Learning Acyclic Directed Mixed Graphs from Observations and Interventions

Jose M. Peña JOSE.M.PENA@LIU.SE
Department of Computer and Information Science
Linköping University (Sweden)

Abstract

We introduce a new family of mixed graphical models that consists of graphs with possibly directed, undirected and bidirected edges but without directed cycles. Moreover, there can be up to three edges between any pair of nodes. The new family includes Richardson's acyclic directed mixed graphs, as well as Andersson-Madigan-Perlman chain graphs. These features imply that no family of mixed graphical models that we know of subsumes the new models. We also provide a causal interpretation of the new models as systems of structural equations with correlated errors. Finally, we describe an exact algorithm for learning the new models from observational and interventional data via answer set programming.

Keywords: Acyclic directed mixed graphs; causal models; answer set programming.

1. Introduction

Undirected graphs (UGs), bidirected graphs (BGs), and directed and acyclic graphs (DAGs) have extensively been studied as representations of independence models. DAGs have also been studied as representation of causal models, because they can model asymmetric relationships between random variables. DAGs and UGs (respectively BDs) have been extended into chain graphs (CGs), which are graphs with possibly directed and undirected (respectively bidirected) edges but without semidirected cycles. Therefore, CGs can model both symmetric and asymmetric relationships between random variables. CGs with possibly directed and undirected edges may represent a different independence model depending on whether the Lauritzen-Wermuth-Frydenberg (LWF) or the Andersson-Madigan-Perlman (AMP) interpretation is considered (Lauritzen, 1996; Andersson et al., 2001). CGs with possibly directed and bidirected edges have a unique interpretation, the so-called multivariate regression (MVR) interpretation (Cox and Wermuth, 1996). MVR CGs have been extended by (i) relaxing the semidirected acyclity constraint so that only directed cycles are forbidden, and (ii) allowing up to two edges between any pair of nodes. The resulting models are called original acyclic directed mixed graphs (ADMGs) (Richardson, 2003). These are the models in which Pearl's *do*-calculus operates to determine if the causal effect of an intervention is identifiable from observed quantities (Pearl, 2009). AMP CGs have also been extended similarly (Peña, 2016). The resulting models are called alternative ADMGs.

In this paper, we combine the original and alternative ADMGs into what we simply call ADMGs. These are graphs with possibly directed, undirected and bidirected edges but without directed cycles. Moreover, there can be up to three edges between any pair of nodes. This work complements the existing works for the following reasons. To our knowledge, the only mixed graphical models in the literature that subsume AMP CGs are the already mentioned alternative ADMGs, and the so-called marginal AMP CGs (Peña, 2014). However, marginal AMP CGs are simple graphs with

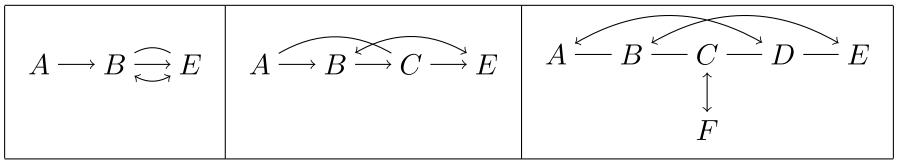

Figure 1: Examples of ADMGs.

possibly directed, undirected and bidirected edges but without semidirected cycles and, moreover, some constellations of edges are forbidden. Therefore, marginal AMP CGs do not subsume ADMGs. Likewise, no other family of mixed graphical models that we know of (e.g. original ADMGs, summary graphs (Cox and Wermuth, 1996), ancestral graphs (Richardson and Spirtes, 2002), MC graphs (Koster, 2002) or loopless mixed graphs (Sadeghi and Lauritzen, 2014)) subsume AMP CGs and hence ADMGs. To see it, we refer the reader to the works by Richardson and Spirtes (2002, p. 1025) and Sadeghi and Lauritzen (2014, Section 4.1).

The rest of the paper is organized as follows. Section 2 introduces two equivalent separation criteria for ADMGs, which define their semantics as a formalism to represent independence models. Section 3 provides an intuitive causal interpretation of ADMGs as systems of structural equations with correlated errors. Section 4 describes an exact algorithm for learning ADMGs from observational and interventional data via answer set programming (Gelfond and Lifschitz, 1988; Niemelä, 1999; Simons et al., 2002). We close the paper with some discussion in Section 5.

2. Separation Criteria

In this section, we introduce some concepts about graphical models. Unless otherwise stated, all the graphs and probability distributions in this paper are defined over a finite set V. The elements of V are not distinguished from singletons. An ADMG G is a graph with possibly directed, undirected and bidirected edges but without directed cycles, i.e. $A \rightarrow \ldots \rightarrow A$ cannot exist in G. There may be up to three edges between any pair of nodes, but the edges must be different. Edges between a node and itself are not allowed. See Figure 1 for examples of ADMGs.

