The Greedy Fast Causal Inference (GFCI) Algorithm for Continuous Variables

This document provides a brief overview of the GFCI algorithm, focusing on a version of GFCI that works with continuous variables, which is called GFCI-continuous (GFCIc).

Purpose

GFCIc [Ogarrio, 2016] is an algorithm that takes as input a dataset of continuous variables and outputs a graphical model called a PAG (see the appendix), which is a representation of a set of causal networks that may include hidden confounders[1]. The PAG that GFCIc returns serves as a data-supported hypothesis about causal relationships that exist among the variables in the dataset. Such models are intended to help scientists form hypotheses and guide the design of experiments to investigate these hypotheses. As mentioned, GFCIc does not presuppose that there are no hidden confounders.

Methodological Approach

GFCIc is a combination of the FGESc [CCD-FGES, 2016] algorithm and the FCI algorithm [Spirtes, 1993] that improves upon the accuracy and efficiency of FCI. In order to understand the basic methodology of GFCIc, it is necessary to understand some basic facts about the FGESc and FCI algorithms.

The FGESc algorithm [Ramsey, 2015; CCD-FGES, 2016] is a score-based greedy search algorithm that takes as input sample data and optional background knowledge, and in the large sample limit outputs an equivalence class of CBNs[2] that receives the highest score on the sample data. When its assumptions are satisfied, It is a fast and accurate algorithm; however, it presupposes that there are no unmeasured confounders in the true model.

The FCI algorithm is a constraint-based algorithm that takes as input sample data and optional background knowledge and in the large sample limit outputs an equivalence class of CBNs that (including those with hidden confounders) that entail the set of conditional independence relations judged to hold in the population. It is limited to several thousand variables, and on realistic sample sizes it is inaccurate in both adjacencies and orientations.

FCI has two phases: an adjacency phase and an orientation phase. The adjacency phase of the algorithm starts with a complete undirected graph and then performs a sequence of conditional independence tests that lead to the removal of an edge between any two adjacent variables that are judged to be independent, conditional on some subset of the observed variables; any conditioning set that leads to the removal of an adjacency is stored. After the adjacency phase, the resulting undirected graph has the correct set of adjacencies, but all of the edges are unoriented. FCI then enters an orientation phase that uses the stored conditioning sets that led to the removal of adjacencies to orient as many of the edges as possible.

---

[1] A hidden confounder is an unmeasured process (variable) that casually influences two or more measured variables. Measured variables can be statistically dependent due to the presence of hidden confounders. The possible presence of hidden confounders in biomedical data is a key reason that causal discovery from observational biomedical data is challenging.

[2] A CBN structure is a directed acyclic graph in which nodes represent variables and arcs represent direct causation among the nodes, where the meaning of *direct* is relative to the nodes in the CBN. For further information about CBNs, see [Spirtes, 2010; Lagani, 2016; Pearl 2016].

The FGESc algorithm is used to improve the accuracy of both the adjacency phase and the orientation phase of FCI by providing a more accurate initial graph that contains a subset of both the non-adjacencies and orientations of the final output of FCI. The initial set of non-adjacencies given by FGESc is augmented by FCI performing a set of conditional independence tests that lead to the removal of some further adjacencies whenever a conditioning set is found that makes two adjacent variables independent. After the adjacency phase of FCI, some of the orientations of FGESc are then used to provide an initial orientation of the undirected graph that is then augmented by the orientation phase of FCI to provide additional orientations. The final output is a PAG (see the Appendix).

Additional details about the GFCIc algorithm are available at [Ogarrio, 2016].

Input Data and Parameters

GFCIc has the following requirements for data input:

- the (training) data are in a table in which columns represent variables, rows represent samples, and the value of each variable in a sample is continuous.
- the first row of the table lists the variable names, which should be unique; the data and variable names are separated by a delimiter (default: tab).

