

Discovering Causal Graphs with Cycles and Latent Confounders: An Exact Branch-and-Bound Approach

Kari Rantanen, Antti Hyttinen, Matti Järvisalo¹

*Helsinki Institute for Information Technology HIIT,
Department of Computer Science,
University of Helsinki, Finland*

Abstract

Understanding causal relationships is a central challenge in many research endeavours. Recent research has shown the importance of accounting for feedback (cycles) and latent confounding variables, as they are prominently present in many data analysis settings. However, allowing for cycles and latent confounders makes the structure learning task especially challenging. The constraint-based approach is able to learn causal graphs even over such general search spaces, but to obtain high accuracy, the conflicting (in)dependence information in sample data need to be resolved optimally. In this work, we develop a new practical algorithmic approach to solve this computationally challenging combinatorial optimization problem. While recent advances in exact algorithmic approaches for constraint-based causal discovery build upon off-the-shelf declarative optimization solvers, we propose a first specialized branch-and-bound style exact search algorithm. Our problem-oriented approach enables directly incorporating domain knowledge for developing a wider range of specialized search techniques for the problem, including problem-specific propagators and reasoning rules, and branching heuristics together with linear programming based bounding techniques, as well as directly incorporating different constraints on the search space, such as sparsity and acyclicity constraints. We empirically evaluate our implementation of the approach, showing that it outperforms current state of art in exact constraint-based causal discovery on real-world instances.

Keywords: Graphical models; structure learning; causal discovery; branch and bound; optimization.

Email addresses: kari.rantanen@helsinki.fi (Kari Rantanen),
antti.hyttinen@helsinki.fi (Antti Hyttinen), matti.jarvisalo@helsinki.fi (Matti Järvisalo)

¹Corresponding author

1 Introduction

Discovering causal relations from sample data when allowing for latent confounding variables and feedback (that is, cycles) is a very challenging task in the field of graphical models and structure discovery. Although many features of causal structures can in principle be determined even from passive observation (Pearl, 2000; Spirtes et al., 2000), determining which structural features can be identified from finite sample data has proven difficult.

For general search spaces (allowing latent confounders and/or cycles), the constraint-based causal discovery approach is still applicable (Spirtes et al., 2000; Pearl, 2000). Constraint-based learning algorithms combine (in)dependence constraints from statistical tests to find determined features of the underlying causal graph structure. However, most of such approaches, including the classical PC, CCD and FCI algorithms, scale up in terms of number of variables by selecting independence tests based on earlier test results (Spirtes et al., 2000; Richardson, 1996a). Such greedy strategies can lead to non-optimal accuracy in practice, as early mistakes in independence testing guide search towards inaccurate solutions (Claassen and Heskes, 2012; Hyttinen et al., 2014).

On the other hand, for restricted settings without latent confounders and cycles, that is, for Bayesian networks, exact score-based structure discovery algorithms have been developed (Yuan and Malone, 2013; Bartlett and Cussens, 2017; van Beek and Hoffmann, 2015). A central motivation in developing efficient exact algorithms is that they output a guaranteed optimal solution without making compromises or approximations in their computation. Such *provably globally optimal graphs* have been shown to exhibit better accuracy (Malone et al., 2015). However, much less progress has been made for exact discovery algorithms for more general search spaces that allow for latent confounders and cycles.

In the context of constraint-based discovery, it has been shown that better accuracy can be obtained when a predetermined, large set of tests are conducted before the actual search, and then, conflicting test results are resolved in an optimal way via exact methods (Hyttinen et al., 2014; Magliacane et al., 2016; Borboudakis and Tsamardinos, 2016). However, the general search space with latent confounders and cycles induces a combinatorial optimization problem over a drastically larger search space compared to more restricted settings such as Bayesian network structures (DAGs). Furthermore, the objective functions considered are computationally more complicated to evaluate. Thus improvements to (exact) algorithms for the more general search spaces in terms of running time performance and scalability without trading off accuracy is a major challenge.

In this work, we take on the challenge of improving the scalability of practical exact algorithms for the general search space of causal graph allowing for latent confounding variables and cycles. Recently, there has been noticeable interest in developing algorithmic solutions to this general problem setting and its variants (Triantafillou et al., 2010; Triantafilou et al., 2010; Hyttinen et al., 2013, 2014; Magliacane et al., 2016; Borboudakis and Tsamardinos, 2016; Zhalama et al., 2017; Hyttinen et al., 2017a). The first exact approach to the problem

46 we focus on here was proposed in (Hyttinen et al., 2014), based on declaratively
47 encoding the underlying optimization task as answer set programming (ASP)
48 and applying an ASP solver to obtain provably optimal solutions to the prob-
49 lem. This approach was further refined as a maximum satisfiability (MaxSAT)
50 based approach in (Hyttinen et al., 2017b), where domain-specific techniques
51 were integrated to the extent possible to a MaxSAT solver, relying on a MaxSAT
52 solver to solve the search problem starting with a declarative encoding of the
53 problem. This resulted in the *Dseptor* system which currently represents the
54 state of the art in terms of running time performance for the problem at hand.
55 All in all, this line of work has so far focused on using declarative solving tech-
56 niques, relying in terms of efficiency on generic off-the-shelf declarative methods
57 such as Boolean satisfiability (SAT) (Biere et al., 2009) solvers and their exten-
58 sions to Boolean optimization. While declarative methods offer flexibility and
59 remove implementation-level burden of developing optimized search algorithms
60 for the underlying combinatorial optimization tasks, in this work we explore the
61 alternative of developing domain-specific search algorithms instead of directly
62 relying on declarative solver to perform the search.

63 In this paper we propose a first specialized branch-and-bound style exact
64 search algorithm for optimal causal graphs, allowing the presence of both cy-
65 cles and latent confounding variables. Our problem-oriented view enables di-
66 rectly incorporating domain knowledge for a wider range of specialized search
67 techniques, including problem-specific propagators, branching heuristics, and
68 bounding techniques, as well as directly incorporating restrictions on the search
69 space, such as sparsity and acyclicity constraints. In particular, we develop a
70 branch-and-bound approach to directly search over the general search space,
71 together with several different performance-improving search techniques. These
72 include (i) a problem-specific branching heuristic, (ii) lower bounding techniques
73 applicable during search based on problem-specific unsatisfiable cores and lin-
74 ear programming relaxations, (iii) optimized algorithms for evaluating the ob-
75 jective function of the problem—over exponentially many independence and
76 dependence constraints—during search under partial solutions, and (iv) infer-
77 ence rules—with correctness proofs—for detecting which edges are irrelevant
78 in terms of d-connectivity under a current partial solution. We provide an
79 open-source implementation *because* of the approach, and empirically evaluate
80 its performance on problem instances obtained from real-world datasets from
81 several perspectives: (i) the marginal contribution of the different proposed
82 search techniques, (ii) the impact of the scoring function used for obtaining
83 constraint weights on the efficiency of the approach, and (iii) the efficiency of
84 the approach with respect to current state of the art. In particular, we show
85 that the proposed approach compares favourably with current state of the art
86 in exact constraint-based causal discovery on real-world data sets with respect
87 to running time performance.

88 This article considerably extends a preliminary version published at the
89 PGM 2018 conference (Rantanen et al., 2018). In particular, in this article
90 we describe more effective, earlier unpublished techniques for efficient evalua-
91 tion of the objective function and formalize further inference rules which allow

92 for disregarding undecided edges under partial solutions during search, thereby
93 further speeding up the overall search for an optimal causal graph. We have
94 now implemented these new techniques in a new release version of the *bcause*
95 system. Empirical results presented here have been obtained using this new ver-
96 sion; compared to the version presented at PGM 2018, the additional techniques
97 presented in this article have resulted in non-negligible running time improvements
98 (obtaining up to 10x speed-up and 2x average speed-up) over the version of the
99 system presented at PGM 2018. We have also considerably extended the em-
100 pirical evaluation of the approach with earlier unpublished results: we present
101 empirical data on the marginal contributions of the various search techniques
102 implemented in *bcause* to the overall efficiency of the approach in practice, as
103 well as a running time comparison with the earlier state-of-the-art *Dseptor* sys-
104 tem (Hytinen et al., 2017b). In addition to these new technical contributions,
105 we have considerably extended the discussion and included various examples for
106 improved readability and self-containment.

107 The rest of this article is organized as follows. We begin by detailing the nec-
108 essary background on causal discovery, including causal graphs with latent vari-
109 ables and cycles, the combinatorial optimization task of finding optimal causal
110 graph, and approaches for obtaining well-defined objective function coefficients
111 in terms of weights on the independence and dependence constraints (Section 2).
112 We then continue with detailing the proposed branch-and-bound approach to
113 optimal causal graphs and several efficiency-improving search techniques for the
114 approach (Section 3). We present results from an extensive empirical evaluation
115 of the approach in Section 4). Before conclusions, we discuss the connections of
116 our contributions to related work (Section 5).

117 2. Constraint-based Causal Discovery

118 In this section we give necessary background on causal graphs and the exact
119 problem definition for the structure discovery task we consider in this work.

