_2 \vee a_4$ is shown in Figure 6.

The motivation behind this gadget is as follows. The path is blocked unless each collider vertex (or one if its descendants) is selected in the separator. Using this construction, the selection of a collider vertex (or a descendant) will correspond to the selection of the corresponding literal in C_j. Thus the clause will be satisfied unless none of the literals in the clause are in the variable assignment, that is, (decendants of) all three negations of these literals are in the separator. To enforce this property we need to show how the variable and clause gadgets connect up. This is simple. For each literal $\bar{a}_i$ in C_j we add an arc from the collider vertex a_i in the clause gadget for C_j to the vertex $a_{i,j}$ in the ith variable gadget; similarly, for each literal a_k in C_j we add an arc from the collider vertex

$\bar{a}_k$ in the clause gadget for C_j to the vertex $\bar{a}_{k,j}$ in the ith variable gadget. This is shown in Figure 7.

This completes the description of the directed graph G. We remark that it is easy to see that G is also acyclic.

Next we describe how to choose the five setpairs $(\mathcal{X}_t, \mathcal{Y}_t), 1 \leq t \leq 5$. To define the these setpairs, recall that the clause-variable incidence graph $\mathcal{H}$ is planar. Let $\mathcal{H}'$ be the auxiliary graph created taking each clause vertex in $\mathcal{H}$ and adding a clique on its neighbourhood. Thus, in $\mathcal{H}'$, two variable vertices are adjacent if and only if their corresponding variables appear together in some clause.

Claim 3 *The auxiliary graph $\mathcal{H}'$ is also planar.*

Proof: Each clause vertex in $\mathcal{H}$ has three neighbours. Thus each clique we add is simply a triangle. Consequently, it is straightforward to obtain a planar drawing of $\mathcal{H}'$ given a planar drawing of $\mathcal{H}$. $\square$

Thus, by the Four-Colour Theorem ([1], [2]), we can assign the variables and clauses to four colour groups such that (i) variables of the same colour never appear in the same clause, and (ii) a variable has a different colour than any clause it is contained in.

Observe that we have four types of terminal: standard terminals in variable gadgets (the X_i and Y_i); standard terminals in clause gadgets (the X_j and Y_j); "hat" terminals in variable gadgets (the $\hat{X}_i$ and $\hat{Y}_i$); dummy terminals in variable gadgets and clause gadgets. We need to partition these four types of terminal into the five setpairs. To define the first four setpairs, we will create one setpair for each colour class, $1 \leq p \leq 4$. Specifically, let

$$\begin{aligned} \mathcal{X}_p &= \{X_i : \texttt{col}(a_i) = p\} \cup \{X_j : \texttt{col}(C_j) = p\} \cup \{\hat{X}_i : \texttt{col}(a_i) = p+1\} \\ \mathcal{Y}_p &= \{Y_i : \texttt{col}(a_i) = p\} \cup \{Y_j : \texttt{col}(C_j) = p\} \cup \{\hat{Y}_i : \texttt{col}(a_i) = p+1\} \end{aligned}$$

Finally we have one setpair containing all the dummy terminals.

$$\mathcal{X}_5 = \{X_d : d \text{ is a dummy terminal}\} \qquad \mathcal{Y}_5 = \{Y_d : d \text{ is a dummy terminal}\}$$

This completes the construction. We now have to show that a d-separator exists in G if and only if a satisfying assignment exists for $\mathcal{S}$. Before doing this, we emphasize again that each setpair induces many terminal pairs. Paths between such terminal pairs (called *terminal paths*) may cross between gadgets and, indeed, can have endpoints in different gadgets. Thus we will need to be very careful when examining whether or not a proposed set Z is a d-separator.

Let's start with the easy direction. Suppose that $\mathcal{S}$ is not satisfiable. As argued above, the variable gadgets ensure that any d-separator must induce a variable assignment. Since $\mathcal{S}$ is not satisfiable this variable assignment must violate some clause C_j. The corresponding clause-path in the jth clause gadget will be unblocked. Thus no d-separator can exist.

