On expressiveness of the chain graph interpretations

Dag Sonntag and Jose M Pena

Original Publication:

Dag Sonntag and Jose M Pena, On expressiveness of the chain graph interpretations, 2016, International Journal of Approximate Reasoning, (68), 91-107.
http://dx.doi.org/10.1016/j.ijar.2015.07.009

On Expressiveness of the Chain Graph Interpretations

Dag Sonntag, Jose M. Peña

ADIT, IDA, Linköping University, Sweden
dag.sonntag@liu.se, jose.m.pena@liu.se

Abstract

In this article we study the expressiveness of the different chain graph interpretations. Chain graphs is a class of probabilistic graphical models that can contain two types of edges, representing different types of relationships between the variables in question. Chain graphs is also a superclass of directed acyclic graphs, i.e. Bayesian networks, and can thereby represent systems more accurately than this less expressive class of models. Today there do however exist several different ways of interpreting chain graphs and what conditional independences they encode, giving rise to different so called chain graph interpretations. Previous research has approximated the number of representable independence models for the Lauritzen-Wermuth-Frydenberg and the multivariate regression chain graph interpretations using an MCMC based approach. In this article we use a similar approach to approximate the number of models representable by the latest chain graph interpretation in research, the Andersson-Madigan-Perlman interpretation. Moreover we summarize and compare the different chain graph interpretations with each other. Our results confirm previous results that directed acyclic graphs only can represent a small fraction of the models representable by chain graphs, even for a low number of nodes. The results also show that the Andersson-Madigan-Perlman and multivariate regression interpretations can represent about the same amount of models and twice the amount of models compared to the Lauritzen-Wermuth-Frydenberg interpretation. However, at the same time almost all models representable by the latter interpretation can only be represented by that interpretation while the former two have a large intersection in terms of representable models.

Keywords: Chain graphs, Lauritzen-Wermuth-Frydenberg interpretation, Andersson-Madigan-Perlman interpretation, multivariate regression interpretation, MCMC sampling, expressibility of probabilistic graphical models.

1. Introduction

Chain graphs (CGs) are probabilistic graphical models (PGMs) [8] that extend the formalism of directed acyclic graphs (DAGs) [13], i.e. Bayesian networks, with an additional, non-directed, edge. The directed edge works similarly as in DAGs when it comes to representing independences while the non-directed

edge can be interpreted in different ways, giving rise to different so called CG interpretations. This means that CGs can represent every independence model representable by any DAG while the opposite does not hold. The most researched CG interpretations are the Lauritzen-Wermuth-Frydenberg (LWF) interpretation [6, 9], the Andersson-Madigan-Perlman (AMP) interpretation [1] and the multivariate regression (MVR) interpretation [3, 4]. Each interpretation has its own way of determining conditional independences in a CG and it can be noted that no interpretation subsumes another in terms of representable independence models [5, 16].

The question of how much more expressive CGs are compared to DAGs, i.e. how many more independence models that can be represented by CGs compared to those representable by DAGs, has recently been studied for CGs of the LWF interpretation (LWF CGs) and the MVR interpretation (MVR CGs) [11, 17]. The results from these studies show that the ratio of the number of DAG models (independence models representable by DAGs) compared to the number of CG models (independence models representable by CGs) decreases exponentially as the number of nodes in the graphs increases.

In this article we carry out a similar study for CGs of the AMP interpretation (AMP CGs) and conclude the research of expressiveness of independence models for the three CG interpretations. This allows us to summarize the method for approximating the number of CG models as well as the results we have seen for the different CG interpretations for up to 20 nodes. We also compare the three CG interpretations with each other, as well as to DAGs, in terms of representable independence models and number of graphs per Markov equivalence class. Finally we also compare the three CG interpretations to their non-directed subclasses which are undirected graphs (UGs) [13], i.e. Markov networks, for the LWF and AMP CGs respectively bidirected graphs (BDs) [3], i.e. covariance graphs, for MVR CGs.

Measuring the number of representable independence models should in principle be easy to do. We only have to enumerate every possible CG and check whether it is the unique representative of the Markov equivalence class, i.e., CG model, for the CG interpretation we study. The problem here is however that since both the number of CGs and CG models grows superexponentially as the number of nodes in the graphs increases, it is infeasible to enumerate all of them for more than five nodes. Hence we instead use a Markov chain Monte Carlo (MCMC) approach that allows us approximate the ratio of CG models to DAG models for each interpretation. Then, since we know the number of DAG models from previous research [11, 17], we can calculate the number of CG models for the different CG interpretations.

The MCMC approach consists of creating a Markov chain whose states are all possible CG models for a given number of nodes and whose stationary distribution is the uniform distribution over these models. What CG models are possible depends on what CG interpretation we study. By transitioning through this Markov chain, CG models can then be sampled from approximately the uniform distribution over all CG models of the CG interpretation in question. For each of these CG models it is then possible to be check whether the represented

independence model also can be represented as a DAG, and hence to calculate the ratio of CG models to DAG models. The CG model samples also allows us to calculate other ratios, such as whether the independence model can be represented by some other CG interpretation. Moreover we make the CG models publicly available online and it is our intent that they can be used for further studies in the field. This can for example be when evaluating CG learning algorithms to get more accurate evaluations than what is achieved today when randomly generated CGs are used instead of randomly generated CG models [10, 12].

The rest of the article is organised as follows. In the next section we discuss the theory of the general MCMC sampling algorithm used for all the CG interpretations. This is followed by Section 3 where we define how this general algorithm is implemented for the AMP CG interpretation. The implementations for the LWF and MVR interpretations can be found in their appropriate articles [11, 17]. In Section 4 we then present the results we can calculate from using the sampling algorithm and in Section 5 we have a short conclusion. The article also has an Appendix containing the proofs for the theorems presented in Section 3 as well as a notation section necessary for these proofs. Note that the article itself contains no notation section since the terms used are common in the probabilistic graphical model community. However, if some term is unfamiliar to the reader all terms used are properly defined in the Appendix.

2. The Markov Chain Monte Carlo Approach

In this section we cover the general theory behind the MCMC sampling approach used in this article. Note that we only cover the theory of the MCMC sampling method very briefly and for a more complete introduction of the sampling method we instead refer the reader to the work by Häggström [7].

The MCMC sampling approach consists of creating a Markov chain whose unique stationary distribution is the desired distribution and then sample this Markov chain after a number of transitions. The Markov chain is defined by a set of operators that allows us to transition from one state to another. It can then be shown that if the set of operators have certain properties and the number of transitions goes to infinity then all states are visited according to the stationary distribution [7]. Moreover, in practice it has also been shown that the number of transitions can be relatively low and a good approximation of the stationary distribution can still be achieved.

In our case the possible states of the Markov chain are all possible independence models representable by CGs, i.e. CG models, of the interpretation we study and the stationary distribution is the uniform distribution. These are then represented by the appropriate unique graphical representatives, i.e. the largest chain graphs (LCG) [6] for LWF CGs, the largest deflagged graphs (LDG) [15] for AMP CGs and essential MVR CGs [17] for MVR CGs. The operators, seen in Table 1, then add and remove certain edges in these graphical representatives, allowing the MCMC sampling algorithm to transition between all possible CG models. If the resulting graph is not a unique graphical representative of the

Table 1: Operators used for MCMC sampling for the different CG interpretations. See the corresponding articles for proper definitions for the LWF [11] and MVR [17] CG interpretations and Section 3 for the AMP CG interpretation.

LWF	AMP	MVR
Add directed edge	Add directed edge	Add directed edge
Remove directed edge	Remove directed edge	Remove directed edge
Add undirected edge	Add undirected edge	Add undirected edge
Remove undirected edge	Remove undirected edge	Remove undirected edge
Add immorality	Add two directed edges	Add bidirected edge
Remove immorality	Remove two directed edges	Remove bidirected edge
No change		Add V-collider
		Remove V-collider

appropriate type, such as a LCG in the case of LWF CGs, the operator is not applied.

As noted in the description of Table 1 only the titles of the operations are presented and the reader is referred to the corresponding article or section for a proper description. We can however note that most of the operations work by adding or removing edges between any ordered set of nodes in the graph. This also holds for the "Add immorality" and "Remove immorality" operators for LWF CGs which basically adds respectively removes two directed edges from the LCG. Hence, an example of a Markov chain for the AMP CG interpretation, with two nodes x and y, would contain two states since there exists two LDGs: $x-y$ and $x \quad y$ (i.e. no edge between them). The possible operators in each state would be add respectively remove $x \rightarrow y$, $y \leftarrow x$, $x-y$ and $y-x$. However, only two such operations would be valid for any possible state (remove $x-y$ and remove $y-x$ for the state $x-y$ respectively add $x-y$ and add $y-x$ for the state $x \quad y$). All other operators would simply result in no change.

If we look at the whole sampling process the procedure follows as described in Algorithm 1. It starts with the empty graph and then performs a set number of transitions before it samples the current state, i.e. the current model. This is referred to as the *burn-in* phase and is included to remove any bias caused by the choice of initial state (the empty graph in our case). This process is then essentially repeated until the chosen number of samples have been sampled. However, to additionally ensure that the initial state does not bias the sampling the previous sample is chosen as starting point for the next sample. It can be noted that the sampling method described here corresponds to MCMC sampling using *thinning* and not the more common MCMC sampling method (not using thinning) where each state is sampled after the burn-in phase. Using the common approach is however not possible since the large set of possible states, and the long distances between some states with the chosen operators, would require an infeasible large amount of CGs to be sampled.

To ensure that the stationary distribution of the Markov chain is the uniform distribution of all possible CG models we have to prove that the operators fulfill the following properties [7]: aperiodicity, irreducibility and reversibility.

Input: A CG interpretation $\mathcal{I}$, a set of graphical operators $\mathcal{O}$ that fulfills the aperiodicity, irreducibility and reversibility properties, a number of variables n, the desired number of samples k and the number of burn-in transitions l.
Output: A set of graphs sampled from the uniform distribution over all CG models with n nodes of interpretation $\mathcal{I}$.

Let S be the empty set of graphs.
Let G be the empty graph with n nodes.
Repeat k times:
  Repeat for l iterations:
    Choose uniformly an operator o from $\mathcal{O}$.
    If $o(G)$[1] is a unique graphical representative in the CG interpretation $\mathcal{I}$:
      Let $G = o(G)$.
  Copy G into S.
Return S.

[1] $o(G)$ is the resulting graph from applying operator o onto the graph G.