Some of the key parameters taken by GFCI are as follows:

- alpha: Significance level used in conditional independence tests.
- data: The data file.
- exclude-variables: A file containing variables to exclude.
- faithfulness-assumed: If set to “yes, it means that whenever two variables are marginally independent, then FGESc will not consider adding an edge between them. In that case, the algorithm is speeded up, but may make additional errors in the output in certain unusual circumstances. The default value is "yes."
- knowledge: The user may specify knowledge by providing a file that describes precedence and required and/or forbidden edges in the structure that is output. By default, the algorithm assumes no prior knowledge about the causal graph structure
- max degree: The maximum degree of the graph allowed in the FGES phase of GFCI.
- out: Output directory for the file containing the output PAG.
- penalty-discount: The specification of a complexity penalty parameter c that is used in the BIC equation in FGESc [see CCD-FGES, 2016]. The default value is 4.
- thread: Number of threads to use in running GFCI when multiple cores are available.

Additional GFCI parameters are described in
http://www.ccd.pitt.edu/wiki/index.php?title=Documentation_on_using_the_causal-cmd_software

Output

In the large sample limit, the GFCIc algorithm outputs a single PAG [Spirtes 1999; Zhang 2008] that entails the set of conditional independence relations judged to hold in the population represented by its input dataset.

Algorithmic Assumptions

This section describes a sufficient set of assumptions for the application of GFCIc to achieve the guarantees described in the next section. While the pattern output by GFCIc may still include correct edges (and perhaps many correct edges) even if one or more of these assumptions are violated, there are no theoretical guarantees it will do so.

A sufficient set of conditions for recovering the causal structure of the data-generating process in the large sample limit (i.e., as the sample size grows without bound) is as follows: Assume that the causal process generating the data *D* given to GFCIc is accurately modeled by a CBN containing only continuous variables, some of which may not be measured, which we call ***G***. Assume that each variable (node) in ***G*** is a linear function of its parents with Gaussian noise. These constitute a sufficient set of conditions to yield the guarantees below.

While the above procedure is simple, it includes several assumptions that may not be immediately obvious. Key among them are the following:

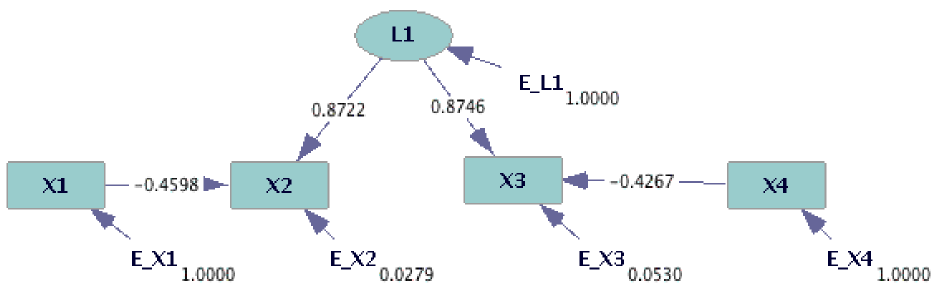
- cases (samples) in the data *D* are independent and identically distributed.
- in a causally sufficient set of variables, the causal Markov condition holds [Spirtes, 2010]. This condition states that a variable is independent of its non-effects, given its direct causes (parents). It expresses a form of local causality.
- the causal faithfulness condition holds with probability 1 [Spirtes, 2010]. This condition states that all the independence relationships among the measured variables are implied by the causal Markov condition.
- there are no missing data. The user must fill in missing data before running GFCIc. Many statistical packages provide methods for handling missing values, including imputing them.
- there is no selection bias [Spirtes, 1999]. This means that the chance a case (sample) was selected from the population for inclusion in dataset *D* did not depend on the values of any of the measured variables in the data.
- there are no feedback cycles among the measured variables. Extensions to CBNs, such as causal Dynamic Bayesian Networks (DBNs) [Neapolitan, 2003], do allow feedback cycles, but they are not currently implemented in GFCIc.

Structure Learning Performance Guarantees

If the assumptions in the previous section hold and the triangle heuristic admits all adjacencies that need to be removed[3], then in the large sample limit the PAG output by GFCIc (see the appendix) will contain edges that are each correct.