120 2.1. Causal Graphs

121 Causal structure can be represented by directed graphs where directed edges
122 denote causal relations and nodes correspond to random variables for different
123 measurements (Pearl, 2000; Spirtes et al., 2000). Although graphs are sometimes
124 restricted to be acyclic, here we allow for directed cycles to be able to represent
125 feedback (Spirtes, 1995; Richardson, 1996a,b).

126 In most analysis situations, we are not able to observe all relevant variables
127 or all background factors. Fortunately, the use of bi-directed edges allow for
128 a canonical representation of causal structures as a graph over the observed
129 variables (Pearl, 2000; Spirtes et al., 2000). A bidirected edge $X \leftrightarrow Z$ represents
130 a *latent confounder*, e.g. structure $X \leftarrow L \rightarrow Z$, where L is an unmeasured
131 common cause of two observed variables X and Z . This prompts us to use the
132 following graphs to represent causal structures.

133 **Definition 1** (Causal graphs). *A causal graph is a pair $G = (\mathcal{V}, E)$ with set of*
 134 *nodes \mathcal{V} , where the edge relation $E = E_{\rightarrow} \cup E_{\leftrightarrow}$ is composed of directed edges*
 135 *$E_{\rightarrow} \subseteq \mathcal{V} \times \mathcal{V}$ and (symmetric) bi-directed edges $E_{\leftrightarrow} \subseteq \{\{X, Y\} : X, Y \in \mathcal{V}\}$.*

136 The class of causal graphs is denoted by \mathcal{G} . Note that when the directed edges
 137 are not allowed to form cycles, causal graphs are semi-Markovian graphs (Pearl,
 138 2000). Importantly, in both cases causal graphs are closed under marginaliza-
 139 tion.

140 The central reachability criterion for causal graphs is the following d-separation
 141 (Pearl, 2000). We follow here the definition of Studený (1998) which has been
 142 shown to be equivalent to Pearl’s standard definition.

143 **Definition 2** (d-separation). *Two nodes X and Y in a causal graph $G = (\mathcal{V}, E)$*
 144 *are d-connected given a conditioning set $C \subseteq \mathcal{V} \setminus \{X, Y\}$ if there is at least one*
 145 *d-connecting walk between them; otherwise they are d-separated. A walk is a*
 146 *sequence of edges in the graph (allowing for repeated edges and nodes). A node*
 147 *is a collider on a walk if both its adjacent edges on the walk have an arrow*
 148 *head into the node. A walk is d-connecting given a conditioning set C if every*
 149 *collider on the walk is in C and no other nodes on the walk are in C .*

150 **Example 1.** *The causal structure in Figure 1 a) with unobserved L can be*
 151 *canonically represented by the causal graph in Figure 1 b). In the structure of*
 152 *Figure 1 a), X and W are d-connected given Y by $X \leftarrow L \rightarrow Z \rightarrow Y \leftarrow Z \leftarrow$
 153 $T \leftarrow W$. In the corresponding canonical representation in Figure 1 b), X and
 154 W are d-connected given Y by $X \leftrightarrow Z \rightarrow Y \leftarrow Z \leftarrow T \leftarrow W$. In the causal
 155 graph in Figure 1 b) nodes Y and Q are d-separated given W as all walks between
 156 violate the d-connection criterion at node X .*

157 Self-loops $X \rightarrow X$ do not affect d-connectivity of the graph: for any d-
 158 connecting walk through $X \rightarrow X$ there is a shorter walk that skips the arc $X \rightarrow$
 159 X . Thus, self-loops are inherently unidentifiable here; they are unidentifiable
 160 also in other settings (Lacerda et al., 2008; Hyttinen et al., 2012). Without
 161 loss of generality, we do not consider self-loops through the rest of this article.
 162 Similarly, without loss of generality we do not consider arcs $X \leftrightarrow X$. We
 163 emphasize that any $X \rightarrow X$ or $X \leftrightarrow X$ may be present in the true structure
 164 regardless of the result of the algorithmic approach developed in this article.

165 2.2. Statistical Dependence & Reachability in Graphs

166 Under the commonly used causal Markov assumption (Spirtes et al., 2000),
 167 d-separation in the true *acyclic* structure implies statistical independence in the
 168 generated distribution.

169 A similar result on cyclic causal graphs applies under the following assump-
 170 tions. The parametric models to cyclic graphs are non-recursive structural equa-
 171 tion models (SEMs) (Wright, 1934; Bollen, 1989; Richardson, 1996b). We make
 172 the standard assumption that each data sample is obtained at the unique solu-
 173 tion to the structural equations (given the external disturbances). When
 174 the structural equations are linear, d-separation implies independence (Spirtes,

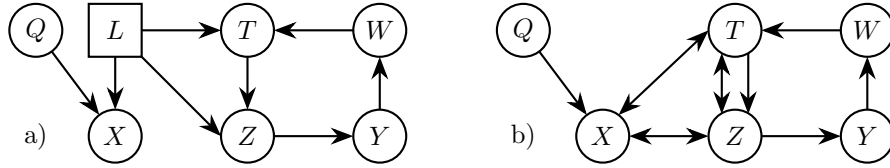


Figure 1: Example graphs: a) a causal graph with an unobserved node L , b) the canonical representation of a) using bidirected edges.

175 1995). The same result applies for discrete random variables, when the struc-
 176 tural equations to every ancestral subset (a set of nodes and their ancestors)
 177 has a unique solution (Forré and Mooij (2017): Theorem 3.8.12 on page 112,
 178 see also Pearl and Dechter (1996), Neal (2000)).

179 Under the commonly used faithfulness assumption (Spirtes et al., 2000),
 180 statistical dependence becomes equivalent to (a type of) reachability in the
 181 graph: two random variables are conditionally dependent given a set of variables
 182 C if and only if they are d-connected given C in the generating causal structure
 183 G . In the rest of the article we use $X \perp\!\!\!\perp Y|C$ ($X \not\perp\!\!\!\perp Y|C$) to denote statistical
 184 independence (dependence) and d-separation (d-connection).

185 **Example 2.** *Given enough samples from a causal model with the structure in*
 186 *Figure 1 b) (or a)), we would expect to find X statistically dependent on W*
 187 *given Y , and Y statistically independent of Q given W .*

188 2.3. Problem Definition

189 In constraint-based causal discovery, the aim is to find an equivalence class
 190 of graphs whose d-separation and d-connection properties respectively match the
 191 statistical independence and dependence relations in the data. The (in)dependence
 192 constraints K are obtained by running statistical independence tests on the
 193 data. Since the tests produce some errors on finite sample data, constraint-
 194 based causal discovery can be viewed as the following abstract optimization
 195 problem (Hyttinen et al., 2014).

Input: A set K of conditional (in)dependence constraints over given set of
 variables \mathcal{V} , and a non-negative weight $w(k)$ for each $k \in K$.

196 **Task:** Find a causal graph $G^* = (\mathcal{V}, E^*)$ such that

$$G^* \in \operatorname{argmin}_{G \in \mathcal{G}} \sum_{k \in K : G \not\models k} w(k). \quad (1)$$

197 In words, our goal is to find a single graph G^* that minimizes the sum of
 198 the weights of the (in)dependence constraints *not* implied ($\not\models$) by G^* . The
 199 weight function $w(\cdot)$ describes the reliability of each constraint (obtained by
 200 independently run tests): conflicts among the constraints are well-resolved when
 201 the sum of the weights of the constraints not satisfied is minimized. Apart
 202 from this constraint satisfaction perspective, Section 2.5 gives a probabilistic
 203 motivation for this objective function.

204 **Example 3.** Let the nodes be $\mathcal{V} = \{X, Y, Z\}$ and let the (in)dependence con-
 205 straints K be as follows (weights in parenthesis):

$$206 \begin{array}{l} X \perp\!\!\!\perp Y \mid Z \quad (1098) \\ X \not\perp\!\!\!\perp Z \mid Y \quad (101804) \end{array} \quad \left| \quad \begin{array}{l} Y \perp\!\!\!\perp Z \mid X \quad (97) \\ Y \not\perp\!\!\!\perp Z \quad (4935) \end{array} \right| \quad \begin{array}{l} X \not\perp\!\!\!\perp Z \quad (106837) \\ X \not\perp\!\!\!\perp Y \quad (3935) \end{array}$$

207 This includes all relations testable in passively observed data over three variables.

208 As the example shows we include in K only one constraint for nodes $\{X, Y\}$
 209 and set $C \subseteq \mathcal{V} \setminus \{X, Y\}$, either in the form of an independence $X \perp\!\!\!\perp Y \mid C$ or a
 210 dependence $X \not\perp\!\!\!\perp Y \mid C$. Several constraints for the same $\{X, Y\}$ and C can be
 211 compressed to a single constraint just by summing up the weights appropriately.

212 The score function trivially satisfies *score equivalence* (Heckerman et al.,
 213 1995): all Markov equivalent structures imply the same d-separations and there-
 214 fore obtain the exact same score regardless of the weight is used (for a fixed set
 215 constraints K). Thus, an optimal causal graph G^* is a representative of the
 216 (Markov) equivalence class closest to the input constraints.