On the other hand, suppose that $\mathcal{S}$ is satisfiable. Then take a satisfying assignment. If a_i is TRUE then select all the $a_{i,r}$ vertices in the clause-variable paths, select all a_i vertices in a clause path, and select a_i in the variable path. Similarly, if a_i is FALSE then select all the $\bar{a}_{i,r}$ vertices in the clause-variable paths, select all $\bar{a}_i$ vertices in a clause path and select $\bar{a}_i$ in the variable path. Call the set of selected vertices Z.

We need to show that every terminal path P is blocked by Z. A terminal path P can take on many forms, for example, depending upon the types of the two terminals at its ends and how many gadgets the path crosses. So, in order to deal with the multitude of possible cases, we present a few structural claims concerning terminal paths that will help simplify the analysis.

Claim 4 *Every terminal path in* $(\mathcal{X}_5, \mathcal{Y}_5)$ *is blocked.*

Proof: If a terminal path uses a dummy terminal Y_d then it must also use the collider c between X_d and Y_d. Since c is not in Z and has no descendants, the path is blocked. □

Claim 5 *Any path that internally contains some* X_i *or some* Y_i *is blocked.*

Proof: Without loss of generality, assume that P uses X_i as an internal vertex. As X_i has no out-neighbours, it must be a collider on P. But X_i is not in Z. The path is, therefore, blocked. Similarly for Y_i. □

Claim 6 *No terminal path can internally contain some* $\hat{X}_i, \hat{Y}_i, X_j$ *or* Y_j.

Proof: The terminal vertices $\hat{X}_i, \hat{Y}_i, X_j$ and Y_j are the endpoints of variable paths or clause paths. By construction, these vertices all have exactly one neighbour, so cannot be internal vertices in paths. □

Claim 7 *Any terminal path that contains both* $a_{i,r}$ *and* $\bar{a}_{i,r'}$ *is blocked.*

Proof: As $a_{i,r}$ and $\bar{a}_{i,r'}$ are non-terminals they must be internal on the path. Their only out-neighbours are X_i and Y_i, respectively. By Claim 5, X_i and Y_i are not used internally on the path. Thus, either X_i and Y_i are not used on P or they are endpoints of P. This gives three cases.
(i) X_i and Y_i are both endpoints of P. Then we may assume, without loss of generality, that $a_{i,r}$ and $\bar{a}_{i,r'}$ are adjacent to X_i and Y_i, respectively, on P. But then $a_{i,r}$ and $\bar{a}_{i,r'}$ are both non-colliders on P. Since one of them is also in Z, the path is blocked.
(ii) Exactly one of X_i and Y_i is an endpoint of P. We may assume that X_i is the endpoint and that $a_{i,r}$ is adjacent to X_i on P. Thus $a_{i,r}$ is a non-collider. So if $a_{i,r}$ is in Z then the path is blocked. If not, $\bar{a}_{i,r'}$ must be in Z. Moreover, Y_i is not on the path, and so $\bar{a}_{i,r'}$ must be a collider on P. Now observe that $\bar{a}_{i,r'}$ has at most three other neighbours, namely $d_{i,r'}$, $\bar{a}_i$ in the variable gadget, and $\bar{a}_i$ in the clause gadget corresponding to r'. Thus at least one of the two $\bar{a}_i$ is a non-collider on P. But these are both in Z as $\bar{a}_{i,r'}$ is, so the path is blocked.
(iii) Neither X_i nor Y_i is an endpoint of P. Then $a_{i,r}$ and $\bar{a}_{i,r'}$ must both be colliders on the path P. We know one of them, say $a_{i,r}$, is not in Z. The only descendent of $a_{i,r}$, namely X_i, is also not in Z. Thus, the path is blocked. □

Claim 8 *Any terminal path that contains an arc* $(a_i, a_{i,r})$ *and uses* $a_{i,r}$ *as a collider is blocked.*

Proof: Recall that the vertex a_i refers to a vertex in a variable gadget and in a clause gadget. The proof is the same in either case. Now a_i cannot be a collider as the arc $(a_i, a_{i,r})$ is used on the path P. Thus, if a_i is in Z the path is blocked. So $a_i \notin Z$. Therefore, $a_{i,r}$ is not in Z either. But $a_{i,r}$ is a collider on P and because X_i is the only descendent of $a_{i,r}$, the path must be blocked. □