Practice Dataset

We used the CBN shown in Figure 1 to generate simulated data containing 300 cases downloadable from the following link Media:Gfcic practice data.txt . In Figure 1 the numbers next to the directed edges represent linear coefficients, and the numbers next to the error terms represent the variances of the error terms, e.g. X2 = 0.4598 • X1 + 0.8722 • L1 + EX2, where EX2 has variance 0.0279. The user may wish to apply GFCIc (with its default settings) to the dataset and verify that the PAG structure obtained is the one shown in Figure 2.



.

Figure 1. The CBN structure used to generate the practice dataset.

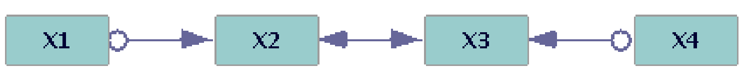


Figure 2. The PAG structure output by GFCI when given data in Media:Gfcic practice data.txt

Performance on Simulated Data

Ogarrio et al. [Ogarrio, 2016] used simulated data on 1000 variables to evaluate GFCI and several other algorithms that also model latent confounding. The results are shown in Tables 1 and 2. The primary measures of performance are precision, recall, and run time on a single processor. These measures are averaged over more than 100 runs, where each run consists of data generated from a different data-generating Bayesian network. The study used a machine with a single Intel I7 3.4Ghz processor running Ubuntu 14.04 with 20G memory allocated.

Overall, GFCI had the best precision and recall for adjacency, arrow/tail, and edge-type determination. Although GFCI was generally slower than the other algorithms, it only required a

[3] The triangle heuristic removes an adjacency between variables *A* and *B* from the output of FGES only when that adjacency occurs in a triangle (i.e., there is a third variable *C* such that *A*, *B*, and *C* are all pairwise adjacent in the output of FGES.) This speeds up GFCI. It has not been proved that the algorithm is correct when the heuristic is employed, although we have found no case where the heuristic is incorrect. In the next version of GFCI, there will be a switch that allows the triangle heuristic to be turned off, in which case GFCI will be slower, but provably correct under the assumptions described in the section above on Algorithmic Assumptions.

few seconds to run on datasets containing 1000 variables. See [Ogarrio, 2016] for additional details about the evaluation.

Table 1. This table lists the average running time and the average precision (prec.) and recall (rec.) for the different accuracy measurements for every algorithm in every parameterization with 1000 variables and α = 0.01. The bold values represent the best value in that column for that parameterization.