Given an ADMG G, we represent with $A \circ\!\!\rightarrow B$ that $A \rightarrow B$ or $A \leftrightarrow B$ (or both) is in G. The parents of $X \subseteq V$ in G are $Pa_G(X) = \{A | A \rightarrow B \text{ is in } G \text{ with } B \in X\}$. The spouses of $X \subseteq V$ in G are $Sp_G(X) = \{A | A \leftrightarrow B \text{ is in } G \text{ with } B \in X\}$. The ancestors of $X \subseteq V$ in G are $An_G(X) = \{A | A \rightarrow \ldots \rightarrow B \text{ is in } G \text{ with } B \in X \text{ or } A \in X\}$. A route between a node V_1 and a node V_n on G is a sequence of (not necessarily distinct) nodes $V_1, \ldots, V_n$ such that V_i and V_{i+1} are adjacent in G for all $1 \leq i < n$. We do not distinguish between the sequences $V_1, \ldots, V_n$ and $V_n, \ldots, V_1$, i.e. they represent the same route. If the nodes in the route are all distinct, then the route is called a path. The subgraph of G induced by $X \subseteq V$, denoted as G_X, is the graph over X that has all and only the edges in G whose both ends are in X. Given an UG H, the marginal graph of H over $X \subseteq V$, denoted as H^X, is the UG over X such that $A - B$ is in H^X if and only if $A - B$ is in H or $A - V_1 - \ldots - V_n - B$ is H with $V_1, \ldots, V_n \notin X$.

A node C on a path in an ADMG G is said to be a collider on the path if $A \circ\!\!\rightarrow C \leftarrow\!\!\circ B$ or $A \circ\!\!\rightarrow C - B$ is a subpath. Moreover, the path is said to be connecting given $Z \subseteq V$ when

- every collider on the path is in $An_G(Z)$, and

- every non-collider C on the path is outside Z unless $A - C - B$ is a subpath and $Pa_G(C) \setminus Z \neq \emptyset$ or $Sp_G(C) \neq \emptyset$.

Let X, Y and Z denote three disjoint subsets of V. When there is no path in G connecting a node in X and a node in Y given Z, we say that X is separated from Y given Z in G and denote it as $X \perp_G Y | Z$. Note that this separation criterion generalizes the existing separation criteria for UGs, BDs, DAGs, AMP and MVR CGs, and original and alternative ADMGs. In other words, we can use the criterion above on all these families of graphical models.

Unlike in UGs, BDs, DAGs, and AMP and MVR CGs, two non-adjacent nodes in an ADMG are not necessarily separated. For example, $A \perp_G E | Z$ does not hold for any Z in the ADMGs in Figure 1. This drawback is shared by the original ADMGs (Evans and Richardson, 2013, p. 752), summary graphs and MC graphs (Richardson and Spirtes, 2002, p. 1023), and ancestral graphs (Richardson and Spirtes, 2002, Section 3.7). For ancestral graphs, the problem can be solved by adding edges to the graph without altering the separations represented until every missing edge corresponds to a separation (Richardson and Spirtes, 2002, Section 5.1). A similar solution does not exist for ADMGs (we omit the details).

Finally, we present an alternative to the separation criterion introduced above. The alternative is easier to work with in some cases. The theorem below proves that both criteria are equivalent. Specifically, a node C on a route in an ADMG G is said to be a collider on the route if $A \multimap\!\!\!\rightarrow C \leftarrow\!\!\!\circ B$ or $A \multimap\!\!\!\rightarrow C - B$ is a subroute. Note that maybe $A = B$. Moreover, the route is said to be connecting given $Z \subseteq V$ when

- every collider on the route is in Z, and
- every non-collider C on the route is outside Z unless $A - C - B$ is a subroute and $Sp_G(C) \neq \emptyset$.

Let X, Y and Z denote three disjoint subsets of V. When there is no route in G connecting a node in X and a node in Y given Z, we say that X is separated from Y given Z in G and denote it as $X \perp_G Y | Z$.

Theorem 1 *Given $\alpha, \beta \in V$ and $Z \subseteq V \setminus (\alpha \cup \beta)$, there is a path in an ADMG G connecting α and β given Z if and only if there is a route in G connecting α and β given Z.*