217 Solving this problem exactly has the following consistency result (Hyttinen
 218 et al., 2013, 2014). Under the assumptions discussed in Section 2.2 we have that
 219 statistical independence is equivalent to d-separation. When the weights are
 220 obtained by a test that consistently detects statistical dependence, we have that
 221 in the infinite sample limit, K includes independence and dependence relations
 222 that correspond respectively to d-separation and d-connection relations in the
 223 true graph. Consequently, the optimal solution will be in the equivalence class
 224 of the true graph and satisfy all constraints in K .

225 Completeness depends on what set K is used (Hyttinen et al., 2013). If
 226 not all testable relations are in K , there may be information in the additional
 227 relations in the data that allow for further identification of structural features. In
 228 this article we include in K all $\binom{n}{2}2^{n-2}$ relations testable in passively observed
 229 data of n variables. Therefore a structural feature determined by relations
 230 testable in the data will be uniquely determined in the equivalence class of the
 231 top scoring causal graph. The properties of the equivalence class can be studied
 232 for example with the SAT-based procedure of Hyttinen et al. (2013) or in the
 233 acyclic case by FCI (Spirtes et al., 2000).

234 2.4. Weights for Independence Constraints

235 The algorithmic approach developed in this article is agnostic in terms of
 236 how weights are obtained. One way to obtain weights is through Bayesian
 237 model selection (Cooper, 1997; Steck and Jaakkola, 2002; Abellán et al., 2006;
 238 Margaritis and Bromberg, 2009; Hyttinen et al., 2014). For each independence
 239 statement $X \perp\!\!\!\perp Y \mid C$, consider two models

$$\begin{aligned} M_{\perp} &: P(X, Y \mid C) = P(X \mid C)P(Y \mid C) \\ M_{\not\perp} &: P(X, Y \mid C) = P(X \mid C)P(Y \mid X, C) \end{aligned}$$

where the first postulates independence, and the second postulates dependence.
 Given data D on X, Y, C and a prior probability of independence $P(M_{\perp}) = \alpha$

the probability associated with $k = X \perp\!\!\!\perp Y|C$ simplifies to

$$P(k|D) = \frac{P(Y|C)\alpha}{P(Y|C)\alpha + P(Y|X,C)(1-\alpha)}.$$

240 The marginal likelihoods $P(Y|C)$ and $P(Y|X,C)$ correspond directly the local
 241 scores in the score-based Bayesian network structure learning framework, which
 242 can be evaluated in closed form for categorical variables when using a Dirichlet
 243 prior (Buntine, 1991; Cooper and Herskovits, 1992) and for continuous vari-
 244 ables with linear relations and Gaussian disturbances using an inverse Wishart
 245 Gaussian prior (Geiger and Heckerman, 2002). Note that since both scores are
 246 score-equivalent, the same probabilities are obtained if $M_{\mathcal{X}}$ uses factorization
 247 $P(X, Y|C) = P(X|Y, C)P(Y|C)$ instead.

Since we optimize the sum of violated constraints, for nodes X, Y and set C we include k (independence or dependence relation) that obtained the higher probability with weight obtained by the following log transformation:

$$w(k) = \log P(k|D) - \log P(\neg k|D) \quad (2)$$

248 There are also several alternative ways of obtaining weights that can be
 249 directly used by our procedure. Jabbari et al. (2017) use similar Bayesian model
 250 selection, but dependence is modeled by $P(Z|C)$ where Z is a random variable
 251 whose values are a Cartesian product of the values for X and Y . Natori et al.
 252 (2017) study the use of different priors. Also BIC approximations can be utilized
 253 (Hyttinen et al., 2017a). The approach of Claassen and Heskes (2012) obtains
 254 probabilities for d-separation relations by Bayesian model averaging over graphs.
 255 Triantafilou et al. (2010); Magliacane et al. (2016) employ frequentist statistical
 256 hypothesis testing to obtain similar reliability weights.

257 2.5. Motivation for the Objective Function

258 Apart from a constraint satisfaction perspective, the objective function in
 259 Equation 1 can be given a probabilistic motivation (Hyttinen et al. (2014):
 260 Appendix B, Jabbari et al. (2017): Section 4). The posterior probability of a
 261 graph G given data D can be written as

$$P(G|D) = \sum_{K_i \in \mathcal{K}} P(G|K_i, D)P(K_i|D),$$

262 where \mathcal{K} includes all sets of (in)dependence constraints that can be obtained
 263 from the data.

264 The standard assumption underlying constraint-based causal discovery is
 265 that the (in)dependence constraints exhaust all information on the causal graph
 266 in the data, $G \perp\!\!\!\perp D|K_i$ (Jabbari et al., 2017; Hyttinen et al., 2014):

$$P(G|D) = \sum_{K_i} P(G|K_i)P(K_i|D).$$

267 Another standardly made assumption is that constraints are distributed in-
 268 dependently given the data D (Claassen and Heskes, 2012; Hyttinen et al., 2014;
 269 Triantafilou et al., 2010; Magliacane et al., 2016; Jabbari et al., 2017):

$$P(G|D) = \sum_{K_i} P(G|K_i) \prod_{k \in K_i} P(k|D).$$

270 Note that this is different from only assuming mutual independence of con-
 271 straints *unconditional on the data*.²

272 The term $P(G|K_i)$ is non-zero only for the constraints $K_i = K_G$ implied by
 273 G . Since an independence constraint in K_G may correspond to a dependence in
 274 K and vice versa, we further have

$$P(G|D) = \prod_{k \in K_G} P(k|D) = \prod_{k \in K : G \models k} P(k|D) \prod_{k \in K : G \not\models k} P(\neg k|D).$$

275 For finding the optimal G , we can take the logarithm to obtain

$$\log P(G|D) = \sum_{k \in K : G \models k} \log P(k|D) + \sum_{k \in K : G \not\models k} \log P(\neg k|D),$$

and subtract term $\sum_{k \in K} \log P(k|D)$ that is constant with respect to G , obtain-
 ing

$$\log P(G|D) - \sum_{k \in K} \log P(k|D) = \sum_{k \in K : G \not\models k} [\log P(\neg k|D) - \log P(k|D)] = - \sum_{k \in K : G \not\models k} w(k),$$

276 where $w(k)$ is defined as in Equation 2. Thus, under these modeling assump-
 277 tions, maximizing posterior probability of a graph given the data $P(G|D)$ is
 278 equivalent to minimizing the objective in Equation 1.

279 3. Branch and Bound for Constraint-based Causal Discovery

280 In this section we describe a first specialized branch-and-bound approach
 281 to finding optimal causal graphs. After an overview we give details on an effi-
 282 cient method for determining the satisfied/violated (in)dependence constraints
 283 in each search tree branch (Section 3.2), an effective domain-specific branching
 284 heuristic (Section 3.4), and how to obtain tight bounds during search using lin-
 285 ear programming relaxations (Section 3.5). Furthermore, we describe how struc-
 286 tural restrictions on the search space, such as enforcing acyclicity and degree
 287 restrictions (Section 3.6), can be integrated. Before detailing these techniques,
 288 we start with an overview of the core branch-and-bound routine.

² Jabbari et al. (2017) use a sampling-based approach to account for dependencies among input constraint in an inexact approach. In their simulations the solutions closely corresponded to the solutions assuming independent constraints given the data.

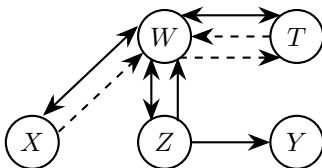


Figure 2: A partial solution; solid edges have been decided present, dashed edges remain undecided, others are decided absent.

289 *3.1. Overview*

290 The overall structure of the branch-and-bound search is presented as Al-
 291 gorithm 1. A The algorithm performs a complete depth-first search over the
 292 causal graphs within the well-known general algorithmic framework of branch
 293 and bound, extending partial solutions towards fully defined causal graphs, and
 294 using bounding techniques for pruning out partial solutions which can be deter-
 295 mined not to improve the current best solution.

296 In this context, a *partial solution* G is a graph in which each edge is either
 297 decided *absent*, decided *present* or *undecided*.

298 **Example 4.** Consider the illustration of a partial solution in Figure 2. The
 299 solid edges and absent edges have been decided to be present and absent, re-
 300 spectively. The dashed edges represent undecided edges in the partial solution,
 301 meaning that (in case the search branch represented by the partial solution is not
 302 pruned out before this) the search will subsequently traverse over the subsearch
 303 space spanned by the undecided edges.