DAG						Adjacency		Arrow		Tail	
#Latents	#Edges	#Samples	Alg.	Runs	Time	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
50	1000	200	GFCI	116	4.776s	**0.98**	**0.81**	**0.96**	**0.70**	**0.78**	**0.77**
			RFCI	116	0.274s	0.77	0.71	0.37	0.56	0.50	0.26
			uRFCI	116	1.858s	0.77	0.71	0.37	0.56	0.50	0.26
			FCI+	116	3.007s	0.77	0.71	0.37	0.56	0.50	0.26
50	1000	2000	GFCI	116	5.739s	**1.00**	**0.94**	**0.98**	**0.88**	**0.86**	**0.96**
			RFCI	116	0.387s	0.74	0.93	0.40	**0.88**	0.60	0.69
			uRFCI	116	2.216s	0.74	0.93	0.40	**0.88**	0.60	0.69
			FCI+	116	4.112s	0.74	0.93	0.40	**0.88**	0.60	0.69
200	1000	200	GFCI	116	2.795s	**0.98**	**0.48**	**0.95**	**0.29**	**0.50**	**0.53**
			RFCI	116	0.171s	0.70	0.45	0.35	**0.29**	0.39	0.17
			uRFCI	116	1.494s	0.70	0.45	0.35	**0.29**	0.39	0.17
			FCI+	116	2.109s	0.70	0.45	0.35	**0.29**	0.39	0.17
200	1000	2000	GFCI	116	4.981s	**1.00**	0.69	**0.95**	0.52	0.41	**0.83**
			RFCI	116	0.319s	0.71	**0.71**	0.40	**0.59**	**0.42**	0.52
			uRFCI	116	2.051s	0.71	0.70	0.40	**0.59**	**0.42**	0.52
			FCI+	116	3.268s	0.71	**0.71**	0.40	**0.59**	**0.42**	0.52
50	2000	200	GFCI	111	11.328s	**0.98**	**0.78**	**0.96**	**0.70**	**0.84**	**0.81**
			RFCI	111	0.678s	0.95	0.50	0.55	0.40	0.64	0.18
			uRFCI	111	2.439s	0.95	0.49	0.55	0.40	0.64	0.18
			FCI+	111	4.136s	0.95	0.50	0.55	0.40	0.64	0.18
50	2000	2000	GFCI	111	15.273s	**1.00**	**0.88**	**0.98**	**0.82**	**0.92**	**0.93**
			RFCI	111	3.14s	0.96	0.79	0.59	0.73	0.62	0.47
			uRFCI	111	5.877s	0.97	0.78	0.60	0.72	0.62	0.47
			FCI+	111	8.856	0.96	0.79	0.59	0.73	0.62	0.47
200	2000	200	GFCI	104	8.217s	**0.97**	**0.42**	**0.94**	**0.28**	**0.70**	**0.62**
			RFCI	104	0.466s	0.94	0.27	0.55	0.18	0.61	0.13
			uRFCI	104	1.907s	0.94	0.27	0.55	0.17	0.61	0.13
			FCI+	104	2.851s	0.94	0.27	0.55	0.18	0.61	0.13
200	2000	2000	GFCI	104	19.375s	**0.99**	**0.53**	**0.96**	**0.40**	**0.71**	**0.78**
			RFCI	104	2.596s	0.96	0.49	0.61	0.39	0.58	0.34
			uRFCI	104	5.979s	0.96	0.48	0.61	0.38	0.58	0.35
			FCI+	104	7.066s	0.96	0.49	0.61	0.39	0.58	0.35

Table 2. This table lists the average running time and the average precision (prec.) and recall (rec.) for the different accuracy measurements for every algorithm in every parameterization with 1000 variables and $\alpha = 0.01$. The bold values represent the best value in that column for that parameterization.