Proof The only if part is trivial. To prove the if part, first replace every edge $A \leftrightarrow B$ in G with the subgraph $A \leftarrow \lambda_{AB} \rightarrow B$, where λ_{AB} is a newly created node. The result is an alternative ADMG G' over $V \cup \lambda$, where λ denotes all the newly created nodes. Then, note that the route ϱ in G connecting α and β given Z can be transformed into a route ϱ' in G' connecting α and β given Z by simply replacing every edge $A \leftrightarrow B$ in ϱ with the subgraph $A \leftarrow \lambda_{AB} \rightarrow B$. To see that ϱ' is connecting, it may be worth noting that if $A - C - B$ is a subroute of ϱ with $C \in Z$ and $Pa_G(C) \setminus Z = \emptyset$, then $Sp_G(C) \neq \emptyset$ for ϱ to be connecting and, thus, $Pa_{G'}(C) \setminus Z \neq \emptyset$ since $\lambda_{CD} \in Pa_{G'}(C)$ for any $D \in Sp_G(C)$, and $\lambda_{CD} \notin Z$ since $Z \subseteq V$. Finally, note that ϱ' can be transformed into a path ρ' in G' connecting α and β given Z (Peña, 2016, Theorem 2), which can be transformed into a path ρ in G connecting α and β given Z by simply replacing every subpath $A \leftarrow \lambda_{AB} \rightarrow B$ of ρ' with the edge $A \leftrightarrow B$. To see that ρ is connecting, it may be worth noting that if $A - C - B$ is a subpath of ρ' with $C \in Z$ and $Pa_{G'}(C) \setminus Z \neq \emptyset$, then $A - C - B$ is a subpath of ρ with $Pa_G(C) \setminus Z \neq \emptyset$ or $Sp_G(C) \neq \emptyset$. ■

```
1  Set $G' = G$
2  For each edge $A \leftrightarrow B$ in $G$
3     Add the node $\lambda_{AB}$ to $G'$
4     Replace $A \leftrightarrow B$ in $G'$ with the subgraph $A \leftarrow \lambda_{AB} \rightarrow B$
5  For each node $A$ in $G$
6     Add the node $\epsilon_A$ and the edge $\epsilon_A \rightarrow A$ to $G'$
7  For each edge $A - B$ in $G$
8     Replace $A - B$ in $G'$ with the edge $\epsilon_A - \epsilon_B$
```

Figure 2: Magnification of an ADMG.

G	G'
$$\begin{tikzcd} A \arrow[r] \arrow[d] \arrow[dr] & B \arrow[d] \\ C \arrow[r, dash] \arrow[dd, dash] & D \arrow[dd, dash] \\ & \\ E \arrow[r, bend left] & F \arrow[l, bend left] \end{tikzcd}$$	$$\begin{tikzcd} \epsilon_A \arrow[r] & A \arrow[r] \arrow[d] \arrow[dr] & B \arrow[d] & \epsilon_B \arrow[l] \\ \epsilon_C \arrow[r] \arrow[dd, dash] \arrow[rrr, dash, bend right] & C & D & \epsilon_D \arrow[l] \arrow[dd, dash] \\ & & & \\ \epsilon_E \arrow[r] \arrow[rrr, dash, bend left] & E \arrow[r] & F & \epsilon_F \arrow[l] \\ & \lambda_{EF} \arrow[u] \arrow[ur] & & \end{tikzcd}$$

Figure 3: Example of the magnification of an ADMG.

3. Causal Interpretation

Let us assume that V is normally distributed. In this section, we show that an ADMG G can be interpreted as a system of structural equations with correlated errors. Specifically, the system includes an equation for each $A \in V$, which is of the form

$$A = \sum_{B \in Pa_G(A)} \alpha_{AB} B + \sum_{B \in Sp_G(A)} \beta_{AB} \lambda_{AB} + \epsilon_A \tag{1}$$

where α_{AB} and β_{AB} denote linear coefficients, and λ_{AB} and ϵ_A denote unobserved terms due to latent causes and errors, respectively. These terms are represented implicitly in G. They can be represented explicitly by magnifying G into the ADMG G' as shown in Figure 2. The magnification basically consists in adding nodes for the unobserved terms λ_{AB} and ϵ_A to G and, then, connect them appropriately. Figure 3 shows an example. Note that Equation (1) implies that every node $A \in V$ is determined by $Pa_{G'}(A)$. Likewise, ϵ_A is determined by $A \cup Pa_{G'}(A) \setminus \epsilon_A$, and λ_{AB} is determined by $A \cup Pa_{G'}(A) \setminus \lambda_{AB}$. Let ϵ denote all the error nodes ϵ_A in G', and let λ denote all the latent causes λ_{AB} in G'. Formally, we say that $A \in V \cup \lambda \cup \epsilon$ is determined by $Z \subseteq V \cup \lambda \cup \epsilon$ when $A \in Z$ or A is a function of Z. We use $Dt(Z)$ to denote all the nodes that are determined by Z. From the point of view of the separations, that a node outside the conditioning set of a separation is determined by the conditioning set has the same effect as if the node were actually in the conditioning set. Bearing this in mind, it is not difficult to see that, as desired, G and G' represent the same separations over V. The following theorem formalizes this result.