Algorithm 1: General MCMC sampling algorithm

Aperiodicity, i.e. that the Markov chain does not end up in the same state periodically, and irreducibility, i.e. that any state can be reached from any other state using only the defined operators, proves that the Markov chain has a unique stationary distribution. Reversibility, i.e. that the probability of transitioning from any state A to any other state B is the same as for transitioning from B to A, then proves that this distribution also is the uniform distribution. The operators of the LWF and MVR CG interpretations are proven to fullfill these properties in their corresponding articles [11, 17] while for the operators of the AMP CG interpretation the properties are shown hold in the next section.

3. MCMC Operators for the AMP CG Interpretation

In this section we define the operators used for the AMP CG interpretation and prove that they fulfill the aperiodicity, reversibility and irreducibility properties for LDGs with at least two nodes. The reason why we chose to use LDGs as the unique graphical representation, and not the AMP essential graphs [2], is two-fold. First, and most important, that the structure of the LDGs allowed for irreducibility to be proven using only add and remove edge operators, i.e. without using an add V-collider operation which is required for the MVR CG interpretation [17]. Secondly that, although no formal proof of complexity has been given, the algorithm for testing if a graph is an LDG [15] appears to be faster than the corresponding algorithm for AMP essential graphs [2]. This even though the algorithm described for testing if a graph G is an LDG consists of first checking if G is an AMP CG, then finding the LDG H of the Markov equivalence class of G and finally checking if G and H have the same structure. To find the LDG of the Markov equivalence class of G the algorithm defined by Roverato and Studený was used [15].

The operators used to create the Markov chain for the AMP CG interpretation are defined in Definition 1 and it follows from Theorems 1, 2 and 3 that they fulfill the aperiodicity, reversibility and irreducibility properties for LDGs with at least two nodes.

Definition 1. Markov Chain Operators
Choose uniformly and perform one of the following six operators to transition from an LDG G to the next LDG H in the Markov chain.

- **Add directed edge.** Choose two nodes x, y in G uniformly and with replacement. If x is not adjacent of y in G and $G \cup \{x \rightarrow y\}$ is an LDG let $H = G \cup \{x \rightarrow y\}$, otherwise let $H = G$.
- **Remove directed edge.** Choose two nodes x, y in G uniformly and with replacement. If $x \rightarrow y$ is in G and $G \setminus \{x \rightarrow y\}$ is an LDG let $H = G \setminus \{x \rightarrow y\}$, otherwise let $H = G$.
- **Add undirected edge.** Choose two nodes x, y in G uniformly and with replacement. If x is not adjacent of y in G and $G \cup \{x - y\}$ is an LDG let $H = G \cup \{x - y\}$, otherwise let $H = G$.
- **Remove undirected edge.** Choose two nodes x, y in G uniformly and with replacement. If $x - y$ is in G and $G \setminus \{x - y\}$ is an LDG let $H = G \setminus \{x - y\}$, otherwise let $H = G$.
- **Add two directed edges.** Choose four nodes x, y, z, w in G uniformly and with replacement. If x is not adjacent of y in G, z is not adjacent of w in G and $G \cup \{x \rightarrow y, z \rightarrow w\}$ is an LDG let $H = G \cup \{x \rightarrow y, z \rightarrow w\}$, otherwise let $H = G$. Note that y might be equal to w in this operation.
- **Remove two directed edges.** Choose four nodes x, y, z, w in G uniformly and with replacement. If $x \rightarrow y$ and $z \rightarrow w$ are in G and $G \setminus \{x \rightarrow y, z \rightarrow w\}$ is an LDG let $H = G \setminus \{x \rightarrow y, z \rightarrow w\}$, otherwise let $H = G$. Note that y might be equal to w in this operation.

Theorem 1. *The operators in Definition 1 fulfill the aperiodicity property when G contains at least two nodes.*

Theorem 2. *The operators in Definition 1 fulfill the reversibility property for the uniform distribution.*

Theorem 3. *Given an LDG G any other LDG H can be reached using the operators described in Definition 1 such that all intermediate graphs are LDGs.*

Finally we also have to prove what the conditions are for when the independence model of an LDG can be represented as a DAG respectively an UG.

Theorem 4. *Given an LDG G there exists a DAG H such that G and H represent the same independence model iff G contains no flags and G contains no chordless undirected cycles.*[1]

[1] See Appendix A for definitions of flags and chordless undirected cycles.

Theorem 5. *Given an LDG G there exists an UG H such that G and H represent the same independence model iff G contains no directed edges.*

4. Results

Using the MCMC approach described in Section 2 we were able to sample CG models uniformly for each CG interpretation for up to 20 nodes. For each number of nodes and CG interpretation 10^5 models were sampled with 10^5 transitions between each sample. The implementation was carried out in C++, run on an Intel Core i5 processor and it took approximately one week to complete the sampling of all sample sets per CG interpretation. Moreover we also enumerated all CG models for up to five nodes, for each interpretation, to allow a comparison between the approximated and exact values. The sampled graphs and code are available for public access at
http://www.ida.liu.se/divisions/adit/data/graphs/CGSamplingResources.

From the sampled graphs we could then calculate the ratio of CG models whose independence models could be represented by DAGs, UGs and BGs as well as some other CG interpretation for the LWF, AMP and MVR CG interpretation. The results are shown in Tables 2, 3, 4, 5 and 6 and are discussed in Sections 4.1, 4.2 and 4.3. Before we look into these results we will however shortly discuss the validity of the approximated uniform distributions. We have in our work not been able to calculate the variance in the results due to the sampling process since this would take several months in calculation time. However, there are several indicators suggesting that the models are sampled from almost uniform distributions. First, that the calculated ratios, as well as the number of the different types of edges, have clear smooth trends as the number of nodes increases. Secondly we have also seen, although no formal variance have been calculated, that different runs of the MCMC sampling approach gives no significant difference in the results for up to 20 nodes. For more than 20 nodes we could however see that this was no longer true and the ratios could differ significantly for a certain number of nodes when the algorithm was run multiple times. Hence this indicates that the MCMC sampling approach does not sample from the uniform distribution if it is run with more than 20 nodes and the described parameters, i.e. the number of sampled models and transitions between each sample.

4.1. Ratios of CG Models Representable as Subclasses

When it comes to the results we can start with Table 2 where the ratio of the number of DAG models compared to number of CG models are shown for the different interpretations. We can note that the approximation seems to be very accurate for up to five nodes but for more than five nodes we do not have any exact values to compare the approximations to. We can however plot the ratios of the number of DAG models compared to the number of CG models in a graph with logarithmic scale, as seen in Figure 1. This allows us to see that the ratios seem to be linear in the logarithmic scale and hence follow exponential equations.

Table 2: Exact and approximate ratios of CG models whose independence models can be represented as DAGs.

NODES	EXACT			APPROXIMATE		
	LWF	AMP	MVR	LWF	AMP	MVR
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	0.9250	0.8393	0.8259	0.9327	0.8392	0.8235
5	0.7624	0.6113	0.5905	0.7646	0.6136	0.5900
6				0.5829	0.4382	0.4099
7				0.4179	0.3058	0.2868
8				0.2860	0.2067	0.1951
9				0.1924	0.1407	0.1307
10				0.1286	0.0948	0.0866
11				0.0831	0.0616	0.0565
12				0.0554	0.0403	0.0377
13				0.0349	0.0257	0.0239
14				0.0237	0.0155	0.0159
15				0.0152	0.0108	0.0098
16				0.0096	0.0066	0.0064
17				0.0062	0.0045	0.0049
18				0.0038	0.0028	0.0027
19				0.0027	0.0018	0.0019
20				0.0017	0.0010	0.0011

The equations can be found to be $R_{LWF} = 9.1 * 0.654^n$,$R_{AMP} = 7.2 * 0.645^n$ and $R_{MVR} = 6.2 * 0.653^n$, where n is the number of nodes and R the ratio of the subscripted interpretation. This means that the ratios decreases exponentially as the number of nodes increases for all three CG interpretations.

We can also compare the number of CG models for the different interpretations to the number of models of their non-directed subclasses, i.e. UGs for LWF and AMP CGs respectively BGs for MVR CGs. The exact and approximate ratios of these comparisons are shown in Table 3. Here we can note that the amount of independence models representable by these subclasses are almost non-existent in comparison to the number of CG models for models of 10 or more nodes. Moreover, since the ratios decreases so quickly, we have not been able to find any equations describing the ratios given the number of nodes.

4.2. Ratios of the Number of CGs per CG model and Approximate number of Representable Models

Another interesting ratio that can be studied is the average number of CGs per CG model seen in Table 4. This ratio can be calculated using the equation

$$\frac{\#CGs}{\#CGmodels} = \frac{\#CGs}{\#DAGs} * \frac{\#DAGs}{\#DAGmodels} * \frac{\#DAGmodels}{\#CGmodels} \tag{1}$$

where $\#CGmodels$ represents the number of CG models of a certain interpretation and so on. The ratio $\frac{\#CGs}{\#DAGs}$ can then be found using the iterative equations by Robinsson [14] and Steinsky [18] while $\frac{\#DAGs}{\#DAGmodels}$ can be found in previous studies for DAG models [11]. Finally we can also get the ratio $\frac{\#DAGmodels}{\#CGmodels}$ from

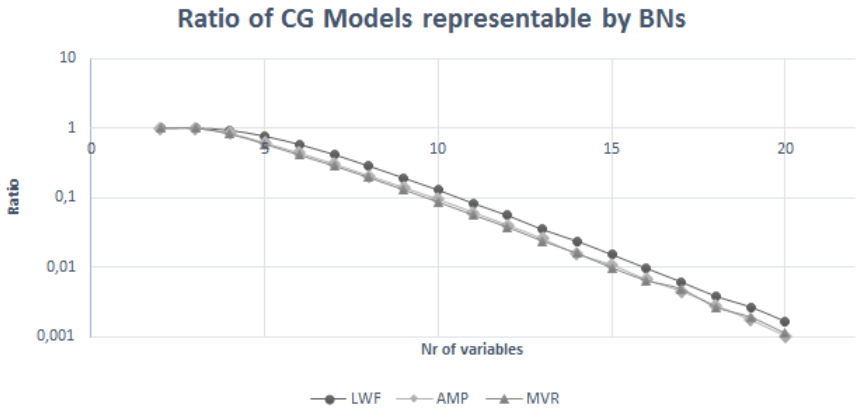


Figure 1: The ratios (displayed with a logarithmic scale) of the number of DAG models compared to the number of CG models for different number of nodes for the different CG interpretations.

Table 3: Exact and approximate ratios of LWF, AMP respectively MVR CG models whose independence models can be represented as UGs respectively BGs.