304 At each search tree node, on Line 2 we compute a lower bound for the
 305 weight of the current partial solution G . If this value is not less than the weight
 306 of the incumbent upper bound solution G^* , we can safely close the current
 307 branch and backtrack. If the current branch cannot be closed, we move on to
 308 Line 3 to select a yet-undetermined edge e^* in G . If no such edges exist, that is,
 309 $e^* = null$, we update the incumbent upper bound solution G^* to G if the current
 310 partial solution has smaller weight. If multiple edge candidates exist, the most
 311 promising one is chosen heuristically (see Section 3.4 for details). On the other
 312 hand, if $e^* \neq null$, i.e., a decidable edge exists, we recursively call Algorithm 1
 313 to open two search tree branches, one where (a) e^* is decided present in G and

Algorithm 1 The core structure of the branch and bound.

```

1: function SEARCH(partial solution  $G$ )
2:   if  $w(G^*) \leq \text{LOWERBOUND}(G)$  then return
3:    $e^* \leftarrow \text{SELECTUNDECIDEDGE}(G)$ 
4:   if  $e^* \neq null$  then
5:     Branch with (a) SEARCH( $G$  with  $e^*$  decided present) and
6:     (b) SEARCH( $G$  with  $e^*$  decided absent) in the preferred order.
7:   else if  $w(G) < w(G^*)$  then  $G^* \leftarrow G$ 

```

314 one where (b) the edge is decided absent. The order in which we visit these
 315 branches is determined heuristically, see Section 3.4 for details. At the end of
 316 the search, G^* is guaranteed to be a solution with globally optimal cost.

317 For computing a simple initial upper bound solution G^* , we first initialize it
 318 as an empty graph, then traverse the dependence constraints $[X \not\perp Y \mid Z] \in K$
 319 in descending weight order (Triantafillou et al., 2010) and add corresponding
 320 edges $X \rightarrow Y$ to the graph as long as this locally improves the weight of G^* . Any
 321 edge addition which would make G^* violate possible search space restrictions
 322 (Section 3.6) is omitted.³

323 We will now provide an example of how our branch-and-bound search would
 324 behave with a simple 3-variable instance.

325 **Example 5.** *Let the nodes be $\mathcal{V} = \{X, Y, Z\}$ and let the (in)dependence con-*
 326 *straints K be as follows (weights in parenthesis):*

$$327 \begin{array}{l} X \perp\!\!\!\perp Y \mid Z \quad (1098) \\ X \not\perp Z \mid Y \quad (101804) \end{array} \quad \left| \quad \begin{array}{l} Y \perp\!\!\!\perp Z \mid X \quad (97) \\ Y \not\perp Z \quad (4935) \end{array} \quad \left| \quad \begin{array}{l} X \not\perp Z \quad (106837) \\ X \not\perp Y \quad (3935) \end{array} \right.$$

328 *Before entering the search itself, we construct the initial upper bound solu-*
 329 *tion. For this purpose we traverse the four dependence constraints in descending*
 330 *weight order and add the edges $X \rightarrow Z$, $Y \rightarrow Z$ and $X \rightarrow Y$ to an empty graph.*
 331 *Each edge addition locally improves the solution’s weight and there are no more*
 332 *variable pairs (with dependence constraints) and as such the resulting graph*
 333 *(shown in Figure 3 (0)) serves as our initial upper bound solution with weight*
 334 *1195.*

335 *We are now ready to enter the branch and bound. The steps 1-5 of the*
 336 *search are illustrated in Figure 3. We start off with a partial solution where*
 337 *all the edges are undecided (Step 1). Assume that we use a branching strategy*
 338 *where we branch first by deciding edges absent between variables that seem most*
 339 *likely to be independent. Concretely, we first decide the edges $X \leftrightarrow Y$, $X \rightarrow Y$*
 340 *and $X \leftarrow Y$ to be absent (Step 2), followed by $Y \leftrightarrow Z$, $Y \rightarrow Z$ and $Y \leftarrow Z$*
 341 *(Step 3). However, after deciding $Y \leftarrow Z$ absent, we obtain a lower bound of*
 342 *8870 for the partial solution, which is larger than our incumbent upper bound of*
 343 *1195. Hence we backtrack, deciding $Y \leftarrow Z$ to be present instead (Step 4).*

344 *We continue the search by deciding $X \leftrightarrow Z$, $X \rightarrow Z$ and $X \leftarrow Z$ to be present*
 345 *(Step 5), as our branching heuristic recognizes that there are no independence*
 346 *constraints between X and Z . Now there are no more edge decisions to be*
 347 *made, and so we evaluate the solution at hand. It violates only the constraint*
 348 *$[Y \perp\!\!\!\perp Z \mid X]$ and thus has weight 97. This is better than our previous solution*
 349 *with weight of 1195, and hence we update our incumbent upper bound to this*
 350 *new one.*

351 *Next we backtrack in the search. For each decision we made in the search*
 352 *tree (except for the presence of $Y \leftarrow Z$), we also have a branch with the opposite*
 353 *decision (i.e., deciding a present edge to be absent, or deciding an absent edge to*

³In practice, however, we have observed empirically that the thereby obtained initial upper bound tends to have only a negligible impact on overall runtime performance of the approach.

354 be present). However, immediately after choosing any of these alternatives, we
 355 obtain a lower bound which closes the corresponding branch. Hence we backtrack
 356 all the way to the root node of the search tree, closing all the branches on the
 357 way, thus determining that the found graph with weight 97 is indeed the optimal
 358 solution.

359 3.2. Efficient Evaluation of the Objective Function

360 Given that there are superpolynomially many (in)dependence constraints
 361 with respect to the number of graph nodes, evaluating the objective function can
 362 be a time-consuming task in itself. In this section we provide ideas for efficient,
 363 incremental tracking of satisfiability for given constraints. There are several
 364 different ways for checking whether (in)dependence constraints are satisfied by
 365 a graph (Studený, 1998; Shachter, 1998). Building on such ideas, here our focus
 366 is to evaluate a large number of constraints incrementally when extending a
 367 branch, and the constraints are evaluated over a partial solution, a graph for
 368 which some edges are decided present and some absent.

369 For a partial solution, each (in)dependence constraint can have exactly one
 370 of three states: *satisfied*, *violated* or *undetermined*. The states are defined in the

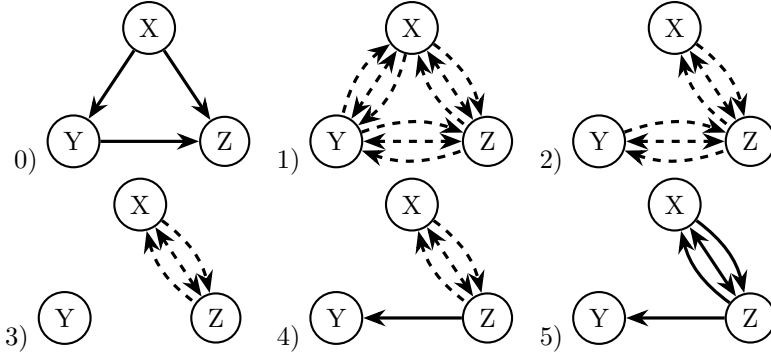


Figure 3: The phases of the example. (0): The initial upper bound solution. (1-5): Gradual construction of a solution in the search.

Algorithm 2 Efficient update of constraint states after an edge decision.

- 1: **function** CHECKCONSTRAINTS(partial solution G , nodes X, Y , edge e)
 - 2: **if** e is present in G **then**
 - 3: $G' \leftarrow \text{minc}(G)$
 - 4: **else**
 - 5: $G' \leftarrow \text{maxc}(G)$ with e
 - 6: **if** e does not affect the d-connectivity of X and Y in G' **then return**
 - 7: $C^+ \leftarrow$ Unavoidable colliders on d-connections between X and Y in G' .
 - 8: $C^- \leftarrow$ Unavoidable non-colliders on d-connections between X and Y .
 - 9: Check constraints of the form $[X \perp\!\!\!\perp Y \mid S]$ (and $[X \not\perp\!\!\!\perp Y \mid S]$)
 - 10: such that $C^+ \subseteq S$ and $S \cap C^- = \emptyset$.
-

371 following way. A *complete solution* or *completion* can be obtained from a partial
 372 solution by deciding the state of all undecided edges. A *maximal completion*
 373 $\text{maxc}(G)$ of a partial solution has all undecided edges marked present (e.g. Fig-
 374 ure 2 *with* the dashed edges), a *minimal completion* $\text{minc}(G)$ has all undecided
 375 edges absent (Figure 2 *without* the dashed edges). An independence constraint
 376 is satisfied if the corresponding d-separation holds in $\text{maxc}(G)$, and violated
 377 if the corresponding d-separation does not hold in $\text{minc}(G)$ (Hyttinen et al.,
 378 2013). A dependence constraint is satisfied if the corresponding d-connection
 379 holds in $\text{minc}(G)$, and violated if the corresponding d-connection does not hold
 380 in $\text{maxc}(G)$. All other constraints are undetermined. In the beginning of the
 381 search, when no edges are decided, the states of all constraints are undetermined.