Theorem 2 *Let G denote an ADMG. Then, $X \perp_G Y|Z$ if and only if $X \perp_{G'} Y|Z$ for all X, Y and Z disjoint subsets of V.*

Proof Let G'_4 denote the graph G' in Figure 2 immediately after line 4. Note that G'_4 is an alternative ADMG over $V \cup \lambda$. We know that $X \perp_{G'_4} Y|Z$ if and only if $X \perp_{G'} Y|Z$ (Peña, 2016, Theorem 9). Therefore, it suffices to show that every path in G connecting α and β given Z can be transformed into a path in G'_4 connecting α and β given Z and vice versa, with $\alpha, \beta \in V$ and $Z \subseteq V \smallsetminus (\alpha \cup \beta)$. This can be proven in much the same way as Theorem 1. Specifically, a path ρ in G connecting α and β given Z can be transformed into a path ρ' in G'_4 connecting α and β given Z by simply replacing every edge $A \leftrightarrow B$ in ρ with the subgraph $A \leftarrow \lambda_{AB} \rightarrow B$. Finally, a path ρ' in G'_4 connecting α and β given Z can be transformed into a path ρ in G connecting α and β given Z by simply reversing the previous transformation. ■

Let $\lambda \sim \mathcal{N}(0, \Lambda)$ such that Λ is diagonal, and $\epsilon \sim \mathcal{N}(0, \Sigma)$ such that $(\Sigma^{-1})_{\epsilon_A, \epsilon_B} = 0$ if $\epsilon_A - \epsilon_B$ is not in G'. Then, G can be interpreted as a system of structural equations of the form of Equation (1) whose errors are correlated as follows

$$covariance(\epsilon_A, \epsilon_B) = \Sigma_{\epsilon_A, \epsilon_B} \tag{2}$$

for all $A, B \in V$. The next two theorems confirm that this causal interpretation of ADMGs works as intended. Let X, Y and Z denote three disjoint subsets of V. Hereinafter, we represent by $X \perp_p Y|Z$ that X and Y are conditionally independent given Z in a probability distribution p.

Theorem 3 *Let G and p denote an ADMG and a probability distribution over V. If p is specified by Equations (1) and (2), then it is Gaussian.*

Proof For each edge $A \leftrightarrow B$ in G, add the node λ_{AB} to G. Then, replace every edge $A \leftrightarrow B$ in G with the subgraph $A \leftarrow \lambda_{AB} \rightarrow B$. Note that G is now an alternative ADMG over $V \cup \lambda$. Moreover, recall that $\lambda \sim \mathcal{N}(0, \Lambda)$. Then, add the equation

$$\lambda_{AB} = \epsilon'_{AB} \tag{3}$$

and let $\epsilon' \sim \mathcal{N}(0, \Lambda)$, where ϵ' denotes all the newly created error terms ϵ'_{AB}. Then, every probability distribution $p(V \cup \lambda)$ specified by Equations (1 - 3) is Gaussian (Peña, 2016, Theorem 10), which implies the desired result. ■

It is worth mentioning that the opposite of the theorem above is not true. This negative result is inherited from the original ADMGs, for which there are Gaussian probability distributions over V that cannot be specified by Equations (1) and (2) (Richardson and Spirtes, 2002, p. 1019).

Theorem 4 *Let G and p denote an ADMG and a probability distribution over V. If p is specified by Equations (1) and (2), then $X \perp_G Y|Z$ implies that $X \perp_p Y|Z$ for all X, Y and Z disjoint subsets of V.*

Proof Transform G into an alternative ADMG over $V \cup \lambda$ as shown in the proof of Theorem 3. Then, $X \perp_G Y|Z$ implies that $X \perp_{p(V \cup \lambda)} Y|Z$ (Peña, 2016, Theorem 11), which implies the desired

```
1  Delete from $G$ all the edges $A \multimap B$ with $B \in X$
2  For each path $A - V_1 - \ldots - V_n - B$ in $G$ with $A, B \notin X$ and $V_1, \ldots, V_n \in X$
3     Add the edge $A - B$ to $G$
4  Delete from $G$ all the edges $A - B$ with $B \in X$
```

Figure 4: Intervention on an ADMG.

result. ■

A more intuitive account of the causal interpretation of ADMGs introduced above is as follows. We interpret the edge $A \rightarrow B$ as A being a cause of B. We interpret the edge $A \leftrightarrow B$ as A and B having an unobserved common cause λ_{AB}, i.e. a confounder. The unobserved causes of the node A that are not shared with any other node are grouped into an error term ϵ_A. We interpret the edge $A - B$ as ϵ_A and ϵ_B being conditionally dependent given the rest of the error terms. The dependence must be due to a non-causal relationships because, otherwise, it should have been represented by the edge $A \leftrightarrow B$. Examples of such relationships are selection bias or tying laws. We say that selection bias is present when ϵ_A and ϵ_B have a common effect that is omitted from the study but influences the selection of the samples in the study (Pearl, 2009, p. 163). We say that ϵ_A and ϵ_B are tied by a law when $f(\epsilon_A, \epsilon_B) = constant$ and f is devoid of causal meaning, much like Boyle's law relates the pressure and volume of a gas as $pressure \cdot volume = constant$ if the temperature and amount of gas remain unchanged within a closed system (Dawid, 2010, p. 77). This causal interpretation of ADMGs generalizes that of the original and alternative ADMGs (Pearl, 2009; Peña, 2016). Note however that the noise in the original ADMGs is not necessarily additive normal.