A similar argument gives:

Claim 9 *Any terminal path that contains an arc* $(\bar{a}_i, \bar{a}_{i,r})$ *and uses* $\bar{a}_{i,r}$ *as a collider is blocked.* □

Corollary 3 *Any terminal path for the setpair* $(\mathcal{X}_t, \mathcal{Y}_t)$*,* $1 \leq t \leq 4$*, is blocked.*

Proof: Take a terminal path P. We have two possibilities: either P is entirely within one gadget or it uses at least two gadgets. In neither case can these be dummy gadgets. Consider the former case. As P is entirely within a gadget we have the following possibilities:
(i) It is a clause path from X_j to Y_j. As we had a satisfying assignment this path is blocked.
(ii) It is from $\hat{X}_i$ to $\hat{Y}_i$. If it is the variable path then it is blocked as we have a consistent variable assignment. Otherwise the path must use some $a_{i,r}$ and some $\bar{a}_{i,r'}$. Thus, by Claim 7, it is blocked.
(iii) It is from X_i to Y_i. Again the path must use some $a_{i,r}$ and some $\bar{a}_{i,r'}$ so is blocked.

Consider next the latter case. Observe that clause gadgets are only connected to variable gadgets, and vice versa. Thus, without loss of generality, P must use an arc $(a_i, a_{i,r})$ where a_i is in a clause gadget C_j and $a_{i,r}$ is in a variable gadget. Now if $a_{i,r}$ is a collider on P then, by Claim 8, it is blocked. So $a_{i,r}$ is a non-collider on P. It has only one out-neighbour, namely X_i. Thus, by Claim 5, we may assume that X_i is an endpoint of P. Let Y be the other endpoint of P. Then we have the following possibilities:
(i) $Y = Y_{j'}$ for some clause $C_{j'}$. Since variable a_i is in clause C_j, we know that $\texttt{col}(a_i) \neq \texttt{col}(C_j)$. Thus $j' \neq j$. Consequently the path P must go via the clause gadget to the variable gadget of another variable $a_{i'}$. As a_i and $a_{i'}$ are both in C_j, we must have $\texttt{col}(a'_i) \neq \texttt{col}(a_i)$. By Claim 5, P cannot go via $X_{i'}$ or $Y_{i'}$. Thus, either Claim 8 or Claim 9 applies and the path is blocked.
(ii) $Y = Y_{i'}$ for some variable $a_{i'}$. If $i \neq i'$ then, as $\texttt{col}(a_i) = \texttt{col}(a_{i'})$, the variables a_i and $a_{i'}$ are never in the same clause. Therefore, the path P must also go via the variable gadget of a third variable $a_{i''}$, where $\texttt{col}(a''_i) \neq \texttt{col}(a_i)$. The path is then blocked via a similar argument to that above. If $i = i'$ then as P is not contained within a single gadget, it must still go via such a gadget $a_{i''}$.
(iii) $Y = \hat{Y}_{i'}$ for some variable $a_{i'}$. Then $\texttt{col}(a_i) \neq \texttt{col}(a_{i'})$ and therefore $i \neq i'$. By Claim 5, P cannot go via $X_{i'}$ or $Y_{i'}$. Thus, either Claim 8 or Claim 9 applies and the path is blocked. □

Thus every terminal path is blocked. This completes the proof of Theorem 6. $\square$

6.2 Singleton Setpairs

The setpairs used in the proof of Theorem 6 are very large. So it is reasonable to ask if the multi-setpair d-separation problem can be solved efficiently if the sets in the setpairs are constrained to be small, for example, singletons. It is easy to see that the problem remains hard if there is allowed to be a large number of setpairs. Observe that the terminals in the construction came in pairs, so simply let each such pair form a singleton setpair. Thus we have:

Corollary 4 *For multiple setpairs, it is NP-hard to determine whether or not a d-separator exists even if every setpair consists of a pair of singleton vertices.* $\square$

On the other hand, positive results can be obtained if the number of setpairs is also fixed. Indeed, the other main result of this section shows that efficient algorithms are then achievable.