382 **Example 6.** *Consider the partial solution in Figure 2. The constraint $[X \not\perp$
 383 $Z \mid W]$ is satisfied, since no matter how one decides the undecided edges, the
 384 path $X \rightarrow W \leftarrow Z$ will always exist in the resulting graph. On the other hand,
 385 the constraint $[W \not\perp Y \mid Z]$ is violated, since no matter how one decides the
 386 undecided edges, all the paths between W and Y in the resulting graph will
 387 contain Z as a non-collider. Moreover, the constraint $[Z \perp\!\!\!\perp T]$ is undetermined
 388 since it can be either satisfied or violated depending on whether the undecided
 389 edge $W \rightarrow T$ is decided present or absent.*

390 When a new edge decision is made, we update the states of the input con-
 391 straints with respect to the current partial solution. This also keeps track of
 392 the total weight of the violated constraints and provides a simple lower bound.
 393 Furthermore, the satisfied/violated information can be given to a linear pro-
 394 gramming solver so that stronger, what we call core-based lower bounds (as
 395 detailed in Section 3.5) stay up to date. Note that the choice of how regularly
 396 we update constraint states from undetermined to satisfied/violated does not af-
 397 fect the correctness of the search, as long as all complete solutions are evaluated
 398 exactly.

399 At each search node, we branch on a currently undecided edge to be either
 400 present or absent in the partial solution. When deciding an edge present, as-
 401 suming that acyclicity or an edge degree limit (see Section 3.6) is not enforced,
 402 we only check whether new d-connections are formed in the minimal completion
 403 of the partial solution. When deciding an edge absent, we only check whether
 404 d-connections disappeared from the maximal completion of the partial solution.
 405 For determining whether a d-connection exists between two nodes X and Y
 406 given some conditioning set C , we use a straightforward algorithm that simply
 407 checks whether there is a path between X and Y where all the colliders are in
 408 C and no other nodes are in C .

409 An efficient way to update constraint states for a given node pair after an
 410 edge decision is presented as Algorithm 2. For example, consider a case where
 411 an edge $A \rightarrow B$ is decided present in a partial solution G . To update constraint
 412 states for a node pair (X, Y) , we first check whether there could be a new
 413 d-connection between X and Y given some C in the minimal completion G'
 414 (Line 6). If not, the constraints need not be updated, because deciding an edge
 415 present does not remove existing d-connections, and hence the constraint states

416 between the node pair remain unchanged. Otherwise, if new d-connections may
417 have been formed, we identify a set of unavoidable colliders C^+ and non-colliders
418 C^- between all d-connecting walks between X to Y in G' (Line 7 and 8). We
419 can then omit checking any constraint states for X and Y where conditioning set
420 does not contain all the colliders C^+ or contains some non-colliders C^- . This is
421 because, by the definition of d-separation, d-connections that contradict these
422 collider/non-collider requirements cannot have been formed in the completion.
423 Each item in these sets halves the number of constraints that we need to check for
424 the node pair in question. Intuitively, for any relatively sparse partial solution,
425 there is likely a shared bottleneck for all walks between two nodes.

426 There is no need for the C^+/C^- sets to contain every single unavoidable
427 collider/non-collider, because this information is merely used to speed up the
428 constraint evaluation, and it does not affect the end-result (i.e., the determined
429 states) of the evaluation. Hence we need to make a trade-off between how much
430 time is used to gather C^+/C^- and how much time is saved by having those
431 sets. For this reason we use the straightforward method described in Algo-
432 rithm 3 for gathering only some (i.e., in general not all) of the unavoidable col-
433 liders/noncolliders when traversing from node X to Y in the (minimal/maximal)
434 completion G' of partial solution G . Here the node A is the starting point and
435 V is the set of nodes that have already been visited. We execute this procedure
436 starting from both X and Y ; i.e., $\text{FINDUNAVOIDABLES}(G', X, \emptyset, X, Y)$ and
437 $\text{FINDUNAVOIDABLES}(G', Y, \emptyset, X, Y)$. For a graph with n nodes, each of these
438 procedure calls have a $\mathcal{O}(n^2)$ time complexity and a $\mathcal{O}(n)$ space complexity.

439 Deciding an edge e absent is essentially analogous to the case where the edge
440 is decided present. However, as mentioned in Algorithm 2, in this case we have
441 to make sure that e still exists in the maximal completion G' . This is because

Algorithm 3 A simple method for finding unavoidable colliders and non-colliders between nodes X and Y in a completion G' starting from node $A \in \{X, Y\}$.

```

1: function FINDUNAVOIDABLES(graph  $G'$ , nodes  $A, X, Y$ , set  $V$  of visited
   nodes)
2:    $\mathcal{N} \leftarrow$  neighbours of  $A$  in  $G'$ 
3:   if  $A \notin \{X, Y\}$  then
4:     if for all  $N \in \mathcal{N}$  the edge  $A \rightarrow N$  is not present in  $G'$  then
5:        $C^+ \leftarrow C^+ \cup \{A\}$ 
6:     else if for all  $N \in \mathcal{N} \setminus V$  neither  $A \leftarrow N$  nor  $A \leftrightarrow N$  is present then
7:        $C^- \leftarrow C^- \cup \{A\}$ 
8:   if  $|\mathcal{N} \setminus V| \neq 1$  or  $\mathcal{N} \setminus V \subseteq \{X, Y\}$  then
9:     return
10:   Let  $B \in \mathcal{N} \setminus V$ 
11:   if neither  $A \rightarrow B$  or  $A \leftrightarrow B$  is present in  $G'$  then
12:      $C^- \leftarrow C^- \cup \{B\}$ 
13:   Call FINDUNAVOIDABLES( $G', B, V \cup \{A\}, X, Y$ )

```

442 here we are interested in checking which d-connections get removed from the
 443 completion due to the edge decision, and hence we want use the unavoidable
 444 nodes C^+ and C^- that existed *before* the edge was removed from the comple-
 445 tion. That is, the edge can only have removed d-connections which satisfy these
 446 collider/non-collider requirements.

447 **Example 7.** *For the partial solution in Figure 2, we find that node Z is an*
 448 *unavoidable non-collider in all d-connecting walks between X and Y (given any*
 449 *conditioning set) in the corresponding minimal completion. This tells us that we*
 450 *do not need to check the existence of d-connections between X and Y that have*
 451 *Z in the conditioning set. That is, we must have $X \perp\!\!\!\perp Y \mid Z$; $X \perp\!\!\!\perp Y \mid Z, W$;*
 452 *$X \perp\!\!\!\perp Y \mid Z, T$ and $X \perp\!\!\!\perp Y \mid Z, W, T$.*

453 3.3. Rules for Inferring Irrelevant Edges

454 In this section we provide a way to reduce the amount of time the search has
 455 to spend at each search tree node. Recall that at each step of the search, we
 456 decide an edge e to be either present or absent in the current partial solution
 457 G , and then use the completions $\text{minc}(G)$ and $\text{maxc}(G)$ to determine whether
 458 states of the (in)dependence constraints changed. Here we provide simple rules
 459 for detecting when an edge decision by itself cannot affect the states of (certain)
 460 constraints. We then use this information to avoid performing a considerable
 461 amount of unnecessary d-separation checks which would create unwanted over-
 462 head to the search.

463 The rules are based on so called *inducing paths* (Verma and Pearl, 1990;
 464 Spirtes et al., 2000). To allow for the development of theory relevant to the
 465 current setting, we define inducing paths in this paper as follows.

466 **Definition 3.** *Let X and Y be nodes in a causal graph $G = (\mathcal{V}, E)$. If there*
 467 *exist distinct $V_1, \dots, V_k \in \mathcal{V}$ where $V_1 = X$ and $V_k = Y$ such that*

- 468 • $V_1 \rightarrow V_2$ is present, and
- 469 • both $V_{i-1} \rightarrow V_i$ and $V_{i-1} \leftrightarrow V_i$ is present for each $i > 2$,

470 *then we denote $X \rightsquigarrow Y$.*

471 The following lemma will be useful for arguing about the correctness of the rules
 472 (Lemma 3 of Verma and Pearl (1990)).

473 **Lemma 1.** *Vertices A and B are d-connected in graph $G = (\mathcal{V}, E)$ given any*
 474 *conditioning set $C \subseteq \mathcal{V} \setminus \{A, B\}$ if $A \rightsquigarrow B$ is present in G .*

475 *Proof.* Let $A \rightsquigarrow B$. By the definition, there exists $V_1, \dots, V_k \in \mathcal{V}$ for some $k \geq 2$
 476 such that $V_1 = A, V_k = B$ where $V_1 \rightarrow V_2$ is present, and both $V_{i-1} \rightarrow V_i$ and
 477 $V_{i-1} \leftrightarrow V_i$ is present for each $i > 2$. When $k = 2$, the lemma's claim is trivial.
 478 When $k = 3$, we have the edges $A \rightarrow V_2 \rightarrow B$ and $V_2 \leftrightarrow B$ in G , so A and B
 479 are clearly connected given C regardless whether $V_2 \in C$.

480 Assume that the claim holds for some $k \geq 3$. We show that it holds for
 481 $k + 1$. We have $A \rightsquigarrow V_k \rightarrow B$ and $V_k \leftrightarrow B$ in G , so A and V_k are d-connected



Figure 4: Illustration of of the edge irrelevancy rule 1. The dashed edge corresponds to the edge whose impact to the d-connectivity is to be checked. Note that the rule would trigger even if the graph would contain arbitrary additional nodes and edges.