DAG				o–o		o→		→		↔	
#Latents	#Edges	#Samples	Alg.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
50	1000	200	GFCI	**0.83**	**0.82**	**0.92**	**0.71**	**0.78**	**0.77**	**0.76**	**0.02**
			RFCI	0.61	0.30	0.42	0.22	0.50	0.26	0.02	**0.02**
			uRFCI	0.61	0.31	0.41	0.22	0.50	0.26	0.02	**0.02**
			FCI+	0.61	0.30	0.41	0.22	0.50	0.26	0.02	**0.02**
50	1000	2000	GFCI	**0.98**	**0.96**	**0.99**	**0.90**	**0.86**	**0.96**	**0.74**	**0.33**
			RFCI	0.94	0.35	0.65	0.38	0.60	0.69	0.04	0.07
			uRFCI	0.94	0.36	0.65	0.38	0.60	0.69	0.04	0.07
			FCI+	0.94	0.35	0.65	0.38	0.60	0.69	0.04	0.07
200	1000	200	GFCI	**0.68**	**0.72**	**0.83**	**0.41**	**0.50**	**0.54**	**0.85**	0.01
			RFCI	0.56	0.27	0.38	0.15	0.39	0.17	0.07	**0.05**
			uRFCI	0.55	0.27	0.38	0.15	0.39	0.17	0.07	**0.05**
			FCI+	0.56	0.27	0.38	0.15	0.39	0.17	0.07	**0.05**
200	1000	2000	GFCI	**0.90**	**0.85**	**0.94**	**0.62**	0.41	**0.83**	**0.68**	0.14
			RFCI	0.89	0.25	0.59	0.23	**0.42**	0.52	0.14	**0.17**
			uRFCI	0.89	0.25	0.59	0.23	**0.42**	0.52	0.14	**0.17**
			FCI+	0.89	0.25	0.59	0.23	**0.42**	0.52	0.14	**0.17**
50	2000	200	GFCI	**0.80**	**0.76**	**0.89**	**0.69**	**0.84**	**0.82**	**0.50**	0.01
			RFCI	0.35	0.20	0.41	0.11	0.64	0.18	0.02	**0.04**
			uRFCI	0.35	0.20	0.41	0.11	0.64	0.18	0.02	**0.04**
			FCI+	0.35	0.20	0.41	0.11	0.64	0.19	0.02	**0.04**
50	2000	2000	GFCI	**0.97**	**0.93**	**0.98**	**0.86**	**0.92**	**0.93**	**0.69**	0.16
			RFCI	0.87	0.16	0.74	0.16	0.62	0.47	0.04	**0.18**
			uRFCI	0.86	0.16	0.74	0.16	0.62	0.48	0.04	0.17
			FCI+	0.87	0.16	0.74	0.16	0.62	0.47	0.04	**0.18**
200	2000	200	GFCI	**0.66**	**0.68**	**0.81**	**0.41**	**0.70**	**0.63**	**0.72**	0.01
			RFCI	0.31	0.19	0.38	0.07	0.61	0.13	0.07	**0.03**
			uRFCI	0.31	0.19	0.38	0.07	0.61	0.13	0.07	**0.03**
			FCI+	0.31	0.19	0.38	0.07	0.61	0.13	0.07	**0.03**
200	2000	2000	GFCI	**0.89**	**0.79**	**0.94**	**0.54**	**0.81**	**0.78**	**0.72**	0.08
			RFCI	0.78	0.12	0.69	0.09	0.58	0.35	0.16	**0.14**
			uRFCI	0.77	0.12	0.70	0.09	0.58	0.35	0.15	0.13
			FCI+	0.78	0.12	0.69	0.09	0.58	0.34	0.16	**0.14**

Appendix:
An Introduction to PAGs

Peter Spirtes

The output of the FCI[4] algorithm [Spirtes, 2001] is a partial ancestral graph (PAG), which is a graphical object that represents a set of causal Bayesian networks (CBNs) that cannot be distinguished by the algorithm.[5] Suppose we have a set of cases that were generated by random sampling from some CBN. Under the assumptions that FCI makes, in the large sample limit of the number of cases, the PAG returned by FCI is guaranteed to include the CBN that generated the data.

An example of a PAG is shown in **Error! Reference source not found.**. This PAG represents the pair of CBNs in **Error! Reference source not found.**a and 1b (where measured variables are in boxes and unmeasured variables are in ovals), as well as an infinite number of other CBNs that may have an arbitrarily large set of unmeasured confounders. Despite the fact that there are important differences between the CBNs in **Error! Reference source not found.**a and 1b (e.g., there is an unmeasured confounder of X_1 and X_2 in **Error! Reference source not found.**b but not in **Error! Reference source not found.**a), they share a number of important features in common (e.g., in both CBNs, X_2 is a direct cause of X_6, there is no unmeasured confounder of X_2 and X_6, and X_6 is not a cause[6] of X_2). It can be shown that every CBN that a PAG represents shares certain features in common. The features that all CBNs represented by a PAG share in common can be read off of the output PAG according to the rules described next.

There are 4 kinds of edges that occur in a PAG: $A \rightarrow B$, A o$\rightarrow$ B, A o–o B, and $A \leftrightarrow B$. The edges indicate what the CBNs represented by the PAG have in common. A description of the meaning of each edge in a PAG is given in

Table A3.

---

[4] The results in this document also hold for the FCI+ [Claassen, 2013] and GFCI [Ogarrio, 2016] algorithms; for simplicity, we will just refer to the FCI algorithm in the remainder of the document. The RFCI algorithm [Colombo, 2012] outputs a slight modification of a PAG. The kind of PAG described here is actually a special case of a more general kind of PAG, where here we assume that there is no selection bias [Spirtes, 1999; Zhang, 2008].