Given the above causal interpretation of an ADMG G, intervening on $X \subseteq V$ so that X is no longer under the influence of its usual causes amounts to replacing the right-hand side of the equations for the random variables in X with expressions that do not involve their usual causes.[1] Graphically, it amounts to modifying G as shown in Figure 4. Line 1 is shared with an intervention on an original ADMG. Lines 2-4 are best understood in terms of the magnified ADMG G': They correspond to marginalizing the error nodes associated with the nodes in X out of G'_ϵ, the UG that represents the correlation structure of the error nodes. In other words, lines 2-4 replace G'_ϵ with $(G'_\epsilon)^{\epsilon \setminus \epsilon_X}$, the marginal graph of G'_ϵ over $\epsilon \setminus \epsilon_X$. This makes sense since ϵ_X is no longer associated with X due to the intervention and, thus, we may want to marginalize it out because it is unobserved. This is exactly what lines 2-4 imply. Note that the ADMG after the intervention and the magnified ADMG after the intervention represent the same separations over V, by Theorem 2. This treatment of interventions on ADMGs generalizes the treatment for the original and alternative ADMGs (Pearl, 2009; Peña, 2016).

We can also extend the separation criteria for ADMGs to account for interventions. Specifically, let $X \perp_G Y | Z, do(W)$ denote that X is separated from Y given Z in an ADMG G after having intervened on W, where X, Y and Z are disjoint subsets of V, and W is a subset of V not necessarily disjoint from X, Y and Z. Likewise, let $X \perp_p Y | Z, do(W)$ represent that X and Y are conditionally independent given Z in a probability distribution p after having intervened on

1. Note that we follow the definition of intervention given by Pearl (2009, pp. 69-70), which is more general than simply setting X to a fixed value x. Note also that X is a random variable after having intervened on it.

W. The corollary below follows from Theorem 4, and provides further evidence that the causal interpretation of ADMGs introduced above works as intended.

Corollary 5 *Let G and p denote an ADMG and a probability distribution over V. If p is specified by Equations (1) and (2). Then, $X \perp_G Y | Z, do(W)$ implies that $X \perp_p Y | Z, do(W)$ for all X, Y and Z disjoint subsets of V, and all W subset of V not necessarily disjoint from X, Y and Z.*

Recall from Section 2 that two non-adjacent nodes in an ADMG G are not necessarily separated. This is not true when interventions are considered, because $A \perp_G B | do(V)$ for all non-adjacent nodes A and B of G. Therefore, some missing edges in G convey information about the observational regime, and some others about the interventional regime.

Finally, note that Equations (1) and (2) specify each node as a linear function of its parents with additive normal noise. The equations can be generalized to nonlinear or nonparametric functions as long as the noise remains additive normal. That is, for any $A \in V$

$$A = f(Pa_{G'}(A) \setminus \epsilon_A) + \epsilon_A$$

with $\epsilon \sim \mathcal{N}(0, \Sigma)$ such that $(\Sigma^{-1})_{\epsilon_A, \epsilon_B} = 0$ if $\epsilon_A - \epsilon_B$ is not in G'. That the noise is additive normal ensures that ϵ_A is determined by $A \cup Pa_{G'}(A) \setminus \epsilon_A$, which is needed for Theorem 2 to remain valid which, in turn, is needed for Theorem 4 and Corollary 5 to remain valid.

4. Learning Algorithm

In this section, we introduce an exact algorithm for learning ADMGs from observational and interventional data via answer set programming (ASP), which is a declarative constraint satisfaction paradigm that is well-suited for solving computationally hard combinatorial problems (Gelfond and Lifschitz, 1988; Niemelä, 1999; Simons et al., 2002). ASP represents constraints in terms of first-order logical rules. Therefore, when using ASP, the first task is to model the problem at hand in terms of rules so that the set of solutions implicitly represented by the rules corresponds to the solutions of the original problem. One or multiple solutions to the original problem can then be obtained by invoking an off-the-shelf ASP solver on the constraint declaration. Each rule in the constraint declaration is of the form `head :- body`. The head contains an atom, i.e. a fact. The body may contain several literals, i.e. negated and non-negated atoms. Intuitively, the rule is a justification to derive the head if the body is true. The body is true if its non-negated atoms can be derived, and its negated atoms cannot. A rule with only the head is an atom. A rule without the head is a hard-constraint, meaning that satisfying the body results in a contradiction. Soft-constraints are encoded as rules of the form `:~ body. [W]`, meaning that satisfying the body results in a penalty of W units. The ASP solver returns the solutions that meet the hard-constraints and minimize the total penalty due to the soft-constraints. In this work, we use the ASP solver `clingo` (Gebser et al., 2011), whose underlying algorithms are based on state-of-the-art Boolean satisfiability solving techniques (Biere et al., 2009).