NODES	**EXACT**			**APPROXIMATE**		
	LWF	AMP	MVR	LWF	AMP	MVR
2	1	1	1	1	1	1
3	0.7273	0.7273	0.7273	0.7188	0.7275	0.7255
4	0.3200	0.2857	0.2857	0.3122	0.2839	0.2855
5	0.0889	0.0689	0.0689	0.0809	0.0632	0.0697
6				0.0165	0.0112	0.0124
7				0.0032	0.0019	0.0019
8				0.0003	0.0002	0.0003
9				0.0002	0.0000	0.0000
10				0.0000	0.0000	0.0000

Table 2. If we study the values in Table 4 we can see that the average number of CGs per CG model appear to converge to approximately 26 for the LWF CG interpretation respectively 17 for the AMP and MVR CG interpretations. This corresponds well with what we have seen for DAGs, although in that case the convergence was around 4 DAGs per DAG model [11]. The means that traversing the space of CG models when learning CG structures is considerably more efficient than traversing the space of all CGs. However, at the same time, it also means that this efficiency does not scale as the number of nodes in the graphs increases.

Finally, in Table 5, we can also see the number of CG models for the different CG interpretations. These numbers follows directly from the number of CGs per CG model, shown in Table 4, and the equations for calculating the number of CGs for a given number of nodes defined by Steinsky [18]. We can here see

Table 4: Exact and approximate numbers of CGs per CG model.

NODES	EXACT			APPROXIMATE		
	LWF	AMP	MVR	LWF	AMP	MVR
2	2	2	2	1.97	1.97	1.97
3	4.55	4.55	4.55	4.47	4.47	4.47
4	8.44	7.54	7.54	8.61	7.75	7.60
5	12.38	9.59	9.59	12.61	10.12	9.73
6				15.80	11.87	11.11
7				18.05	13.21	12.39
8				20.20	14.59	13.77
9				20.97	15.34	14.25
10				22.61	16.66	15.23
11				23.14	17.16	15.74
12				23.66	17.22	16.09
13				22.88	16.85	15.64
14				24.64	16.10	16.54
15				25.63	18.20	16.60
16				24.87	17.15	16.63
17				24.94	18.37	19.67
18				24.24	17.89	16.94
19				26.51	17.38	18.96
20				26.26	16.29	17.72

that the AMP and MVR CG interpretations can represent approximately the same amount of independence models, while the LWF CGs only can represent about 65% of this amount. Hence the AMP and MVR interpretations are the most expressive CG interpretations, in terms of the number of representable independence models, while the LWF interpretation falls behind. The ratio between them does however appear to be constant as the number of nodes in the models increases.

4.3. Intersections between the CG Interpretations in terms of Representable Independence Models

In Table 2 we saw the ratio of CG models whose independence model could be represented by DAGs. The different CG interpretations do however not only intersect in terms of representable independence models over this subclass. For example there exist UGs whose independence models are representable as both AMP CGs and LWF CGs, but not as DAGs. The graphical conditions for when the independence model of a CG of one interpretation can be represented by another interpretation have been defined and proven correct [1, 16] . Hence, using the sampled CG models we can estimate the size of the different intersections. The results of this are shown in Table 6 where the number of independence models in an intersection is compared to the number of all representable models for the relevant CG interpretations. The ratios are also illustrated in Figure 2 to allow the reader a better view of how the spaces of the intersections and representable independence models change as the number of nodes in the models increases. We can here see that almost all independence models representable by

Table 5: Exact and approximate numbers of CG models representable for the different CG interpretations.

NODES	EXACT			APPROXIMATE		
	LWF	AMP	MVR	LWF	AMP	MVR
2	2	2	2	2.03	2.03	2.03
3	11	11	11	11	11	11
4	200	224	224	196	218	222
5	11519	14869	14866	11313	14097	14662
6				1.83 E+6	2.43 E+6	2.60 E+6
7				7.57 E+8	1.03 E+9	1.10 E+9
8				7.31 E+11	1.01 E+12	1.07 E+12
9				1.71 E+15	2.34 E+15	2.52 E+15
10				8.57 E+18	1.16 E+19	1.27 E+19
11				9.95 E+22	1.34 E+23	1.46 E+23
12				2.53 E+27	3.47 E+27	3.71 E+27
13				1.47 E+32	1.99 E+32	2.15 E+32
14				1.65 E+37	2.52 E+37	2.46 E+37
15				4.11 E+42	5.79 E+42	6.35 E+42
16				2.34 E+48	3.40 E+48	3.50 E+48
17				2.75 E+54	3.73 E+54	3.48 E+54
18				7.04 E+60	9.53 E+60	1.01 E+61
19				3.38 E+67	5.16 E+67	4.73 E+67
20				3.78 E+74	6.09 E+74	5.60 E+74

LWF CGs only can be represented by this interpretation while the intersection between the AMP and MVR CGs is quite large (25% for 20 nodes).

5. Conclusion

In this article we have presented an approach for sampling AMP CG models from the approximately uniform distribution of AMP CG models for up to 20 nodes. This has allowed us to finalize the research of expressibility, in terms of the number of representable independence models, for the different CG interpretations that exist in research today. We have presented relevant ratios, such as the ratio of independence models representable by CGs that also can be represented by DAGs, for the different CG interpretations for up to 20 nodes. Moreover, we have been able compare the different CG interpretations with each other as well as calculate how large their intersections approximately are in terms of representable independence models. Finally we have also been able to determine how the relevant ratios and intersections changes as the number of nodes in the models increases, i.e. whether they follow exponential equations, linear equations or remain constant.

The results presented confirm previous results that DAGs only can represent a fraction of the independence models representable by CGs with more than 10 nodes and that the ratio decreases exponentially as the number of nodes increases. New results are however that AMP CGs and MVR CGs can represent approximately the same number of independence models while LWF CGs can represent approximately 35% less models than the other two CG interpreta-

Table 6: Exact and approximate ratios of CG models that intersect between the different CG interpretations.

NODES	Ratio of LWF representable as		Ratio of AMP representable as		Ratio of MVR representable as	
	AMP	MVR	LWF	MVR	LWF	AMP
2	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	0.9327	0.9327	0.8392	0.9300	0.8235	0.9126
5	0.7744	0.7646	0.6215	0.8303	0.5900	0.7984
6	0.5997	0.5829	0.4507	0.7519	0.4099	0.7033
7	0.4344	0.4179	0.3178	0.6845	0.2868	0.6419
8	0.2987	0.2860	0.2159	0.6346	0.1951	0.5990
9	0.2011	0.1924	0.1470	0.5914	0.1307	0.5495
10	0.1344	0.1286	0.0990	0.5432	0.0866	0.4966
11	0.0866	0.0831	0.0642	0.5027	0.0565	0.4613
12	0.0580	0.0554	0.0422	0.4688	0.0377	0.4382
13	0.0365	0.0349	0.0269	0.4352	0.0239	0.4040
14	0.0248	0.0237	0.0162	0.3982	0.0159	0.4092
15	0.0158	0.0152	0.0112	0.3718	0.0098	0.3390
16	0.0101	0.0096	0.0069	0.3453	0.0064	0.3349
17	0.0063	0.0062	0.0047	0.3164	0.0049	0.3387
18	0.0040	0.0038	0.0030	0.2928	0.0027	0.2772
19	0.0028	0.0027	0.0019	0.2759	0.0019	0.3011
20	0.0018	0.0017	0.0011	0.2507	0.0011	0.2726

tions. At the same time almost all independence models representable by LWF CGs can only be represented by this CG interpretation, while the AMP and MVR CG interpretations have a large intersection in terms of representable independence models (25% for 20 nodes). In addition to this we have also been able to approximate the average number of CGs per CG model for the different interpretations. The results show that the average number converges to approximately 17 for the AMP and MVR CG interpretations respectively 26 for LWF CG interpretation. This corresponds well with previous research for DAGs, although in that case the ratio converged to approximately 4 DAGs per DAG model [11].

Moreover, with the presented sampling method for AMP CG models sampling methods for CG models of all CG interpretations are now available [11, 17]. This opens up for MCMC based structure learning algorithms for the different CG interpretations that traverses the space of CG models instead of the space of CGs. More importantly it also opens up for further studies on what independence models and systems the different CG interpretations can represent, which is an important question where the answer is still unclear.

Appendix A

In this appendix we prove the theorems in Section 3. First we do however define the necessary notation. Note that although we also define the notation for LWF and MVR CGs this appendix is focused on the AMP CG interpretation.

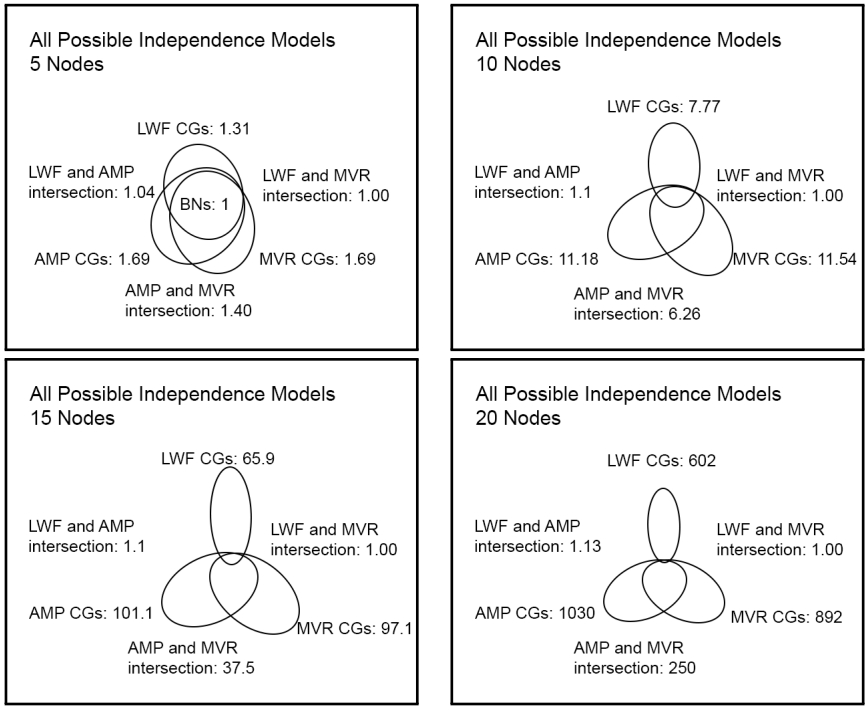


Figure 2: The intersections of representable independence models for the different CG interpretations for different number of nodes in the models. In the figures the space of independence models representable by DAGs have size 1 and all other spaces are relative to this. For example, LWF CGs can represent 31% more independence models than DAGs for 5 nodes while all independence models representable by both LWF and AMP CGs, including the DAGs, only are 4% more than those represetable by DAGs.