[5] In the Gaussian and multinomial cases, the CBNs represented by a PAG cannot be distinguished by any algorithm without further assumptions.

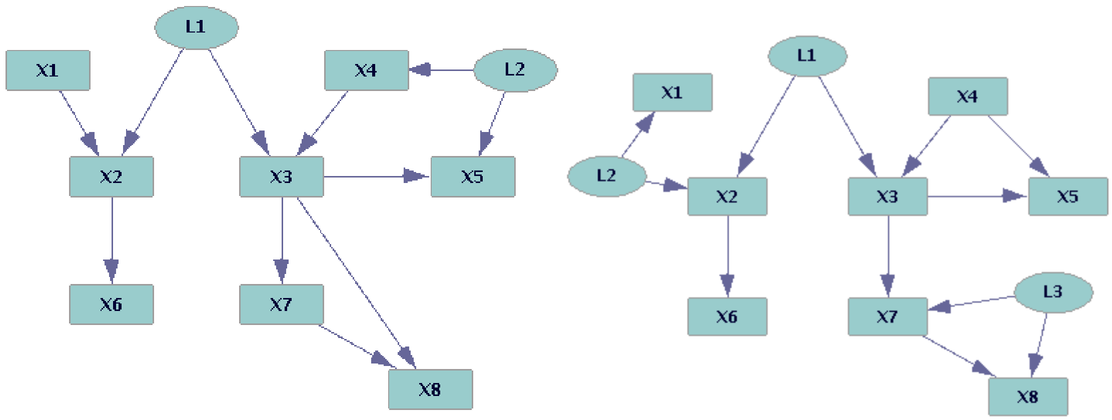
[6] The word "cause" is used in this document to denote a cause that is either a direct or indirect, relative to the measured variables. For example, in Figure 1a, X_1 is a direct cause of X_2 and an indirect cause of X_6.

Table A3: Types of edges in a PAG.

Edge type	**Relationships that are present**	**Relationships that are absent**
$A \rightarrow B$	*A* is a cause of *B*. It may be a direct or indirect cause that may include other measured variables. Also, there may be an unmeasured confounder of *A* and *B*.	*B* is not a cause of *A.*
$A \leftrightarrow B$	There is an unmeasured variable (call it *L*) that is a cause of *A* and *B*. There may be measured variables along the causal pathway from *L* to *A* or from *L* to *B*.	*A* is not a cause of *B*. *B* is not a cause of *A*.
A o→ *B*	Either *A* is a cause of *B*, or there is an unmeasured variable that is a cause of *A* and *B*, or both.	*B* is not a cause of *A.*
A o–o *B*	Exactly one of the following holds: (a) *A* is a cause of *B,* or (b) *B* is a cause of *A,* or (c) there is an unmeasured variable that is a cause of *A* and *B,* or (d) both a and c, or (e) both b and c.	

Table A1 is sufficient to understand the basic meaning of edge types in PAGs. Nonetheless, it can be helpful to know the following additional perspective on the information encoded by PAGs. Each edge has two endpoints, one on the *A* side, and one on the *B* side. For example $A \rightarrow B$ has a tail at the *A* end, and an arrowhead at the *B* end. Altogether, there are three kinds of edge endpoints: a tail "–", an arrowhead ">", and a "o." Note that some kinds of combinations of endpoints never occur; for example, *A* o– *B* never occurs. As a mnemonic device, the basic meaning of each kind of edge can be derived from three simple rules that explain what the meaning of each kind of endpoint is. A tail "–" at the *A* end of an edge between *A* and *B* means "*A* is a cause of *B*"; an arrowhead ">" at the *A* end of an edge between *A* and *B* means "*A* is *not* a cause of *B*"; and a circle "o" at the *A* end of an edge between *A* and *B* means "can't tell whether or not *A* is a cause of *B*". For example $A \rightarrow B$ means that *A* is a cause of *B*, and that *B* is not a cause of *A* in all of the CBNs represented by the PAG.