Figure 5 shows the ASP encoding of our learning algorithm. The predicate `node(X)` in rule 1 represents that X is a node. The predicates `line(X,Y,I)`, `arrow(X,Y,I)` and `biarrow(X,Y,I)` represent that there is an undirected, directed and bidirected edge from the node X to the node Y after having intervened on the node I. The observational regime corresponds to $I = 0$. The rules 2-4 encode a non-deterministic guess of the edges for the observational regime,

```
% input predicates
% nodes(N): N is the number of nodes
% set(X): X is the index of a set of nodes
% dep(X,Y,C,I,W) (resp. indep(X,Y,C,I,W)): the nodes X and Y are dependent (resp.
%                                          independent) given the set of nodes C
%                                          after having intervened on the node I

% nodes
node(X) :- nodes(N), X=1..N.                                               % rule 1

% edges
{ line(X,Y,0) } :- node(X), node(Y), X != Y.                               %      2
{ arrow(X,Y,0) } :- node(X), node(Y), X != Y.
{ biarrow(X,Y,0) } :- node(X), node(Y), X != Y.                            %      4
line(X,Y,I) :- line(X,Y,0), node(I), X != I, Y != I, I > 0.                %      5
line(X,Y,I) :- line(X,I,0), line(I,Y,0), node(I), X != Y, I > 0.
arrow(X,Y,I) :- arrow(X,Y,0), node(I), Y != I, I > 0.
biarrow(X,Y,I) :- biarrow(X,Y,0), node(I), X != I, Y != I, I > 0.          %      8
line(X,Y,I) :- line(Y,X,I).                                                %      9
:- arrow(X,Y,I), arrow(Y,X,I).
biarrow(X,Y,I) :- biarrow(Y,X,I).                                          %      11

% directed acyclity
ancestor(X,Y) :- arrow(X,Y,0).                                             %      12
ancestor(X,Y) :- ancestor(X,Z), ancestor(Z,Y).
:- ancestor(X,Y), arrow(Y,X,0).                                            %      14

% set membership
inside_set(X,C) :- node(X), set(C), 2**(X-1) & C != 0.                     %      15
outside_set(X,C) :- node(X), set(C), 2**(X-1) & C = 0.                     %      16

% end_line/head/tail(X,Y,C,I) means that there is a connecting route
% from X to Y given C that ends with a line/arrowhead/arrowtail

% single edge route
end_line(X,Y,C,I) :- line(X,Y,I), outside_set(X,C).                        %      17
end_head(X,Y,C,I) :- arrow(X,Y,I), outside_set(X,C).
end_head(X,Y,C,I) :- biarrow(X,Y,I), outside_set(X,C).
end_tail(X,Y,C,I) :- arrow(Y,X,I), outside_set(X,C).

% connection through non-collider
end_line(X,Y,C,I) :- end_line(X,Z,C,I), line(Z,Y,I), outside_set(Z,C).
end_line(X,Y,C,I) :- end_line(X,Z,C,I), line(Z,Y,I), biarrow(Z,W,I).
end_line(X,Y,C,I) :- end_tail(X,Z,C,I), line(Z,Y,I), outside_set(Z,C).
end_head(X,Y,C,I) :- end_line(X,Z,C,I), arrow(Z,Y,I), outside_set(Z,C).
end_head(X,Y,C,I) :- end_head(X,Z,C,I), arrow(Z,Y,I), outside_set(Z,C).
end_head(X,Y,C,I) :- end_tail(X,Z,C,I), arrow(Z,Y,I), outside_set(Z,C).
end_head(X,Y,C,I) :- end_tail(X,Z,C,I), biarrow(Z,Y,I), outside_set(Z,C).
end_tail(X,Y,C,I) :- end_tail(X,Z,C,I), arrow(Y,Z,I), outside_set(Z,C).

% connection through collider
end_line(X,Y,C,I) :- end_head(X,Z,C,I), line(Z,Y,I), inside_set(Z,C).
end_head(X,Y,C,I) :- end_line(X,Z,C,I), biarrow(Z,Y,I), inside_set(Z,C).
end_head(X,Y,C,I) :- end_head(X,Z,C,I), biarrow(Z,Y,I), inside_set(Z,C).
end_tail(X,Y,C,I) :- end_line(X,Z,C,I), arrow(Y,Z,I), inside_set(Z,C).
end_tail(X,Y,C,I) :- end_head(X,Z,C,I), arrow(Y,Z,I), inside_set(Z,C).     %      33

% derived non-separations
con(X,Y,C,I) :- end_line(X,Y,C,I), X != Y, outside_set(Y,C).               %      34
con(X,Y,C,I) :- end_head(X,Y,C,I), X != Y, outside_set(Y,C).
con(X,Y,C,I) :- end_tail(X,Y,C,I), X != Y, outside_set(Y,C).
con(X,Y,C,I) :- con(Y,X,C,I).                                              %      37

% satisfy all dependences
:- dep(X,Y,C,I,W), not con(X,Y,C,I).                                       %      38

% maximize the number of satisfied independences
:~ indep(X,Y,C,I,W), con(X,Y,C,I). [W,X,Y,C,I]                             %      39

% minimize the number of lines/arrows
:~ line(X,Y,0), X < Y. [1,X,Y,1]                                           %      40
:~ arrow(X,Y,0). [1,X,Y,2]
:~ biarrow(X,Y,0), X < Y. [1,X,Y,3]                                        %      42

% show results
#show.
#show line(X,Y) : line(X,Y,0), X < Y.
#show arrow(X,Y) : arrow(X,Y,0).
#show biarrow(X,Y) : biarrow(X,Y,0), X < Y.
```