Hence, for proper examples of the terms used regarding LWF and MVR CGs the reader is referred to articles handling these interpretations [6, 9, 16].

Notation

All graphs are defined over a finite set of variables V. If a graph G contains an edge between two nodes v_1 and v_2, we denote with $v_1 \rightarrow v_2$ a *directed edge*, with $v_1 \leftrightarrow v_2$ a *bidirected edge* and with $v_1 - v_2$ an *undirected edge*. A set of nodes is said to be *complete* if there exist edges between all pairs of nodes in the set.

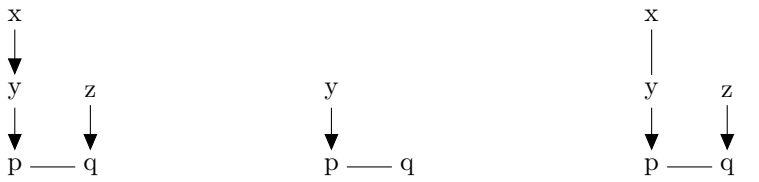
The *parents* of a set of nodes X of G is the set $pa_G(X) = \{v_1 | v_1 \rightarrow v_2$ is in G, $v_1 \notin X$ and $v_2 \in X\}$. The *children* of X is the set $ch_G(X) = \{v_1 | v_2 \rightarrow v_1$ is in G, $v_1 \notin X$ and $v_2 \in X\}$. The *spouses* of X is the set $sp_G(X) = \{v_1 | v_1 \leftrightarrow v_2$ is in G, $v_1 \notin X$ and $v_2 \in X\}$. The *neighbours* of X is the set $nb_G(X) = \{v_1 | v_1 - v_2$ is in G, $v_1 \notin X$ and $v_2 \in X\}$. The *boundary* of X is the set $bd_G(X) = pa_G(X) \cup nb_G(X) \cup sp_G(X)$. The *adjacents* of X is the set $ad_G(X) = bd_G(X) \cup ch_G(X)$.

A *route* from a node v_1 to a node v_n in G is a sequence of nodes $v_1, \ldots, v_n$

such that $v_i \in ad_G(v_{i+1})$ for all $1 \leq i < n$. A *section* of a route is a maximal (w.r.t. set inclusion) non-empty set of nodes $b_1, \ldots, b_n$ such that the route contains the subpath $b_1 - b_2 - \ldots - b_n$. It is called a *collider section* if $b_1, \ldots, b_n$ together with the two neighbouring nodes in the route, a and c, form the subpath $a \rightarrow b_1 - b_2 - \ldots - b_n \leftarrow c$. For any other configuration the section is a non-collider section. A *path* is a route containing only distinct nodes. The length of a path is the number of edges in the path. A path is *descending* if $v_i \in bd_G(v_{i+1})$ for all $1 \leq i < n$. A path is called a *cycle* if $v_n = v_1$. A cycle is called a *semi-directed cycle* if it is descending and $v_i \rightarrow v_{i+1}$ is in G for some $1 \leq i < n$. A path $\pi = v_1, \ldots, v_n$ is *minimal* if there exists no other path π_2 between v_1 and v_n such that $\pi_2 \subset \pi$ holds. The *descendants* of a set of nodes X of G is the set $de_G(X) = \{v_n |$ there is a descending path from v_1 to v_n in G, $v_1 \in X$ and $v_n \notin X\}$. A path is *strictly descending* if $v_i \in pa_G(v_{i+1})$ for all $1 \leq i < n$. The *strict descendants* of a set of nodes X of G is the set $sde_G(X) = \{v_n |$ there is a strict descending path from v_1 to v_n in G, $v_1 \in X$ and $v_n \notin X\}$. The *ancestors* (resp. *strict ancestors*) of X form the set $an_G(X) = \{v_1 | v_n \in de_G(v_1), v_1 \notin X, v_n \in X\}$ (resp. $san_G(X) = \{v_1 | v_n \in sde_G(v_1), v_1 \notin X, v_n \in X\}$).

A directed acyclic graph (DAG), i.e. a Bayesian network, is a graph containing only directed edges and no semi-directed cycles while an undirected graph (UG), i.e. a Markov network (respectively a bidirected graph (BG), i.e. a covariance graph) is a graph containing only undirected edges (respectively bidirected edges). A CG under the Lauritzen-Wermuth-Frydenberg (LWF) interpretation, denoted LWF CG, contains only directed and undirected edges but no semi-directed cycles. Likewise a CG under the Andersson-Madigan-Perlman (AMP) interpretation, denoted AMP CG, is a graph containing only directed and undirected edges but no semi-directed cycles. A CG under the multivariate regression (MVR) interpretation, denoted MVR CG, is a graph containing only directed and bidirected edges but no semi-directed cycles. A *connectivity component* C in a LWF CG or an AMP CG (respectively MVR CG) is a maximal (with respect to set inclusion) set of nodes such that there exists a path between every pair of nodes in C containing only undirected edges (respectively bidirected edges). We denote the set of all connectivity components in a CG G by $cc(G)$ and the component to which a set of nodes X belong in G by $co_G(X)$. A *subgraph* of G is a subset of nodes and edges in G. A subgraph of G induced by a set of its nodes X is the graph over X that has all and only the edges in G whose both ends are in X. With the *skeleton* of a graph G we mean a graph with the same adjacencies as G but where all edges have been replaced by undirected edges.

To illustrate these concepts we can study the AMP CG G with five nodes shown in Figure 3a. In the graph we can for example see that the only child of x is y and that p is a neighbour of q. p is also a strict descendant of x due to the strictly descending path $x \rightarrow y \rightarrow p$, while q is not. q is however in the descendants of x together with y and p. x is therefore an ancestor of all nodes except itself and z. We can also see that G contains no semi-directed cycles since it contains no cycle at all. Moreover we can see that G contains four connectivity components: $\{x\}, \{y\}, \{z\}$ and $\{p, q\}$. In Figure 3b we can see a



(a) An AMP CG G (b) The subgraph of G induced by $\{y, p, q\}$ (c) An LDG H

Figure 3: Three different AMP CGs

subgraph of G with the nodes y, p and q. This is also the induced subgraph of G with these nodes since it contains all edges between the nodes y, p and q in G.

Let X, Y and Z denote three disjoint subsets of V. We say that X is *conditionally independent* from Y given Z if the value of X does not influence the value of Y when the values of the variables in Z are known, i.e. $pr(X, Y|Z) = pr(X|Z)pr(Y|Z)$ holds and $pr(Z) > 0$. When it comes to graphs we say that X is *separated* from Y given Z denoted as $X \perp_G Y|Z$ if the following criterion is met: If G is a LWF CG then X and Y are separated given Z iff there exists no route between X and Y such that every node in a non-collider section on the route is not in Z and some node in every collider section on the route is in Z or $an_G(Z)$. If G is a MVR CG or BG then X and Y are separated given Z iff there exists no d-connecting path between X and Y. A path is said to be *d-connecting* iff every non-collider on the path is not in Z and every collider on the path is in Z or $san_G(Z)$. A node b is said to be a *collider* in a MVR CG or BG G between two nodes a and c on a path if one of the following configurations exists in G: $a \rightarrow b \leftarrow c$, $a \rightarrow b \leftrightarrow c$,$a \leftrightarrow b \leftarrow c$ or $a \leftrightarrow b \leftrightarrow c$. For any other configuration the node b is a non-collider. If G is an AMP CG, DAG or UG X and Y are separated given Z iff there exists no S-open route between X and Y. A route is said to be *S-open* iff every non-head-no-tail node on the route is not in Z and every head-no-tail node on the route is in Z or $san_G(Z)$. A node b is said to be a *head-no-tail* in an AMP CG, DAG or UG G between two nodes a and c on a route if one of the following configurations exists in G: $a \rightarrow b \leftarrow c$, $a \rightarrow b - c$ or $a - b \leftarrow c$. Moreover G is also said to contain a *triplex* $(\{a, c\}, b)$ iff one such configuration exists in G and a and c are not adjacent in G. A triplex $(\{a, c\}, b)$ is said to be a *flag* (respectively a *collider*) in an AMP CG or DAG G iff G contains one following subgraphs induced by a, b and c: $a \rightarrow b - c$ or $a - b \leftarrow c$ (respectively $a \rightarrow b \leftarrow c$). If an AMP CG G is said to contain a *biflag* we mean that G contains the induced subgraph $a \rightarrow b - c \leftarrow d$ where a and d might be adjacent, for four nodes a, b, c and d. Given a graph G we mean with $G \cup \{a \rightarrow b\}$ the graph H with the same structure as G but where H also contains the directed edge $a \rightarrow b$ in addition to any other edges in G. Similarly we mean with $G \setminus \{a \rightarrow b\}$ the graph H with the same structure as G but where H does not contain the directed edge $a \rightarrow b$. Note that if G did not contain the directed edge $a \rightarrow b$ then H is not a valid graph.

To illustrate these concepts we can once again look at the graph G shown in Figure 3a. G contains no colliders but two flags, $y \rightarrow p - q$ and $z \rightarrow q - p$, that together form the biflag $y \rightarrow p - q \leftarrow z$. This means that $y \perp_G q$ holds but that $y \perp_G q | p$ does not hold since p is a head-no-tail node on the route $y \rightarrow p - q$.

The *independence model* M induced by a graph G, denoted as $I(G)$, is the set of separation statements $X \perp_G Y | Z$ that hold in G. We say that two graphs G and H are *Markov equivalent* or that they are in the same *Markov equivalence class* iff $I(G) = I(H)$. Moreover we say that G and H belong to the same *strong Markov equivalent class* iff $I(G) = I(H)$ and G and H contain the same flags. By saying that an independence model is *perfectly represented* in a graph G we mean that $I(G)$ contains all and only the independences in the independence model.

An AMP CG *model* (respectively DAG model, UG model, BG model, LWF CG model or MVR CG model) is an independence model representable by an AMP CG (respectively DAG, UG, BG, LWF CG or MVR CG). AMP CG models do today have two possible unique graphical representations, *largest deflagged graphs* (LDGs) [15] and *AMP essential graphs* [2]. In this article we will only use LDGs. An LDG is the AMP CG of a Markov equivalence class that has the minimum number of flags while at the same time contains the maximum number of undirected edges for that strong Markov equivalence class. LWF CG models only have one unique graphical representation, the *largest chain graphs* (LCGs) [6]. A LCG is the LWF CG of a Markov equivalence class with the maximum number of undirected edges. Similarly MVR CG models only have one unique graphical representation, the essential MVR CGs [17]. An essential MVR CG of a Markov equivalence class is a graph with the same skeleton as every MVR CG in that Markov equivalence class and which contain an arrowhead on an edge iff that arrowhead exists in every MVR CG in the Markov equivalence class. Hence these graphs might contain three types of edges, directed, undirected and bidirected, and are therefore not strictly chain graphs.