482 given C . Now, if $V_k \in C$, choose d-connecting path $A \rightsquigarrow V_k \leftrightarrow B$, otherwise
 483 $A \rightsquigarrow V_k \rightarrow B$. Hence A and B are d-connected in G given C . \square

484 We will now formally define four rules which allow for inferring that, given
 485 that particular edges are present/absent in the current partial solutions, the
 486 presence or absence of a currently undecided edge is irrelevant in terms of d-
 487 connectivity.

488 The first rule captures situations where adding an edge would form a ‘short-
 489 cut’ between two nodes that are already connected by an inducing path. Par-
 490 ticularly, if the added edge has the same direction as the inducing path, then
 491 the edge does not increase d-connectivity in the graph since one could always
 492 use the inducing path instead of the shortcut.

493 **Rule 1.** *Let A and B be nodes in a causal graph such that $A \rightsquigarrow B$ is present.*
 494 *Then adding the edge $A \rightarrow B$ does not affect the d-connectivity of the graph.*

495 For an illustration of Rule 1, see Figure 4. In the figure, the dashed edge
 496 corresponds to the edge whose impact to the d-connectivity is to be checked.
 497 Note that the rule would trigger even if the graph would contain arbitrary
 498 additional nodes and edges.

499 *Proof.* (Correctness of Rule 1) Assume that $A \rightsquigarrow B$ exists in G . Consider the
 500 path $A \rightarrow B$ and all the paths via $A \rightsquigarrow B$. In both cases, for each conditioning
 501 set $C \subseteq \mathcal{V} \setminus \{A, B\}$, there exists a path which d-connects A and B in G (by
 502 Lemma 1), and in all the paths (1) an edge is pointing outwards from A , and
 503 (2) an edge is pointing inwards to B . Hence the edge $A \rightsquigarrow B$ does not affect
 504 the d-connectivity in G . \square

505 The second rule captures situations where the added ‘shortcut’ edge between
 506 two nodes is bi-directional. In this case the edge and the inducing path point
 507 differently at one of the nodes and similarly to the other node. Intuitively, the
 508 side of the structure in which the edge and the inducing path agree behaves
 509 similarly to the first rule. The side in which the edge and the inducing path dis-
 510 agree requires us to make sure that the shortcut does not create a new potential
 511 collider.

512 **Rule 2.** *Let A and B be nodes in a causal graph such that $A \rightsquigarrow B$ is present.*
 513 *Adding the edge $A \leftrightarrow B$ does not affect the d-connectivity of nodes X and Y if*

- 514 1. $A = X$ or $A = Y$, or

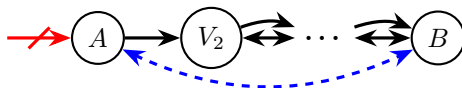


Figure 5: Illustration of of the edge irrelevancy rule 2. The dashed edge corresponds to the edge whose impact to the d-connectivity is to be checked. The crossed-out edge indicates that the target node cannot have incoming edges apart from the dashed one. Note that, apart from the restrictions denoted by the crossed-out edge, the rule would trigger even if the graph would contain arbitrary additional nodes and edges.

515 2. A has no incoming edges besides from B .

516 For an illustration of Rule 2, see Figure 5. The crossed-out edge indicates that
 517 the target node cannot have incoming edges apart from the ones in the figure
 518 that are not crossed out. Note that, apart from the restrictions denoted by
 519 the crossed-out edge, the rule would trigger even if the graph would contain
 520 arbitrary additional nodes and edges.

521 *Proof.* (Correctness of Rule 2) Assume that $A \rightsquigarrow B$ is present in G . Consider the
 522 path $A \leftrightarrow B$ and all the paths via $A \rightsquigarrow B$. In both cases, for each conditioning
 523 set $C \subseteq \mathcal{V} \setminus \{A, B\}$, there exists a path which d-connects A and B in G (by
 524 Lemma 1), and in all the paths an edge is pointing inwards to B in a path. The
 525 key difference is that in the path $A \leftrightarrow B$ there is an edge pointing to A , whereas
 526 on all paths via $A \rightsquigarrow B$ there is no edge pointing to A .

527 Assume first that $A = X$ or $A = Y$. Now, A cannot be a collider or a non-
 528 collider on d-connecting paths via X and Y , and so it does not matter whether
 529 or not an edge points to A . Hence $A \leftrightarrow B$ does not affect the d-connectivity of
 530 X and Y .

531 Assume then that A has no incoming edges besides from B (and $A \neq X, B \neq$
 532 Y). Now, A is a non-collider on all paths from X and Y which go through the
 533 path $A \leftrightarrow B$ or go through any path of $A \rightsquigarrow B$. Hence $A \leftrightarrow B$ does not affect
 534 the d-connectivity of X and Y . \square

535 The third rule captures situations where the added ‘shortcut’ edge between
 536 two nodes points to the opposite direction wrt. the inducing path. Intuitively,
 537 since the edge and the inducing path disagree on both sides of the structure,
 538 this corresponds to two instances of the situation from the second rule where a
 539 node is pointed differently by the edge and the path.

540 **Rule 3.** Let A and B be nodes in causal graph such that $A \rightsquigarrow B$ is present.
 541 Adding the edge $B \leftarrow A$ does not affect the d-connectivity of nodes X and Y if
 542 both of the following hold:

- 543 1. $A = X$ or $A = Y$ or A has no incoming edges besides from B , and
 544 2. $B = X$ or $A = Y$ or B has no incoming edges besides from A .

545 For an illustration of Rule 3, see Figure 6. Again, the crossed-out edges indicate
 546 that the target node cannot have incoming edges apart from the dashed one;

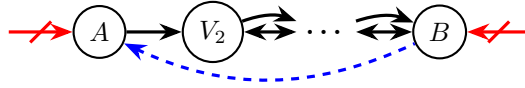


Figure 6: Illustration of of the edge irrelevancy rule 3. The dashed edge corresponds to the edge whose impact to the d-connectivity is to be checked. The crossed-out edges indicate that the target node cannot have incoming edges apart from ones in the figure that are not crossed out. Note that, apart from the restrictions denoted by the crossed-out edges, the rule would trigger even if the graph would contain arbitrary additional nodes and edges.

547 apart from the restrictions denoted by the crossed-out edge, the rule would
 548 trigger even if the graph would contain arbitrary additional nodes and edges.

549 *Proof.* (Correctness of Rule 3) Assume that $A \rightsquigarrow B$ is present in G . Consider the
 550 path $A \leftarrow B$ and all the paths via $A \rightsquigarrow B$. In both cases, for each conditioning
 551 set $C \subseteq \mathcal{V} \setminus \{A, B\}$, there exists a path which d-connects A and B in G (by
 552 Lemma 1). The key differences are that (1) in $A \leftarrow B$ there is an edge pointing
 553 to A whereas in all paths via $A \rightsquigarrow B$ there is not, and (2) in all paths via $A \rightsquigarrow B$
 554 there is an edge pointing to B whereas in $A \leftarrow B$ there is not.

555 By applying similar reasoning as in the proof of Theorem 2, we have that
 556 if $A = X$ or $A = Y$ or A has no incoming edges besides from B , then on all
 557 paths from X to Y which go through the path $A \leftarrow B$ or go through any path
 558 via $A \rightsquigarrow B$, the d-connectivity of X and Y is not affected by whether an edge
 559 points to A . The same holds symmetrically in the case where $B = X$ or $B = Y$
 560 or B has no incoming edges from A . Therefore, when both conditions of the rule
 561 hold, then the d-connectivity of X and Y is not affected by whether there is an
 562 edge pointing to A or B , and hence $A \leftarrow B$ does not affect the d-connectivity
 563 of X and Y in G . \square

564 Similarly as Rule 1, Rules 2 and 3 can be used globally without concerning
 565 what the variables X and Y are. That is, if merely the ‘has no incoming edges’
 566 conditions hold, we can omit checking the (in)dependence constraints for all
 567 variables.

568 Finally, the following fourth rule states that checking constraint states with
 569 respect to a maximal completion is unnecessary unless the completion is sparse
 570 enough.

571 **Rule 4.** *Removing an edge e from a causal graph does not affect the d-connectivity*
 572 *of X and Y if (1) $X \rightsquigarrow Y$, (2) $Y \rightsquigarrow X$ or (3) $X \leftrightarrow Y$ would still hold in the*
 573 *graph after removing e .*

574 *Proof.* (Correctness of Rule 4) When any of the three conditions hold, X and
 575 Y are d-connected given any conditioning set, regardless of e . Note that edges
 576 $X \rightarrow Y$ and $Y \rightarrow X$ are special cases of the first two conditions. \square

577 All four edge relevancy rules introduced in this section are straightforward
 578 to check in $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space, where n is the number of nodes in
 579 the graph. This is because the computationally most demanding part is just to
 580 verify whether an inducing path exists between two nodes in the graph.