The PAG in **Error! Reference source not found.** shows examples of each type of edge, and the CBNs in **Error! Reference source not found.** show some examples of what kinds of CBNs can be represented by that PAG.



(a) (b)

Figure 1. Two CBNs that FCI (as well as FCI+, GFCI, and RFCI) cannot distinguish.

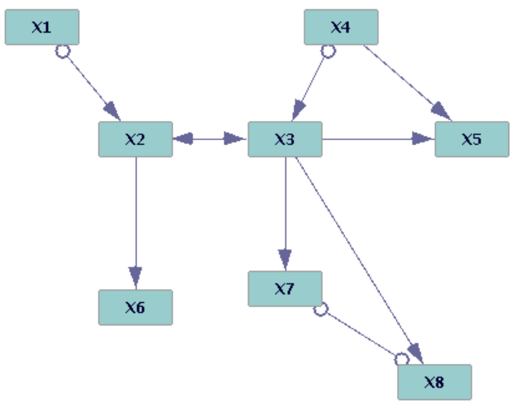


Figure 2. The PAG that represents the CBNs in both Figures 1a and 1b.

References

CCD-FGES (2016). Fast greedy search (FGES) algorithm for continuous variables. Available at: http://www.ccd.pitt.edu/wiki/index.php?title=Fast_Greedy_Search_(FGES)_Algorithm_for_Continuous_Variables

Claasen T, Mooij JM, Heskes T (2013). Learning sparse causal models is not NP-hard. *Uncertainty In Artificial Intelligence*, 29, 172-181.

Colombo D, Maathuis MH, Kalisch M, Richardson TS (2012). Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics,* 40, 294-321.

Lagani V, Triantafillou S, Ball G, Tegner J, Tsamardinos I (2016). Probabilistic computational causal discovery for systems biology. *Uncertainty in Biology*, 17, 33-73.

http://www.mensxmachina.org/files/publications/Probabilistic%20Causal%20Discovery%20for%20Systems%20Biology_prePrint.pdf

Neapolitan RE (2003). *Learning Bayesian Networks* (Pearson).

Ogarrio JM, Spirtes P, Ramsey J (2016). A hybrid causal search algorithm for latent variable models. *JMLR Workshop and Conference Proceedings*, 52, 368-379. http://www.jmlr.org/proceedings/papers/v52/ogarrio16.pdf

Pearl J, Glymour M, Jewell NP (2016). *Causal Inference in Statistics – A Primer* (John Wiley & Sons). https://books.google.com/books?hl=en&lr=&id=IqCECwAAQBAJ&oi=fnd&pg=PT1&dq=Causal+Inference+in+Statistics&ots=NPpnh1N4lC&sig=-CyGyDAsTQP1vFstnAZh3dt-Ih8#v=onepage&q=Causal%20Inference%20in%20Statistics&f=false

Ramsey J (2015). Scaling up greedy equivalence search for continuous variables. http://arxiv.org/ftp/arxiv/papers/1507/1507.07749.pdf

Spirtes P, Glymour C, Scheines R (1993). Causation, Prediction, and Search (Springer). https://books.google.com/books/about/Causation_Prediction_and_Search.html?id=oUjxBwAAQBAJ&printsec=frontcover&source=kp_read_button&hl=en#v=onepage&q&f=false

Spirtes P, Glymour C, Scheines R (2001). *Causation, Prediction, and Search* (MIT Press).

Spirtes P, Richardson T, Meek C (1999). Causal discovery in the presence of latent variables and selection bias. In G Cooper & C Glymour (Eds.) *Computation, Causality, and Discovery*, pp. 211-252 (AAAI Press).

Spirtes P (2010). Introduction to causal inference. *Journal of Machine Learning Research,* 11, 1643-1662. http://www.jmlr.org/papers/volume11/spirtes10a/spirtes10a.pdf

Tetrad system (2016). http://www.phil.cmu.edu/tetrad/current.html

Zhang J. (2008). On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias”. *Artificial Intelligence*, *Volume 172, Issues 16–17,* 1873-1896.