For AMP CGs there exists a set of operations that allows for changing the structure of edges within an AMP CG without altering the Markov equivalence class it belongs to. In this article we use the *feasible split* operation [16] and the *legal merging* [15] operation. A split is said to be feasible for a connectivity component C of an AMP CG G iff it can be divided into two disjoint non-empty sets U and L such that $U \cup L = C$ and if replacing all undirected edges between U and L with directed edges oriented towards L results in an AMP CG H such that $I(G) = I(H)$. It has been shown that such a split is possible iff the following conditions hold in G: [16] (1) $\forall v_i \in ne_G(L) \cap U, L \subseteq ne_G(v_i)$, (2) $ne_G(L) \cap U$ is complete and (3) $\forall v_j \in L, pa_G(ne_G(L) \cap U) \subseteq pa_G(v_j)$.

A merging is on the other hand said to be legal for two connectivity components U and L in an AMP CG G, such that $U \in pa_G(L)$, iff replacing all directed edges between U and L with undirected edges results in an AMP CG H such that G and H belong to the same strong Markov equivalence class. It has been shown that such a merging is possible in G iff the following conditions hold [15]: (1) $pa_G(L) \cap U$ is complete in G, (2) $\forall v_j \in pa_G(L) \cap U, pa_G(L) \setminus U = pa_G(v_j)$ and (3) $\forall v_i \in L, pa_G(L) = pa_G(v_i)$. Note that a legal merging is not the reverse

operator of a feasible split since a feasible split handles Markov equivalence classes, while a legal merging handles strong Markov equivalence classes.

If we once again look at the CGs in Figure 3 we can see that G, shown in Figure 3a, and H, shown in Figure 3c, are Markov equivalent since $I(G) = I(H)$. Moreover we can note that G and H must belong to the same strong Markov equivalence class since they contain the same flags. This means that G cannot be an LDG since H is larger than G, i.e. contains more undirected edges. In G it exists no feasible split, but one legal merging which is the merging of the connectivity components $\{x\}$ and $\{y\}$ that results in a CG with the same structure as H. In H it does on the other hand exist no legal merging but one feasible split which is the split of the connectivity component $\{x, y\}$ into either $x \rightarrow y$ or $y \rightarrow x$. We can finally note that this feasible split would result in no additional flags and hence, since no legal mergings are possible in H, that H must be an LDG.

Proofs for Theorems in Section 3

Theorem 1. *The operators in Definition 1 fulfill the aperiodicity property when G contains at least two nodes.*

Proof. To prove this we need to show that there exists at least one operator such that H is equal to G, and at least one operator such that it is not, for any possible G with at least two nodes. The latter follows directly from Theorem 3 since there exist more than one possible state when G contains at least two nodes. To see that the former must hold note that if the add directed edge operation results in an LDG for some nodes x and y in an LDG G, then clearly the remove directed edge $x \rightarrow y$ operation must result in an LDG H equal to G for that G. □

Theorem 2. *The operators in Definition 1 fulfill the reversibility property for the uniform distribution.*

Proof. Since the desired distribution for the Markov chain is the uniform distribution proving that reversibility holds for the operators simplifies to proving that symmetry holds for them. This means that we need to show that for any LDG G the probability to transition to any LDG H is equal to the probability to transition from H to G. Here it is simple to see that each operator has a reverse operator such as remove directed edge for add directed edge etc. and that the "forward" operator and "reverse" operator are chosen with equal probability for a certain set of nodes. Moreover, with the exception of add respectively remove undirected edge operator, we can also see that if H is G with one operator performed upon it, such that $H \neq G$, then clearly there can exist no other operator that transition G to H. For the add respectively remove undirected edge operator we can see that two operators transforms an LDG G into the same H, i.e. add $x - y$ respectively add $y - x$, but that there also exist two reverse operators that transforms H into G. Hence the operators fulfill the symmetry property and thereby the reversibility property. □

Theorem 3. *Given an LDG G any other LDG H can be reached using the operators described in Definition 1 such that all intermediate graphs are LDGs.*

Proof. To show that the theorem hold we must prove prove that any LDG H can be reached from the empty graph $G_\emptyset$ since we, due to the reversibility property, then also know that $G_\emptyset$ can be reached from H (or G). A procedure to reach H from $G_\emptyset$ is detailed in Algorithm 2 and its correctness follows from Theorem 7. □

Theorem 4. *Given an LDG G there exists a DAG H such that G and H represent the same independence model iff G contains no flags and G contains no chordless undirected cycles.*

Proof. Since H is a DAG and DAGs is a subclass of AMP CGs we know that for $I(G) = I(H)$ to hold H must be in the same Markov equivalence class as G if it is interpreted as an AMP CG. However, since G is an LDG and contains a flag we know that that flag must exist in every AMP CG in the Markov equivalence class of G. Hence $I(G) = I(H)$ cannot hold if G contains a flag. Moreover it is also well known that the independence model of an UG containing chordless cycles cannot be represented as a DAG. On the other hand, if G contains no flag and no chordless undirected cycles, then clearly all triplexes in G must be unshielded colliders. Hence, together with the fact that G cannot contain semi-directed cycles, we can orient all undirected edges in G to directed edges without creating any semi-directed cycles as shown by Koller and Friedman [8, Theorem 4.13]. This means that the resulting graph must be a DAG H such that $I(G) = I(H)$. □

Theorem 5. *Given an LDG G there exists an UG H such that G and H represent the same independence model iff G contains no directed edges.*

Proof. This follows directly from the fact that G contains a triplex iff it also contains a directed edge and UGs cannot represent any triplexes. It is also clear that if G contains no directed edges than it can only contain undirected edges and hence be interpreted as an UG. □

Theorem 6. *An AMP CG G is an LDG iff no legal merging is possible in G and there exists no sequence of feasible splits of G such that the resulting graph H contains fewer flags than G.*

Proof. From the definition of an LDG it directly follows that G is not an LDG if a legal merging exists for G or if there exists another AMP CG H with fewer flags than G such that $I(H) = I(G)$. To see that G must be an LDG if the conditions are fulfilled we can note that the maximally oriented AMP CG can be reached by iteratively repeating feasible splits onto G [16, Theorem 4]. This graph must contain the minimum number of flags for any AMP CG in the Markov equivalence class of G, and hence belong to the same strong Markov equivalence class as the LDG of the Markov equivalence class of G. From the maximally oriented AMP CG it has then been shown that the LDG can be reached by iteratively applying legal mergings [15, Proposition 16]. □

Given an LDG H the following algorithm constructs G from the the empty graph using the operators in Definition 1 such that $G = H$ when the algorithm terminates and all intermediate graphs are LDGs.

1 Let G be the empty graph containing no nodes.
2 Repeat until $G = H$:
3 Let C be a connectivity component in H such that the subgraphs of G and H induced by $an_H(C)$ have the same structure. Also let H_C be the subgraph of H induced by C and the nodes in G.
4 If all nodes in C have the same set of parents in H:
5 Perform the steps shown in Algorithm 3 with $C = C, G = G$ and $H = H_C$.
6 Else: (when all nodes in C do not have the same set of parents)
7 Perform the steps shown in Algorithm 4 with $C = C, G = G$ and $H = H_C$.
8 Restart the loop in line 2.

Algorithm 2: Construction Algorithm

Below follows one of the main contributions of this paper, i.e. an algorithm that shows that the operators in Definition 1 can be used to reach any LDG with the given number of nodes. The proof of the algorithms correctness, and the algorithm itself, is however quite long and split into several theorems and subalgorithms each handling different parts of it. Moreover, due to the algorithms complexity, an example illustrating how it works is included in the end of the appendix.

Theorem 7. *Algorithm 2 is correct, i.e. defines a valid sequence of operations such that any LDG H can be reached from the empty graph such that all intermediate graphs are LDGs.*

Proof. To prove this theorem we need to show that all intermediate graphs are LDGs and that $G = H$ when the algorithm terminates.

As can be seen in Algorithm 2 each iteration of the loop in line 2 consists of taking one connectivity component C and add it to G until G and H have the same structure. C is chosen in such a way, in line 3, that the subgraphs of G and H induced by the $an_H(C)$ have the same structure. After each iteration we then know, as shown by Lemmas 8 and 10, that the subgraphs of G and H induced by $an_G(C) \cup C$ have the same structure.

To see that we can add the connectivity components this way it is enough to note that neither the conditions for when a split is feasible nor the conditions for when a merging is legal takes the children or descendants of C respectively L into consideration. This means that when we check if a connectivity component fulfills the criteria for an LDG we only look at the connectivity component itself and its ancestors. In turn this also means that an LDG G can only be an LDG if $\forall C \in cc(G)$ the subgraphs induced by $C \cup an_G(C)$ are LDGs. Moreover, this also means that when adding new directed edges to, or undirected edges within, a component C chosen in line 3 in a graph G, we only have to check if a sequence of splits, that removes some flag, is feasible in C and not the whole

Given an LDG H with a component C, such that $\forall v_i \in C$ $pa_H(v_i) = pa_H(C)$ and $ch_H(C) = \emptyset$, and an LDG G, such that $G = H \setminus C$, the following algorithm transforms G into H using the operators in Definition 1 such that all intermediate graphs are LDGs.

1 Add the nodes of C to G.
2 Repeat for all nodes $x \in C$:
3 Apply the first applicable line until $pa_G(x) = pa_H(x)$:
4 If there exist two nodes $y, z \in pa_H(x) \setminus pa_G(x)$ such that $y \notin ad_H(z)$ then add $y \to x, z \to x$ to G with the add two directed edges operation.
5 If there exist two nodes $y, z \in pa_H(x)$ such that $z \in pa_G(x)$, $y \notin pa_G(x)$ and $y \notin ad_H(z)$ then add $y \to x$ to G with the add directed edge operation.
6 If there exist a node $y \in pa_H(x) \setminus pa_G(x)$ such that $G \cup \{y \to x\}$ is an LDG then add $y \to x$ to G with the add directed edge operation.
7 For any two nodes x and y in C such that $x - y$ is in H but not in G, add $x - y$ to G.

Algorithm 3: Component addition algorithm for when $\forall v_i \in C$ $pa(v_i) = pa(C)$

graph G. This is because we already know that the subgraph of G induced by the ancestors of C must be an LDG. Similarly we only have to consider C and its parent components when we check if some mergings are legal in G.