581 *3.4. Problem-Specific Branching Heuristics*

582 The branching heuristics applied within the branch and bound are crucial for
 583 the performance of the algorithm. In this section we propose problem-specific
 584 heuristics for our approach.

Let $K^+(X, Y)$ ($K^-(X, Y)$) be the set of undetermined dependence (independence) constraints between nodes (X, Y) by the current partial solution. We will also use

$$K^+(X) = \bigcup_Y K^+(X, Y) \quad \text{and} \quad K^-(X) = \bigcup_Y K^-(X, Y)$$

585 to denote the undetermined dependence and independence constraints involving
 586 node X , respectively. Furthermore, let $w^+(X, Y), w^-(X, Y), w^+(X), w^-(X)$ be
 587 the sum of weights of the constraints in sets $K^+(X, Y), K^-(X, Y), K^+(X), K^-(X)$,
 588 respectively. We use the following rules in order for choosing the next pair for
 589 which an edge to be decided absent or present. Here (X, Y) and (A, B) denote
 590 distinct pairs of nodes.

- 591 1. Choose (X, Y) over (A, B) if all edges are decided between (A, B) or more
 592 edges have been decided present between (A, B) than between (X, Y) .
- 593 2. Choose (X, Y) over (A, B) if $w^-(A, B) \leq w^-(X, Y)$ and $w^-(X, Y) > 0$.
3. Choose (X, Y) over (A, B) if

$$w^+(X) + w^+(Y) + \max_{k \in K^+(X, Y)} w(k) \geq w^+(A) + w^+(B) + \max_{k \in K^+(A, B)} w(k).$$

594 The first rule captures our preference of setting edge decisions throughout
 595 the entire graph instead of deciding all edges between a single pair of nodes
 596 immediately. The second rule captures the preference for edges absences when
 597 the involved nodes are found independent given one or many conditioning sets.
 598 Deciding these absences of edges early via the heuristic directs the search to-
 599 wards sparser solutions for which d-connection checks are faster to evaluate.
 600 This relates to previous literature: PC algorithm decides the absence of an edge
 601 between X, Y upon finding a single conditioning set given which the nodes are
 602 independent (Spirtes et al., 2000). Thus, a problem-specific greedy (and often
 603 unreliable) strategy can act as a good heuristic in exact search. Finally via
 604 the third rule we prefer satisfying strong dependencies with large weights using
 605 direct connections.

606 For a graph with n nodes, it takes $\mathcal{O}(n^2)$ time and space to select the
 607 next node pair to branch with, assuming that the compared values ($w^+(\cdot)$,
 608 $\max_{k \in K^+(\cdot)} w(k)$ etc.) have been precomputed into cache. Indeed, we gather
 609 this information while evaluating the constraint states (Section 3.2), resulting
 610 in no additional complexity.

After the best node pair (X, Y) is chosen out of the possible options, we branch in the search by deciding an arbitrary yet-undecided edge between the nodes $(X \rightarrow Y, X \leftarrow Y$ or $X \leftrightarrow Y)$. We branch by deciding the edge absent first if and only if

$$[X \perp\!\!\!\perp Y \mid Z] \in K \quad \text{for any} \quad Z \subseteq \mathcal{V} \setminus \{X, Y\}.$$

- | | |
|-------|---|
| (i) | $\{X \perp Z S; X \perp Y S; X \not\perp Z Y, S\}$ |
| (ii) | $\{X \not\perp Z S; Y \not\perp Z S; X \perp Y S; X \perp Y Z, S\}$ |
| (iii) | $\{X \not\perp Z Y, S; Y \not\perp Z X, S; X \perp Y S; X \perp Y Z, S\}$ |
| (iv) | $\{Y \not\perp Z S; X \not\perp Z S; Z \perp W X, Y, S; X \perp Y Z, S; X \perp Y W, S\}$ |
| (v) | $\{Y \not\perp Z S; X \not\perp Z S; Z \perp W Y, S; X \perp Y Z, S; X \perp Y W, S\}$ |
| (vi) | $\{X \not\perp Y Z, S; Y \not\perp Z X, W, S; W \not\perp Y Z, S;$
$W \perp X Y, Z, S; X \perp Z W, S\}$ |
| (vii) | $\{X \not\perp Y Z, S; Y \not\perp Z X, W, S; W \not\perp Y S; W \perp X Y, S; X \perp Z W, S\}$ |

Figure 7: Small minimal unsatisfiable core patterns used for computing lower bounds via linear programming.

611 *3.5. Computing Tight Lower Bounds by Linear Programming*

612 We now describe how we compute strong lower bounds using core pat-
613 terns (Hyttinen et al., 2017b). An *unsatisfiable core* is a set of (in)dependence
614 constraints that cannot be simultaneously satisfied by any graph in \mathcal{G} . Some
615 example cores are marked by rectangles in Figure 8. We use the seven core
616 patterns from (Hyttinen et al., 2017b), shown in Figure 7, to find cores for the
617 input dataset in the beginning of the search. Using these, we can compute lower
618 bounds by formulating a minimum-cost hitting set problem⁴ where the unsatis-
619 fiable cores represent the sets and the (in)dependence constraints represent the
620 elements. The objective is then to find a minimum-cost subset of constraints
621 that contains a constraint from each core. To obtain the bounds in practice,
622 similarly as in Hyttinen et al. (2017b), we solve linear relaxations of the fol-
623 lowing standard integer programming formulation of these hitting set problems
624 using a linear programming (LP) solver.

Concretely, the objective of the integer program (IP) formulation of the
minimum-cost hitting set problem is

$$\min \sum_{k \in K} w(k) \cdot x_k,$$

625 where each binary variable $x_k \in \{0, 1\}$ indicates whether the (in)dependence
626 constraint $k \in K$ is included in the hitting set. In the linear relaxation, we have
627 $x_k \in [0, 1]$.

⁴Given a collection of sets over a set of weighted elements, a minimum-cost hitting set is a subset of the elements such that (i) the hitting set contains at least one element from each of the sets in the collection, and (ii) the sum of the weights of the elements in the hitting set is smallest among all hitting sets of the collection.

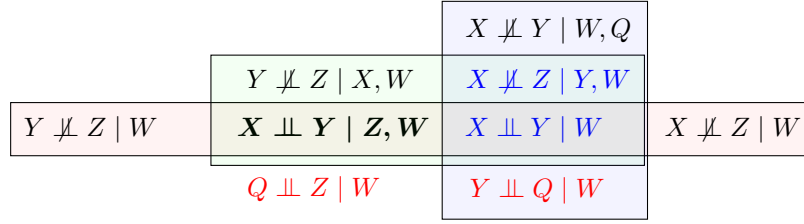


Figure 8: Example of core-based lower bounding.

The unsatisfiable cores form the linear constraints of the IP: for each unsatisfiable core over a set of (in)dependence constraints $k_1, k_2, \dots, k_m \in K$, we include the corresponding linear constraint

$$x_{k_1} + x_{k_2} + \dots + x_{k_m} \geq 1$$

628 to the program, enforcing that at least one of the (in)dependence constraints
629 needs to be included in a hitting set.

630 For obtaining bounds via the linear relaxation of the minimum-cost hitting
631 set IP at a search node during the branch-and-bound search, we simplify the
632 linear relaxation by enforcing 0/1 values on those LP variables corresponding to
633 known (in)dependence constraint states (recall Section 3.2) under the current
634 partial solution. In particular, if a constraint $k \in K$ is known to be satisfied
635 (violated, respectively) under the current partial solution, we enforce $x_k = 0$
636 ($x_k = 1$, respectively) in the linear relaxation, stating that we are not allowed
637 to (we must, respectively) choose k into the hitting set. This way the core-based
638 lower bounds are updated to match the current search tree branch.

Example 8. Suppose we have the cores in Figure 8 and the partial solution satisfies $X \not\perp Z \mid Y, W$ and $X \perp Y \mid W$ (in blue), and violates $Y \perp Q \mid W$ and $Q \perp Z \mid W$ (in red). One constraint in each core marked by the rectangles must be chosen. The violated constraint $Y \perp Q \mid W$ hits the core marked by the vertical violet rectangle. If for simplicity the weights are all constants, the remaining cores can be optimally hit by $X \perp Y \mid Z, W$ (in bold). Thus, the final lower bound for the partial solution is

$$w(Y \perp Q \mid W) + w(Q \perp Z \mid W) + w(X \perp Y \mid Z, W),$$

639 where the last term in the sum tightens the bound compared to the simple bound
640 due to just violated constraints.

641 3.6. Imposing Acyclicity and Sparsity

642 Our approach also allows for integrating different structural search space
643 restrictions. We now explain how to enforce two types of constraints: acyclicity
644 and sparsity.

To enforce acyclicity (in terms of directed edges), we keep track of the set $R[X]$ of nodes reachable by a directed path of decided edges from node X in

the current partial solution. Initially $R[X] = \emptyset$ for each node X . After an edge $X \rightarrow Y$ is decided present, we update

$$R[Z] \leftarrow R[Z] \cup \{Y\} \cup R[Y]$$

645 for each $Z \in \{X\} \cup \{Z' : X \in R[Z']\}$. Using this information, we can decide
646 any edge $X \rightarrow Y$ as absent in all completions of the current partial solution
647 where $X \in R[Y]$.