Figure 5: ASP encoding of the learning algorithm.

```
nodes(3).   % three nodes
set(0..7).  % all subsets of three nodes

% observations
dep(1,2,0,0,1).
dep(1,2,4,0,1).
dep(2,3,0,0,1).
dep(2,3,1,0,1).
dep(1,3,0,0,1).
dep(1,3,2,0,1).

% interventions on the node 1
dep(1,2,0,1,1).
dep(1,2,4,1,1).
dep(2,3,1,1,1).
dep(1,3,0,1,1).
dep(1,3,2,1,1).

% interventions on the node 2
indep(1,2,0,0,2,1).
indep(1,2,4,0,2,1).
dep(2,3,0,2,1).
dep(2,3,1,2,1).
indep(1,3,2,2,1).

% interventions on the node 3
dep(1,2,4,0,3,1).
indep(2,3,0,3,1).
indep(2,3,1,3,1).
indep(1,3,0,3,1).
indep(1,3,2,3,1).
```

Figure 6: ASP encoding of the (in)dependences in the domain.

which means that the ASP solver will implicitly consider all possible graphs during the search, hence the exactness of the search. The edges under the observational regime are used in the rules 5-8 to define the edges in the graph after having intervened on I, following the description in Section 3. Therefore, the algorithm assumes continuous random variables and additive normal noise when the input contains interventions. The random variables do not need to be normally distributed though, as discussed at the end of Section 3. The algorithm makes no such assumption when the input consists of just observations. The rules 9-11 enforce the fact that bidirected and undirected edges are symmetric and that there is at most one directed edge between two nodes. The predicate `ancestor(X,Y)` represents that the node X is an ancestor of the node Y. The rules 12-14 enforce that the graph has no directed cycles. The predicates in the rules 15-16 represent whether a node X is or is not in a set of nodes C. The rules 17-33 encode the alternative separation criterion introduced in Section 2. The predicate `con(X,Y,C,I)` in rules 34-37 represents that there is a connecting route between the node X and the node Y given the set of nodes C after having intervened on the node I. The rule 38 enforces that each dependence in the input must correspond to a connecting route. The rule 39 represents that each independence in the input that is not represented implies a penalty of W units. The rules 40-42 represent a penalty of 1 unit per edge. Other penalty rules can be added similarly.

Figure 6 illustrates with an example how to encode the (in)dependences in the probability distribution at hand, e.g. as determined from some available data. Specifically, the predicate `nodes(3)` represents that there are three nodes in the domain at hand, and the predicate `set(0..7)` represents that there are eight sets of nodes, indexed from 0 (empty set) to 7 (full set). The predicate `indep(X,Y,C,I,W)` (respectively `dep(X,Y,C,I,W)`) represents that the nodes X and Y are conditionally independent (respectively dependent) given the node set index C after having intervened on the node I. Observations correspond to $I = 0$. The penalty for failing to represent an (in)dependence is W. The penalty for failing to represent a dependence is actually superfluous in our algorithm, since rule 38 in Figure 5 enforces that all the dependences in the input are represented

$\begin{tikzcd} 1 \arrow[r] & 2 \arrow[r] \arrow[r, bend left, no head] & 3 \end{tikzcd}$	$\begin{tikzcd} 1 \arrow[r] & 2 \arrow[r] \arrow[r, bend right, leftrightarrow] & 3 \end{tikzcd}$	$\begin{tikzcd} 1 \arrow[r] & 2 \arrow[r] \arrow[r, bend left, no head] \arrow[r, bend right, leftrightarrow] & 3 \end{tikzcd}$

Figure 7: ADMGs that represent the (in)dependences in the domain.

by the graph in the output. Note also that it suffices to specify all the (in)dependences between pair of nodes, because these identify uniquely the rest of the independences in the probability distribution (Studený, 2005, Lemma 2.2). Note also that we do not assume that the probability distribution at hand is faithful to some ADMG or that it satisfies the composition property, as it is the case in most heuristic learning algorithms.