Finally, assuming that for each C chosen in line 3 the subgraphs of G and H induced by $C \cup an_H(C)$ have the same structure after the iteration, it is easy to see that the loop must terminate when $G = H$ due to the top down structure of components in H. □

Corollary 1. Given an LDG H and an LDG G such that $G = H \setminus C$ where C is a connectivity component such that $ch_H(C) = \emptyset$, it is sufficient to check whether a sequence of splits is feasible in C or a merging legal for the nodes in $C \cup pa_G(C)$ for determining whether G is an LDG when edges in H are added to G.

Lemma 8. *Algorithm 3 is correct, i.e. transforms an LDG G into an LDG H, if $G = H \setminus C$ and C is a component in H such that all nodes in C have the same parents and $ch_H(C) = \emptyset$, using the operators in Definition 1 such that all intermediate graphs are LDGs.*

Proof. First note that C does not contain any flags. This means, together with corollary 1, that for a condition for a legal merging of C and some parent component to fail, the same condition must fail for every node in C, regardless of its neighbours in C.

From this it follows that we can add the parents to each node $x \in C$, chosen in line 2, independently of the other nodes in C and that any merging must be illegal due to the parents of x when $pa_G(x) = pa_H(x)$. To see that all intermediate graphs are LDGs when the parents of x in H are added to x in G we can study the loop in line 3 and the graph altering lines 4, 5 and 6. First note

that no mergings can be legal. For lines 4 and 5 this follows directly from the fact that parents added by these lines are part of unshielded colliders, and hence no merging can be legal between C and some parent component of C. For line 6 it does on the other hand follow from the condition that the resulting graph must be an LDG. To see that $pa_G(x) = pa_H(x)$ must hold when line 6 is no longer applicable assume the contrary, i.e. that the set $S = pa_H(x) \setminus pa_G(x)$ is non-empty. We then know that $\forall s_i \in S\ pa_H(x) \subseteq ad_H(s_i)$ must hold, or line 4 or 5 would have been applicable. Moreover we know that $\forall s_i \in S\ pa_H(s_i) \subseteq pa_H(x)$ and $de_H(s_i) \cap pa_G(x) = \emptyset$ must hold or the graph $G \cup \{s_i \rightarrow x\}$ must be an LDG and hence that line 6 would be applicable. This does however mean that there exists a node $y \in S$ such that $de_H(y) \cap S = \emptyset$ holds and hence for which the merging $U = co_H(y)$ and $L = C$ is legal in H which is a contradiction. Hence we must have that the loop in line 3 is executed such that all intermediate graphs are LDGs and $\forall x \in C\ pa_G(x) = pa_H(x)$ holds afterwards.

For line 7 we can note that no flag removing sequence of splits can exist in C since all nodes in C have the same parents. Hence each undirected edge can be added independently without taking the structure of C into consideration while no mergings are legal as described above. □

Before we continue with the last part of the proof of irreducibility we will first define the terms *subcomponent*, *subcomponentorder*, *strong edge* and *strong undirected path* and make some conclusions about these terms. Let the subcomponents of a connectivity component C, in an AMP CG H, be the different components that are subsets of C in the maximally oriented AMP CG H' of H, i.e. H where all feasible splits have been performed, iteratively. This means that H and H' belong to the same strong Markov equivalence class and have the same structure, with the exception that H' might have directed edges where H have undirected edges. We thereby know that a component C in H can consist of several subcomponents in H' and denote these subsets of nodes $C^1, C^2, ..., C^n$. We also define an order of these subcomponents such that C^k must have a lower order than C^l if C^k is an ancestor of C^l in H'. In short this means that the subcomponent with lowest order has no parents in C in H'.

With the term strong edge in an LDG G we mean an edge that must exist in every AMP CG in the Markov equivalence class of G. For example, if an undirected edge $x-y$ is strong in an AMP CG G, then there exists no sequence of feasible splits in G that orients this edge into $x \rightarrow y$ or $y \rightarrow x$. With strong undirected path we mean a path of strong undirected edges.

Lemma 9. *Given an LDG H with a component C, such that $\exists v_i \in C$ for which $pa_H(v_i) \neq pa_H(C)$ and $ch_H(C) = \emptyset$, let H' be the maximally oriented AMP CG of H and C^1 the subcomponent of C in H' with the lowest order. The following statements must then hold:*

1. *No splits are feasible within the subcomponents of C.*
2. *All nodes in $C \setminus C^1$ must be strict descendants of C^1 in H'.*

3. *For every node $c_k \in C \setminus C^1$ that is a strict descendant of some node $c_l \in C^1$ it must hold that $pa_H(c_l) = pa_H(c_k)$.*

4. *Every set of parents that exists to some node in C must also exist to some node in C^1.*

5. *C^1 must be unique, i.e. consists of the same set of nodes no matter how H is split.*

6. *$\forall c_k \in C^i$, $i \neq 1$, $pa_H(c_k) = pa_H(nb_H(c_k))$.*

Proof. Each statement is proved separately:

(1): This follows directly from the way subcomponents are defined.

(2): To see this note that for a split to be feasible the adjacent nodes of L in U must be adjacent of all nodes in L. This means that if there exists a subcomponent C^i such that $\exists c_k \in C^i$, $c_k \in nb_H(C^1)$, $c_k \in ch_{H'}(C^1)$ then any node in $C^1 \cap nb_H(C^i)$ must be a parent of all nodes in C^i. This in turn means that if there exists a subcomponent C^j such that $\exists c_l \in C^j$, $c_l \in nb_H(C^i)$, $c_l \in ch_{H'}(C^i)$ then any node in $C^i \cap nb_H(C^j)$ must be a parent of all nodes in C^j. From this it then iteratively follows that all nodes in $C \setminus C^1$ must be strict descendants of C^1 in H'.

(3) First we will show that for every node $c_k \in C \setminus C^1$ that is a strict descendant of some node $c_l \in C^1$ it must hold that $pa_H(c_k) \subseteq pa_H(c_l)$. To see this assume the contrary and that c_k is a node for which the statement does not hold. A feasible split must then have been performed when H was transformed into H' such that c_k belonged to L and some node c_m belonged to U, such that $pa_H(c_m) \subseteq pa_H(c_l)$ and $pa_H(c_k) \not\subseteq pa_H(c_m)$ hold. This do however lead to a contradiction since such a feasible split would remove a flag from H, which is not possible since H is an LDG. Secondly we will show that for every node $c_k \in C \setminus C^1$ that is a strict descendant of some node $c_l \in C^1$ it must hold that $pa_H(c_l) \subseteq pa_H(c_k)$. To see this assume the contrary and that c_k is a node for which the statement does not hold. Then a split must have been performed when H was transformed into H' such that c_k belonged to L and some node c_m belonged to U, such that $pa_H(c_l) \subseteq pa_H(c_m)$ and $pa_H(c_m) \not\subseteq pa_H(c_k)$. We can then see that condition 3 for a feasible split is not fulfilled which contradicts that H and H' are in the same Markov equivalence class. Hence we have that for every node $c_k \in C \setminus C^1$ that is a strict descendant of some node $c_l \in C^1$ it must hold that $pa_H(c_l) = pa_H(c_k)$.

(4) This follows directly from (3).

(5) This follows directly from (4) and that $\exists v_i \in C$ for which $pa_H(v_i) \neq pa_H(C)$.

(6) To see this note that all nodes c_k for which there exists a path to $c_l \in C^1$ in the subgraph of H induced by $(C \setminus C^1) \cup c_l$ must be strict descendants of c_l in H'. This follows from (2) and that for a split to be feasible all nodes in $U \cap nb_H(L)$ must be adjacent of all nodes in L. Together with (3) it then follows that the statement must hold.

□

Given an LDG H with a component C, such that $\exists v_i \in C$ for which $pa_H(v_i) \neq pa_H(C)$ and $ch_H(C) = \emptyset$, and an LDG G, such that $G = H \setminus C$, the following algorithm transforms G into H using the operators in Definition 1 such that all intermediate graphs are LDGs.

1 Add the nodes in C to G.
2 Let H' be the maximally oriented AMP CG of H and C^1 the subcomponent with lowest order.
3 Let A and B be two empty sets of nodes. Let the nodes in C have an ordering that is updated as nodes are added to the sets A and B and where the order of a node v_i is defined as follows: If $v_i \in A$ then its order is the size of A when it was added to A. If $v_i \in B$ but $v_i \notin A$, then its order is $|C|$ plus the size of A when it was added to B. If it is in neither set let it have a higher order than any node in A or B.
4 If H' contains an induced subgraph of the form $p \rightarrow x - y \leftarrow q$ then:
5 Add first $x - y$ and then both $p \rightarrow x$ and $q \rightarrow y$ at once to G. Add both x and y to A.
6 Else we know that H' must contain an induced subgraph of the form $y - x - z$ such that $\exists p \in pa_H(x)$ and $p \notin pa_H(z)$.
7 Add $y - x$, $x - z$ and $p \rightarrow x$ to G sequentially. Add x to A and y and z to B.
8 For each node $a_i \in A$ and parent $p_j \in pa_H(a_i)$ such that $p_j \notin pa_G(a_i)$ and p_j is not a parent of all nodes in $A \cup B$ in H, add $p_j \rightarrow a_i$ to G.
9 Repeat until line 10 is no longer applicable:
10 If there exist two nodes $x \in A$ and $y \in nb_{H'}(x) \cap C^1$ such that either $\exists p \in pa_H(x) \setminus pa_H(y)$ or $\exists z \in nb_{H'}(y) \cap C^1 \setminus nb_{H'}(x)$ hold then choose x and y in this way but also choose y (and the corresponding x) so that y has the highest possible order:
11 If $\exists p \in pa_H(x) \setminus pa_H(y)$: Add $p \rightarrow x$ to G if it is not in G, then add $x - y$ to G if it is not in G. Add y to A if does not already belong to A.
12 Else (when $\exists z \in nb_{H'}(y) \setminus nb_{H'}(x)$): Add $x - y$ and $y - z$ to G (sequentially) if they are not already in G. Add y to A and z to B if they not already belong to these sets.
13 For each node $a_i \in A$ and parent $p_j \in pa_H(a_i)$ such that $p_j \notin pa_G(a_i)$ and p_j is not a parent of all nodes in $A \cup B$ in H, add $p_j \rightarrow a_i$ to G.
14 For any two nodes x and y in A such that $x - y$ is in H but not in G, add $x - y$ to G.
15 Repeat until $A = C$:
16 Let a_i be the node with highest order in A such that $\exists c_j \in nb_H(a_i) \setminus A$ and add $a_i - c_j$ to G and c_j to A.
17 Repeat until $\forall a_i \in A \; pa_H(a_i) = pa_G(a_i)$:
18 Let x be the node with lowest order in A such that $pa_H(x) \neq pa_G(x)$. Apply the first applicable line until $pa_G(x) = pa_H(x)$:
19 If there exist two nodes $y, z \in pa_H(x) \setminus pa_G(x)$ such that $y \notin ad_H(z)$ then add $y \rightarrow x, z \rightarrow x$ to G with the add two directed edges operation.
20 If there exist two nodes $y, z \in pa_H(x)$ such that $z \in pa_G(x)$, $y \notin pa_G(x)$ and $y \notin ad_H(z)$ then add $y \rightarrow x$ to G with the add directed edge operation.
21 If there exist a node $y \in pa_H(x) \setminus pa_G(x)$ such that $G \cup \{y \rightarrow x\}$ is an LDG then add $y \rightarrow x$ to G with the add directed edge operation.
22 For any two nodes x and y in A such that $x - y$ is in H but not in G, add $x - y$ to G.