Theorem 7 *For multiple setpairs, there is a polynomial time algorithm for finding a d-separator (if one exists) if there are a fixed number of singleton setpairs.*

Before proving Theorem 7, we will need a few structural lemmas. Given a vertex set S, let $\mathcal{A}(S)$ be the set of vertices that are ancestors of at least one vertex in S. Similarly, let $\mathcal{D}(S)$ be the set of vertices that are descendants of at least one vertex in S. We assume that a vertex is both an ancestor and a descendant of itself. Now let T be the set of terminals.

Lemma 2 *Let Z be a d-separator. Then, for any $z \in Z$ we have that $Z \cup (\mathcal{A}(\{z\}) \setminus T)$ is a d-separator.*

Proof: Take any terminal path P. If P contains a vertex of Z as a non-collider then it clearly contains a vertex of $Z \cup (\mathcal{A}(\{z\}) \setminus T)$ as a non-collider. Hence, P is blocked by $Z \cup (\mathcal{A}(\{z\}) \setminus T)$.

So, assume that no non-collider of P is in Z. Thus there is some collider c of P such that $\mathcal{D}(\{c\}) \cap Z = \emptyset$. Suppose there is a $v \in \mathcal{D}(\{c\}) \cap (\mathcal{A}(\{z\}) \setminus T)$. But then, since v is a descendant of c and an ancestor of z, we have that $z \in \mathcal{D}(\{c\})$. Thus $\mathcal{D}(\{c\}) \cap Z \neq \emptyset$. This contradiction shows that P is blocked by $Z \cup (\mathcal{A}(\{z\}) \setminus T)$. $\square$

We immediately obtain the following corollary.

Corollary 5 *If Z is a d-separator then $\mathcal{A}(Z) \setminus T$ is a d-separator.* $\square$

Given a set of vertices W, we say that F is a *feasible augmentation* of W if $W \cup F$ is a d-separator. Lemma 2 then has the following important consequence.

Corollary 6 *Suppose that W has a feasible augmentation and that every feasible augmentation F contains some vertex in a set S. If there is an $s \in S \setminus T$ with $S \subseteq \mathcal{D}(\{s\})$ then $W \cup (\mathcal{A}(\{s\}) \setminus T)$ has a feasible augmentation.*

Proof: Let F be a feasible augmentation containing $s' \in S$. So by Lemma 2, $W \cup F \cup \mathcal{A}(s') \setminus T$ is a d-separator. Now $s' \in \mathcal{D}(s)$, and so $\mathcal{A}(s) \subseteq \mathcal{A}(s')$. Thus $W \cup (\mathcal{A}(\{s\}) \setminus T)$ has a feasible augmentation. $\Box$

Corollary 6 tells us that if we can ever find such a set S then, if our goal is simply to find any d-separator (rather than the smallest d-separator), it cannot hurt us to select $\mathcal{A}(\{s\}) \setminus T$. We will use this fact repeatedly.

Proof of Theorem 7. First we emphasize that whenever our algorithm selects a vertex s it will also select $\mathcal{A}(\{s\}) \setminus T$.

Now, we say that a path B is a *buffer path* if
(i) The endpoints b_1, b_2 of B both have out-degree one (equivalently, in-degree zero) on the path.
(ii) Every non-collider on B is a terminal. (Its colliders may or may not be terminals).
We remark that the endpoints of a buffer may be terminals of different types.

Next take any terminal path P. Call the first and last segments of P the *outer* segments, The remaining segments are termed *inner* and these form the *inner path* of P. For technical reasons, we allow for the possibility that the outer segments are empty; this allows us to assume that the endpoints of the inner path are summits within P.