By calling the ASP solver with the encodings of the learning algorithm and the (in)dependences in the domain, the solver will essentially perform an exhaustive search over the space of graphs, and will output the graphs with the smallest penalty. Specifically, when only the observations are used (i.e. the last 15 lines of Figure 6 are removed), the learning algorithm finds 104 optimal models, including one UG, one BG, six DAGs, 13 AMP CGs, 13 MVR CGs, 37 original ADMGs, and 37 alternative ADMGs. When all the observations and interventions available are used, the learning algorithm finds two optimal models. These are the models on the left and center of Figure 7. This is the expected result given the last 15 lines in Figure 6. The rightmost model in Figure 7 is not in the output because, although it is indistinguishable from the other two given the observations and interventions in the input, it has more edges and thus receives a larger penalty, which makes it suboptimal.

Finally, the ASP code in Figure 5 can easily be modified to learn some subfamilies of ADMGs such as

- original ADMGs by adding `:- line(X,Y,0).`
- alternative ADMGs by adding `:- biarrow(X,Y,0).`
- AMP CGs by adding `:- biarrow(X,Y,0).`, `:- line(X,Y,0), arrow(X,Y,0).` and `ancestor(X,Y) :- line(X,Y,0).`
- MVR CGs by adding `:- line(X,Y,0).`, `:- biarrow(X,Y,0), arrow(X,Y,0).` and `ancestor(X,Y) :- biarrow(X,Y,0).`
- DAGs by adding `:- line(X,Y,0).` and `:- biarrow(X,Y,0).`
- UGs by adding `:- arrow(X,Y,0).` and `:- biarrow(X,Y,0).`
- BGs by adding `:- arrow(X,Y,0).` and `:- line(X,Y,0).`

5. Discussion

We plan to investigate the following two questions in an extended version of this paper. First, we have defined here the global Markov property for ADMGs by introducing two equivalent separation criteria. We plan to define local and pairwise Markov properties for ADMGs, and study Markov equivalence between ADMGs. Second, Peña (2016, Section 5.1) shows that the original and alternative ADMGs allow for complementary causal reasoning. Specifically, an example is given where the original ADMGs allow for the identification (i.e. computation from observed quantities) of the causal effect of an intervention (e.g. $p(Y|do(X))$) whereas the alternative ADMGs do not, and vice versa. By taking the union of these two examples, we can conclude that ADMGs allow for causal reasoning beyond that allowed by the union of the original and alternative ADMGs. We plan to study this question thoroughly.

References

S. A. Andersson, D. Madigan, and M. D. Perlman. Alternative Markov Properties for Chain Graphs. *Scandinavian Journal of Statistics*, 28:33–85, 2001.

A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability*. IOS Press, 2009.

D. R. Cox and N. Wermuth. *Multivariate Dependencies - Models, Analysis and Interpretation*. Chapman & Hall, 1996.

A. P. Dawid. Beware of the DAG ! *Journal of Machine Learning Research Workshop and Conference Proceedings*, 6:59–86, 2010.

R. J. Evans and T. S. Richardson. Marginal log-linear Parameters for Graphical Markov Models. *Journal of the Royal Statistical Society B*, 75:743–768, 2013.

M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam Answer Set Solving Collection. *AI Communications*, 24:107–124, 2011.

M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proceedings of 5th Logic Programming Symposium*, pages 1070–1080, 1988.

J. T. A. Koster. Marginalizing and Conditioning in Graphical Models. *Bernoulli*, 8:817–840, 2002.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

I. Niemelä. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. *Annals of Mathematics and Artificial Intelligence*, 25:241–273, 1999.

J. M. Peña. Marginal AMP Chain Graphs. *International Journal of Approximate Reasoning*, 55: 1185–1206, 2014.

J. M. Peña. Alternative Markov and Causal Properties for Acyclic Directed Mixed Graphs. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence*, 2016.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2009.

T. Richardson. Markov Properties for Acyclic Directed Mixed Graphs. *Scandinavian Journal of Statistics*, 30:145–157, 2003.

T. Richardson and P. Spirtes. Ancestral Graph Markov Models. *The Annals of Statistics*, 30:962–1030, 2002.

K. Sadeghi and S. L. Lauritzen. Markov Properties for Mixed Graphs. *Bernoulli*, 20:676–696, 2014.

P. Simons, I. Niemelä, and T. Soininen. Extending and Implementing the Stable Model Semantics. *Artificial Intelligence*, 138:181–234, 2002.

M. Studený. *Probabilistic Conditional Independence Structures*. Springer, 2005.