Algorithm 4: Component addition algorithm for when $\exists v_i \in C \; pa(v_i) \neq pa(C)$

Lemma 10. *Algorithm 4 is correct, i.e. transforms an LDG G into an LDG H, if $G = H \setminus C$ and C is a component in H such that all nodes in C do not have the same parents and $ch_H(C) = \emptyset$, using the operators in Definition 1 such that all intermediate graphs are LDGs.*

Proof. The algorithm can then be seen to consist of three different phases where in each phase nodes in C are added to the set of nodes A. When a node x is added A, some edges in H, with x as an endpoint, are also added to G such that x becomes adjacent of some previously added node in A. We will show that by adding nodes as described in Algorithm 4 a structure is constructed that fulfills certain properties such as for example that flag removing sequences of splits are not feasible and mergings are not legal in G. In the first phase, lines 4 to 8, an initial structure which fulfills these properties is identified in H and added to G. In the second phase, lines 9 to 13, this structure is extended with the nodes in C^1, i.e. the subcomponent of C in the maximally oriented AMP CG H' of H with lowest order. In the third and last phase the remaining nodes in C are connected to the structure where after the remaining edges in H also are added to G. We can note that it is enough to show that there exists no flag removing sequence of feasible splits within C in G and that no merging is legal between C and some parent component of C in G when determining whether G is an LDG as described in Corollary 1.

Phase 1, lines 4 to 8:
To see that one of the induced subgraphs shown in line 4 respectively 6 must exist in C^1 in H' assume the contrary. Since all nodes in C do not have the same parents we know, together with (4) in Lemma 9, that all nodes in C^1 do not have the same parents. This means that there must exist two nodes $x, y \in C^1$ such that $\exists p \in pa_H(x) \setminus pa_H(y)$. Let L consist of all nodes in C^1 that have the parent p and U of all nodes that do not. We then know that $\forall c_k \in U \cap nb_{H'}(L)\ L \in nb_{H'}(c_k)$ holds or the induced subgraph described in line 6 must exist in H'. Similarly we know that $U \cap nb_{H'}(L)$ must be complete. Finally we also know that $\forall c_l \in L\ pa_{H'}(nb_{H'}(c_l) \cap U) \subseteq pa_{H'}(c_l)$ or the induced subgraph described in line 4 must exist in H'. This does however mean that a split, with the described U and L is feasible in C^1 in H' which is a contradiction to (2) in Lemma 9. We can also note that the intermediate graphs for lines 5 and 7 are LDGs since no splits are feasible (and no mergings possible) when C has not yet received any parents in G. Once it has received parents through lines 5 or 7 it is easy to see that all parents are part of flags that cannot be removed with some feasible split. We can also note that the undirected edges in C in the resulting G are strong.

In line 8 additional parents are then added to the nodes in A. Note however that all these parents must be parts of flags in G, and hence, since the undirected edges in C in G are strong, cannot be part of a legal merging.

Phase 2, lines 9 to 13:
In this phase the remaining nodes in C^1 are added to A. We can here note that from (2) and (3) in Lemma 9 it follows that any flag over some node in C must be over some node in C^1. Moreover, any collider over some node in C

must also exist over a node in C^1. This means that whatever prevents a legal merging of C and any of its parent components also must be preventing a legal merging between C^1 and its parent components in the subgraph of H induced by $an_H(C) \cup C^1$. Hence, since no split can be feasible in C^1, the subgraph of H induced by $an_H(C) \cup C^1$ must be a LDG. From this it follows that we can add C^1 to G first, and then consider the remaining nodes in C separately.

Below we shown that the nodes in C^1 are added in such a way that an undirected path exists between the initial node(s) in A and the now added nodes. We also show that any undirected edge in this path must be strong if any of its endnodes have at least one parent. This prevents any sequence of feasible splits to be flag removing. Moreover, we show that any added parent to any node in C^1 must be part of a flag, preventing any merging to be legal. To do this we will however first need to make some observations about the loop in line 9 and the structure of C in G during this loop:

(a) When the loop terminates it must hold that $A = C^1$. This follows from (2) in Lemma 9, i.e. that no splits are feasible in C^1. Hence, for every non-empty subset of nodes $V_i \subset C^1$ at least one of the conditions for a feasible split with $U = V_i$ and $L = C^1 \setminus V_i$ must fail. If we study the conditions for a feasible split we can note that this can be for two reasons. Either there exist two nodes $c_l \in V_i$ and $c_k \notin V_i$ such that $c_l - c_k$ is in H' and $pa_{H'}(c_l) \not\subseteq pa_{H'}(c_k)$ or there exist two nodes $c_l \in V_i$ and $c_k \notin V_i$ such that $c_l - c_k$ is in H' and $nb_{H'}(c_k) \not\subseteq nb_{H'}(c_l)$. Hence the conditions for line 10 must be fulfilled for any $A \subset C^1$.

(b) The only nodes in C that can have parents are those in A. This follows directly from how parents are added in the loop.

(c) The only nodes in C that can have any neighbours are those in A and B and $A \cup B \subseteq C^1$.

(d) All parents of C must be part of flags. Hence no mergings can be legal. This follows from that a parent only is added to a node x in G if it is not a parent of some node in G for which there exists an undirected path to x in G.

(e) When the loop terminates all nodes in C^1 must have the same parents in G and H with the exception of the parents that are parents of all nodes in C. This follows from the way that parents are added in line 13 and (4) in Lemma 9.

(f) A strong undirected path must exist between any two nodes in C^1 after the loop in line 9 terminates. This follows directly from that no split is feasible in C^1 in H'.

(g) For any edge $x - y$ added in lines 11 or 12, with the x and y described for those lines, that edge must be strong for the current G and all future G if either $pa_G(x) \neq \emptyset$ or $pa_G(y) \neq \emptyset$. To see this assume the contrary and first assume that there exists a feasible split with $x \in U$ and $y \in L$. Clearly $pa_G(x) \subseteq pa_G(y)$ must then hold, which contradicts that the condition for line 11 is fulfilled. For line 12 the edge $x - y$ can obviously not be split directly into $x \to y$ since $ne_G(y) \not\subseteq ne_G(x)$. We can however imagine that there exists a sequence of splits that orients $y - z$ into $y \to z$, thereby making the split of $x - y$ into $x \to y$ feasible. This would then require that $pa_G(x) \subseteq pa_G(y)$, or the split

would not be feasible. This together with the assumption that $pa_G(x) \neq \emptyset$ and $pa_G(y) \neq \emptyset$ means that $pa_G(y) \neq \emptyset$. Moreover, this means that y must belong to A, since only nodes in A in C have parents in G. In turn this means that we can now restart this part of the proof with the edge $y-z$ instead and see why it cannot be feasible split. It then follows that for every edge there has to exist some other edge that would have to be split before the current edge for which the condition that either $pa_G(x) \neq \emptyset$ or $pa_G(y) \neq \emptyset$ must hold. To see that no split is feasible with x in L and $y \in U$ note that this would require y to be adjacent of all nodes in $A \cup B$ in G which in turn would require $A = C^1$ due to order in which nodes are chosen as y in line 10. For a split to be feasible it would also have to hold that $\forall a_i \in A \cup B \; pa_G(y) \subseteq pa_G(a_i)$, and, since $A = C^1$, that all nodes in C^1 have the same parents in G and H with the exception of the parents that are parents of all nodes in C as discussed in (e). This is however a contradiction since the same split then would be feasible in H'.

With these observations we can now see that line 11 can be performed and that the intermediate graphs must be LDGs. $p \rightarrow x$ can be added since all edges $w-x$ in G for all nodes $w \in nb_G(x)$ must be strong after x has received p as a parent as described in (g). That $x-y$ also is strong follows from (g). For line 12 we can note that a split is feasible of $x-y$ to $x \rightarrow y$ when only $x-y$ have been added to G. Such a split does however not remove any flag. Later, when $y-z$ has been added we also know that the resulting graph must be an LDG as described in (e).

For line 13 we can note that any added parent must be part of a flag in G and hence no legal mergings can be possible. Moreover it follows from (g) that no sequence of feasible splits can remove a flag.

Phase 3, lines 14 to 22:
That line 14 must result in LDGs follows from the fact that a strong undirected path must exist between any two nodes in C^1 as described in (f) and that this path cannot be made weak by adding additional undirected edges that exist in H. Hence, after line 14 the subgraphs of H and G induced by C^1 must have the same structure. Note however that C^1 might have some parents in H that do not exist in G, but only if those parents are parents of all nodes in C as described in (e). These last parents are handled in the loop in line 17.

The loop in line 15 then creates several tree structures, each with a root node in C^1, that extends A to consist of all nodes in C. Note that any two nodes c_i and c_j must belong to the same tree structure if there exists an undirected path between c_i and c_j in the subgraph of H induced by $C \setminus C^1$. That no sequence of feasible splits, that removes a flag, can exist in G follows from the fact that only nodes in C^1 can have parents and no node not in C^1 can be adjacent of all nodes in C^1 in G before line 22.