Buffer paths are of interest to us because:

Claim 10 *Any terminal path whose inner segments do not form a buffer path is blocked.*

Proof: By construction, every non-collider in the inner path is the ancestor of a collider in P. As the inner path is not a buffer it contains a non-terminal vertex x that is not a collider. Let c be the collider on P that has x as an ancestor. If no vertex in $\mathcal{D}(c)$ is selected then the path is blocked. If some vertex $y \in \mathcal{D}(c)$ is selected then so must x be; hence, the path is blocked. $\Box$

By Claim 10, we need only concern ourselves with terminal paths that are either directed paths or hill paths or whose inner paths are buffers.

Suppose we find a directed terminal path P from x_i to y_i $1 \leq i \leq k$ whose oldest non-terminal is v (if no such vertex exists then clearly there can be no d-separator). Then by Corollary 6, we may select $\mathcal{A}(\{v\}) \setminus T$. Similarly, if we find a hill path between x_i and y_i whose summit s is not a terminal, then we may select $\mathcal{A}(\{s\}) \setminus T$.

Thus the only difficult case is what to do with terminal paths whose inner paths are buffers or with hill paths whose summits are terminals. It will become apparent that the latter may be viewed as a special case of the former. So let's take a terminal path P whose inner path Q is a buffer - for clarity call this an *extended* buffer path. Let the outer segments of P be P_1 and P_2; call these the *left* segment and the *right* segment, respectively If P_1 and P_2 both contain no non-terminals then P cannot be blocked. If only P_1 (respectively, P_2) contains a non-terminal then we select $\mathcal{A}(\{v\}) \setminus T$, where v is the oldest non-terminal in P_1 (respectively, P_2) .

Therefore, we only have a choice when both P_1 and P_2 contain non-terminals: we can take the oldest non-terminal in P_1 and its ancestors or take the oldest non-terminal in P_2 and its ancestors. But, there could be an exponential number of such paths so we need an efficient way to deal with such paths. To do this observe that we can classify these paths according to the type of the terminal path and the endpoints of their inner paths. There are k terminal types, and there are then $2(k-1)$ possible choices for each inner path endpoint (we allow both endpoints to be the same, in which case the extended buffer path is a hill path). Thus, there are at most $4k^3$ classes of extended buffer paths. For any extended buffer path in the same class we will always make the same choice, either alway choose from the left segment or always choose from the right segment. Therefore, there are 2^{4k^3} possible ways to make these choices. The algorithm will be run once for each of these 2^{4k^3} combinations. For a given combination, the algorithm will greedily search for unblocked extended buffer paths and block them in accordance with the combination. The algorithm runs in polynomial time since, for each combination, every time we find an unblocked path we either add a new vertex to the potential d-separator or, if no such vertex is available, we move onto the next combination as the current combination cannot be consistent with any d-separator. A more careful analysis shows that the algorithm can be implemented in linear time.

It remains to prove correctness of the algorithm. To do this, we claim that if a d-separator exists then, for at least one combination, the algorithm will find a feasible solution. To prove this we need to show that if there is a d-separator Z then there is a d-separator that is consistent with some combination. Suppose not. Then there must be a terminal pair $\{x_i, y_i\}$ and terminals b_1, b_2 that induce two extended buffer paths $P = (P_1, B, P_2)$ and $P' = (P'_1, B', P'_2)$ where the left segment P_1 is chosen for P and the right segment P'_2 is chosen for P'. Thus, no non-terminal vertex on P_2 or on P'_1 is selected in Z. Suppose P'_1 and P_2 intersect. If s is their first point of intersection then $Q = P'_1[x_i, s] \cup P'_1[s, y_i]$ is a hill path with summit s; here $P[a, b]$ denotes the subpath of P between the vertices a and b. Furthermore $Z \cap Q = \emptyset$, contradicting the fact that Z is a d-separator. Thus, we may assume that P'_1 and P_2 are vertex disjoint. Note that, by definition, B is disjoint from P_2. If B is also disjoint from P'_1 then (P'_1, B, P_2) is unblocked by Z, a contradiction. So, assume that P'_1 first intersects B at y. But then the path $Q = P'_1[x_i, t] \cup B[t, b_2] \cup P_2$ is unblocked by Z. □

Acknowledgements

We thank Guyslain Naves for interesting discussions on this work.