In the loop in line 17 the parents of the nodes in C are added to G. We can now note that in each subtree all nodes must have the same parents as shown in (6) in Lemma 9. Moreover the parents are added from the root and outwards, adding the parents to the leafs last. This gives that no edge $x-y$, such that x has a lower order than y, can be oriented into $x \rightarrow y$ or $y \rightarrow x$ by a sequence of feasible splits that removes a flag from G. If both $x, y \in C^1$ this

follows from (f). If $x \notin C^1$ or $y \notin C^1$ orienting the edge to $x \rightarrow y$ through a feasible split cannot be flag removing since y only has a parent if x has the same parent. Nor can it exist any feasible split that orients the edge to $y \rightarrow x$ since $\exists z \in nb_G(x) \setminus nb_G(y)$. Moreover, once both x and y has received their parents it must hold that $pa_G(x) = pa_G(y)$ as shown in (3) in Lemma 9 and hence any sequence of feasible splits cannot be flag removing. That no mergings are legal before the last node receives its parents follows from that all parents are part of flags. When the last node x, with the highest order in C, receives its parents we can also note that no mergings can be legal. For lines 19 and 20 this follows directly from the fact that every parent added by these lines are part of unshielded colliders, and hence no merging can be legal between C and some parent component of C. For line 21 it does on the other hand follow from the condition that the resulting graph must be an LDG. To see that $pa_G(x) = pa_H(x)$ must hold when the loop in line 17 terminates assume the contrary, i.e. that there exists a node $y \in pa_H(x) \setminus pa_G(x)$. We then know that $pa_H(x) \subseteq ad_H(y)$ must hold, or lines 19 or 20 would have been applicable. We also know that $pa_H(y) \subseteq pa_H(x)$ and $\forall v_i \in C \; y \in pa_H(v_i)$ hold, or the graph $G \cup \{y \rightarrow x\}$ must be an LDG. This does however also mean that a merging is legal in H, since $\forall a_i \in an_H(x) \cup C \setminus x \; bd_G(a_i) = bd_H(a_i)$, which is a contradiction. Hence we must have that $\forall v_i \in C \; pa_H(v_i) = pa_G(v_i)$ when line 22 is reached.

Finally in line 22 the remaining undirected edges can be added to G since all nodes adjacent in C in H but not in G have the same set of parents. Hence no sequence of feasible splits can be flag removing.

□

An Example Illustrating Algorithm 2

In this subsection we show an example of how Algorithm 2 recreates an LDG H from the empty graph such that all intermediate graphs are LDGs using the operators from Definition 1. The LDG H that is to be constructed is shown in Figure 4.

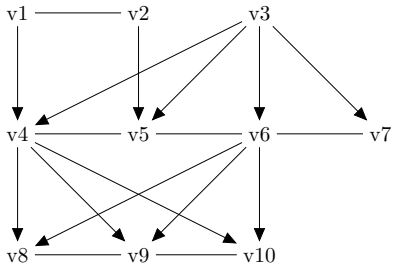
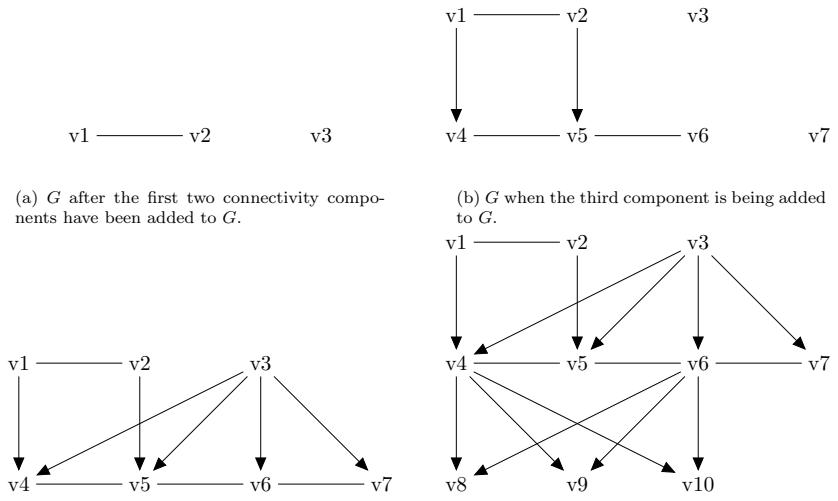


Figure 4: An example LDG H.

The algorithm starts by, in line 3, selecting one connectivity component C in H such that the subgraph of G and H induced by $an_G(C)$ have the same structure. Since G contains no nodes at this time, we can note that this only holds for the components $\{v_1, v_2\}$ and $\{v_3\}$. Let us choose to start with $\{v_3\}$.



(a) G after the first two connectivity components have been added to G.

(b) G when the third component is being added to G.

(c) G when the third connectivity components have been added to G.

(d) G when the last component of H is being added to G.

Figure 5: Intermediate LDGs G during the construction process.

Since $pa_H(v_3) = \emptyset$ we then know that the condition in line 4 is fulfilled and hence that line 5 is performed. Looking in Algorithm 3 we can then note that only line 1 is applicable, since C in this case only contains one node with no parents.

For the second iteration of Algorithm 2 we note that the component $\{v_1, v_2\}$ must be chosen as C, since this is the only component which fulfills the condition in line 3. Once again we have that $pa_H(C) = \emptyset$ and hence that line 5 is performed. In Algorithm 3 line 1 is performed as before but this time line 7 is also applied for the edge $v_1 - v_2$. However, since $pa_H(C) = \emptyset$ lines 2 to 6 is not applicable. The resulting graph after this iteration of the loop in line 2 in Algorithm 1 is then shown in Figure 5a.

For the third iteration the component $\{v_4, v_5, v_6, v_7\}$ must be chosen as C in line 3. In this case the condition in line 4 is not fulfilled, and hence line 6 is performed in Algorithm 2. In Algorithm 4 we can then see that line 1 must be performed, adding the nodes in C to G. We can also note that C^1 must only contain the nodes $\{v_4, v_5, v_6\}$ since a split, with $U = \{v_4, v_5, v_6\}$ and $L = \{v_7\}$, is feasible in H although not flag removing. In phase 1 of Algorithm 4 we can then see that the condition in line 4 is fulfilled for the biflag $v_1 \rightarrow v_4 - v_5 \leftarrow v_2$. This means that in line 5 $v_4 - v_5$ is first added whereafter $v_1 \rightarrow v_4$ and $v_5 \leftarrow v_2$ are added by the add two directed edges operator. We can in line 8 then also note that the parent v_3 is not added to v_4 or v_5 since it is a parent of all nodes in A (i.e. v_4 and v_5). In phase 2 the node v_6 is the only remaining node in C^1 and hence must be chosen as y in line 10 together with v_5 as x. It also fulfills the condition for this since $v_2 \in pa_H(v_5) \setminus pa_H(v_6)$. This means that line 11 is

performed adding the edge $v_5 - v_6$ to G. Once again we can note that v_3 is not added as parent in line 13 since it is a parent of all nodes in A (v_4, v_5 and v_6). The resulting graph is seen in Figure 5b. Since line 10 in Algorithm 4 is not applicable for any other node in C^1 phase 3 now starts where the edge $v_6 - v_7$ is added to G in line 16. After this $A = C$ and the parents of C are to be added in the loop in line 17. This is done by first adding $v_3 \rightarrow v_4$ to G and then the edges $v_3 \rightarrow v_5$, $v_3 \rightarrow v_6$ and $v_3 \rightarrow v_7$ sequentially. For the first two edges this is done in line 20 due to the unshielded colliders with v_1 respectively v_2 while for the last two edges this is done in line 21. We can then note that line 22 is not applicable for any edge in H and that G have the structure shown in Figure 5c when the third iteration of line 2 in Algorithm 2 finishes.

For the fourth and last iteration of the loop in line 2 in Algorithm 2 we see that C must be chosen as $\{v_8, v_9, v_{10}\}$. In this case the condition in line 4 is fulfilled, meaning that line 5 is performed and that Algorithm 3 is executed. This in turn means that line 1 in Algorithm 3 first is applied, adding the nodes in C to G. After this one of the nodes in C is chosen as x in line 2, let's say v_8, and the parents in H is added to this node in G. In this case this is done in line 4 since v_4 and v_6 form an unshielded collider over v_8. This process is then repeated for all nodes in C, resulting in the LDG seen in Figure 5d. Finally the last two undirected edges of C are also added sequentially in line 7 of Algorithm 3 whereafter G and H have the same structure, and the loop in line 2 in Algorithm 2 terminates.

Acknowledgments

This work is funded by the Swedish Research Council (ref. 2010-4808).

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

[2] S. A. Andersson and M. D. Perlman. Characterizing Markov Equivalence Classes For AMP Chain Graph Models. *The Annals of Statistics*, 34:939–972, 2006.

[3] D. R. Cox and N. Wermuth. Linear Dependencies Represented by Chain Graphs. *Statistical Science*, 8:204–218, 1993.

[4] D. R. Cox and N. Wermuth. *Multivariate Dependencies: Models, Analysis and Interpretation*. Chapman and Hall, 1996.

[5] M. Drton. Discrete Chain Graph Models. *Bernoulli*, 15:736–753, 2009.

[6] M. Frydenberg. The Chain Graph Markov Property. *Scandinavian Journal of Statistics*, 17:333–353, 1990.

[7] O. Häggström. *Finite Markov Chains and Algorithmic Applications*. Cambridge University Press, 2002.

[8] D. Koller and N. Friedman. *Probabilistic Graphcal Models*. MIT Press, 2009.

[9] S. L. Lauritzen and N. Wermuth. Graphical Models for Association Between Variables, Some of Which are Qualitative and Some Quantitative. *The Annals of Statistics*, 17:31–57, 1989.

[10] Z. Ma, X. Xie, and Z. Geng. Structural Learning of Chain Graphs via Decomposition. *Journal of Machine Learning Research*, 9:2847–2880, 2008.

[11] J. M. Peña. Approximate Counting of Graphical Models Via MCMC. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, pages 352–359, 2007.

[12] J. M. Peña, D. Sonntag, and J. Nielsen. An Inclusion Optimal Algorithm for Chain Graph Structure Learning. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, pages 778–786, 2014.

[13] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[14] R. W. Robinson. Counting Labeled Acyclic Digraphs. *New Directions in the Theory of Graphs*, pages 239–273, 1973.

[15] A. Roverato and M. Studený. A Graphical Representation of Equivalence Classes of AMP Chain Graphs. *Journal of Machine Learning Research*, 7:1045–1078, 2006.

[16] D. Sonntag and J. M. Peña. Chain Graph Interpretations and Their Relations. *International Journal of Approximate Reasoning*, 58:39–56, 2014.

[17] D. Sonntag, J. M. Peña, and M. Gómez-Olmedo. Approximate Counting of Graphical Models Via MCMC Revisited. *International Journal of Intelligent Systems*, 30:384–420, 2015.

[18] B. Steinsky. Enumeration of Labelled Chain Graphs and Labelled Essential Directed Acyclic Graphs. *Discrete Mathematics*, 270:266–277, 2003.