References

[1] K. Appel and W. Haken. Every planar map is four colorable Part I: discharging. *Illinois Journal of Mathematics*, 21:429–490, 1977. doi:10.1215/ijm/1256049011.

[2] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable Part II: reducibility. *Illinois Journal of Mathematics*, 21:491–567, 1977. doi:10.1215/ijm/1256049012.

[3] J. Berkson. Limitations of the application of fourfold table analysis to hospital data. *Biometrics Bulletin*, 2:47–53, 1946. doi:10.1093/ije/dyu022.

[4] V. Chvatal. Star-cutsets and perfect graphs. *Journal of Combinatorial Theory, Series B*, 39(3):89–199, 1985. doi:10.1016/0095-8956(85)90049-8.

[5] G. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25:71–76, 1961. doi:10.1007/BF02992776.

[6] J. Fukuyama. NP-completeness of the planar separator problems. *Journal of Graph Algorithms and Applications*, 10(2):317–328, 2006. doi:10.7155/jgaa.00130.

[7] N. Garg, V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *Algorithmica*, 18(3):3–20, 1996. doi:10.1145/167088.167266.

[8] S. Greenland, J. Pearl, and J. Robins. Causal diagrams for epidemiological research. *Epidemiology*, 10:37–48, 1999.

[9] M. Grotschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, 2nd edition, 1993. doi:10.1007/978-3-642-78240-4.

[10] K. Hoover. Causality in economics and econometrics. In S. Durlauf and L. Blume, editors, *The New Palgrave Dictionary of Economics.* Palgrave Macmillan, 2009. doi:10.2139/ssrn.930739.

[11] N. Jewell. *Statistics for Epidemiology.* Chapman and Hall/CRC, 2004.

[12] S. Lauritzen, A. Dawid, B. Larsen, and H. Leimer. Independence properties of directed markov fields. *Networks*, 20:491–505, 1990. doi:10.1002/net.3230200503.

[13] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50(2):157–224, 1988.

[14] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11:329–343, 1982. doi:10.1137/0211025.

[15] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. `doi:10.1137/0136016`.

[16] D. Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006. `doi:10.1016/j.tcs.2005.10.007`.

[17] K. Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.

[18] N. Narayanaswamy and N. Sadagopan. Connected (s,t)-vertex separator parameterized by chordality. *Journal of Graph Algorithms and Applications*, 19(1):549–565, 2015. `doi:10.7155/jgaa.00377`.

[19] O. Oellerman. The connected cutset connectivity of a graph. *Discrete Mathematics*, 69(3):301–308, 1988. `doi:10.1016/0012-365X(88)90058-1`.

[20] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Cambridge University Press, 1988. `doi:10.1016/C2009-0-27609-4`.

[21] J. Pearl. Causal inference in statistics: an overview. *Statistics Surveys*, 3:96–146, 2009. `doi:10.1214/09-SS057`.

[22] J. Pearl. *Causality: Models, Reasoning, and Inference.* Morgan and Kaufman, 2nd edition, 2009.

[23] J. Pearl and P. Meshkat. Testing regression models with fewer regressors. In D. Heckerman and J. Whittaker, editors, *Artificial Intelligence and Statistics*, pages 255–259. 1999.

[24] B. Shipley. *Cause and Correlation in Biology.* Cambridge University Press, 2002. `doi:10.1017/CBO9781139979573`.

[25] P. Sprites, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* MIT Press, 2nd edition, 2000.

[26] J. Tian, A. Paz, and J. Pearl. Finding minimal *d*-separators. Technical report, Cognitive Systems Laboratory, Los Angeles, 1988.

[27] A. Tucker. Coloring graphs with stable cutsets. *Journal of Combinatorial Theory, Series B*, 34:258–267, 1983. `doi:10.1016/0095-8956(83)90039-4`.

[28] V. Vazirani. *Approximation Algorithms.* Springer, 2001. `doi:10.1007/978-3-662-04565-7`.

[29] H. Whitney. A theorem on graphs. *Annals of Mathematics*, 32:378–390, 1931. `doi:10.2307/1968197`.

[30] D. Williamson and D. Schmoys. *The Design of Approximation Algorithms.* Cambridge, 2011.