648 We can also enforce sparsity constraints, such as a bound on the maximum
649 degree of nodes (as used by Claassen et al. (2013)), in a straightforward way. We
650 can simply keep track of the degree for each node in the current partial solution,
651 and decide all the yet-undecided edges between a node pair to be absent if the
652 degree for either node has already reached the maximum allowed value.

653 4. Empirical Evaluation

654 We implemented the branch-and-bound approach and all of the search techni-
655 ques described in Section 3. The resulting open-source C++ implementation
656 *because* of the approach is available at

657 <https://www.cs.helsinki.fi/group/coreo/bcause/>

658 In the following, we present results from an empirical evaluation of the run-
659 ning time performance of *because* on problem instances obtained from real-world
660 datasets from several perspectives: (i) the marginal contribution of the different
661 proposed search techniques, (ii) the efficiency of the approach with respect to
662 current state of the art, and (iii) the impact of the scoring function used for
663 obtaining constraint weights on the efficiency of the approach.

664 The benchmark instances were generated from real-world datasets often used
665 for benchmarking exact Bayesian network structure learning algorithms (Yuan
666 and Malone, 2013; Bartlett and Cussens, 2017) and also used in the original
667 paper describing the *Dseptor* system (Hyttinen et al., 2017b); see Table 1 for
668 more details on the benchmarks. As the basis of the causal discovery instances,
669 we considered suitable-sized (n) subsets of the first non-constant variables in the
670 datasets, making the remaining variables thus latent (recall that latent variables
671 are supported by our general search space). This resulted in a total of 120
672 benchmark instances. For parts (i) and (ii) of the evaluation, we obtained the
673 constraint weights by (local) Bayesian model selection with the BDeu (ESS=10,
674 $\alpha = 0.5$) score⁵; further parameter values are considered in part (iii) of the
675 evaluation. When reporting running times, we do not include the constraint
676 weight computation times. We note that in this problem setting, the weight
677 computation times are negligible to the running times of *Dseptor* or *because*.
678 Concretely, obtaining the constraint weights for any single data set used in the

⁵This is the score used in the original paper describing *Dseptor* (Hyttinen et al., 2017b) and also in the preliminary version of this article (Rantanen et al., 2018).

679 experiments takes less than a second. The longest weight computation time was
680 0.7 seconds, on the Link10000 dataset.

681 For the experiments, we used the CPLEX system as the linear programming
682 solver for obtaining core-based bounds within *bcause*. The experiments were run
683 under a 1-h per-instance time limit on Intel Xeon E5-2680 v4 2.4GHz processors
684 and 256-GB RAM.

685 4.1. Impact of Search Techniques

686 We start the overview of the results by looking at the marginal contribution
687 of the proposed individual search techniques as implemented in *bcause*. Here by
688 marginal contribution we refer to switching off an individual search technique,
689 instead applying a more basic (“baseline”) version of the techniques as necessary,
690 while keeping all search techniques intact. Concretely, we study the marginal
691 contribution of three search techniques.

- 692 • The domain-specific branching heuristics detailed in Section 3.4. As a
693 baseline heuristic, we compare to choosing the edge to branch on uniformly
694 at random (“random branching”).
- 695 • The inference rules 1–4 detailed in Section 3.3, allowing for inferring irrel-
696 evant edges under partial solutions. As a baseline comparison, we simply
697 switch off the rules.
- 698 • The lower bounding technique detailed in Section 3.5 based on solving a
699 linear relaxation of the minimum-cost hitting set problem over the unsatis-
700 fiable core patterns under partial solutions. As a baseline comparison, we
701 switch off this additional bounding technique, and only obtain naive lower
702 bounds by summing up the weights of the constraints that are known to
703 be violated by the current partial solution.

704 The results are shown in Figures 9–11. Each plot gives a comparison of the
705 per-instance running times of the default settings of *bcause* on the x-axis (with
706 all three search techniques switched on) and *bcause* with one of the three tech-
707 niques individually switched off on the y-axis. The shapes of points distinguish
708 between the different numbers of random variables in the underlying datasets
709 (excluding latent variables). As shown in Figure 9, the marginal contribution
710 of the domain-specific branching heuristic is noticeable, as it yields considerable
711 performance gains over using random branching, making the domain-specific
712 branching heuristic integral for *bcause*. As shown in Figure 10, while their im-
713 pact is more moderate, the inference rules 1–4 also consistently speed up search.
714 (Recall that these rules do not have an impact on the number of search tree nodes
715 visited, but rather lower the time spent at each search tree node.) Finally, as
716 shown in Figure 11, the use of the core-based lower bounds obtained via linear
717 programming also considerably speed up *bcause*, and most importantly very
718 consistently for harder instances.

719 In summary, each of the three search techniques has a non-negligible marginal
720 contribution to the performance of *bcause*, each consistently improving the run-
721 ning time performance of the approach.

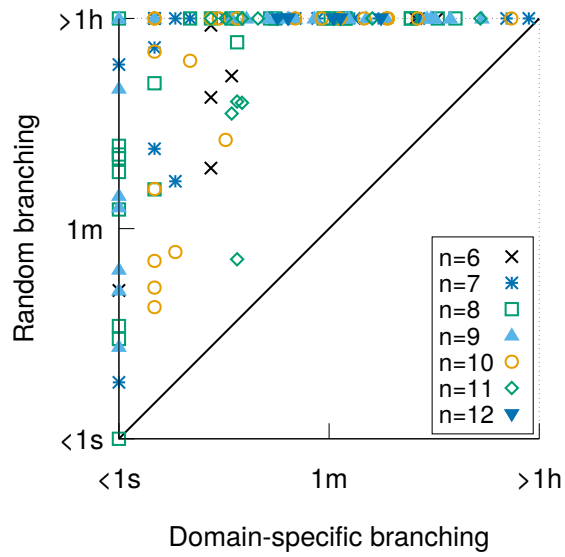


Figure 9: Marginal contribution of the domain-specific branching heuristic: per-instance running time comparison of *bcourse* using domain-specific (x-axis) and random branching (y-axis).

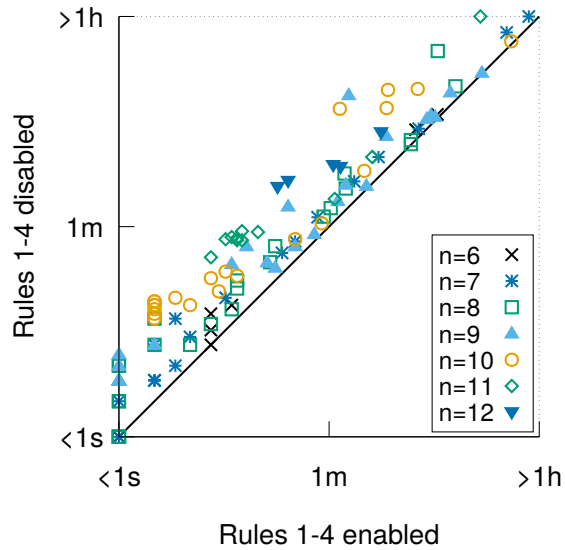


Figure 10: Marginal contribution of the irrelevant edge rules 1–4: per-instance running time comparison of *bcourse* using (x-axis) and not using (y-axis) the rules.

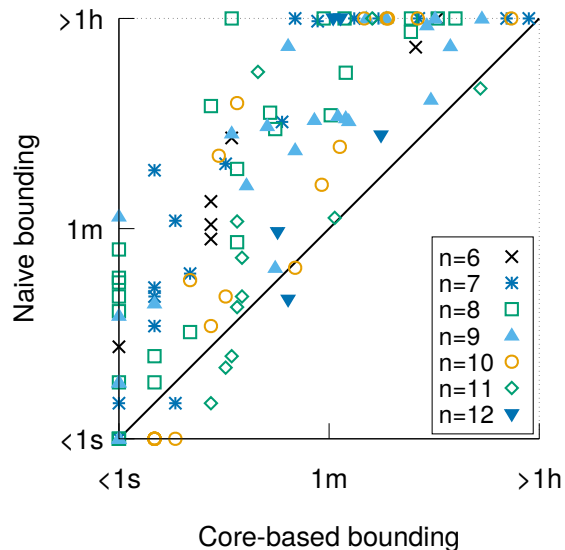


Figure 11: Marginal contribution of the unsatisfiable core linear programming lower bounds: per-instance running time comparison of *bcause* using (x-axis) and not using (y-axis) the core-based bounding.

722 4.2. Comparison with Current State of the Art

723 We compare the running time performance of *bcause* to that of *Dseptor* (Hyt-
 724 tinen et al., 2017b) as representative of the current state of the art. The *Dseptor*
 725 system is based on encoding of the causal discovery problem declaratively using
 726 the Boolean optimization paradigm of maximum satisfiability (MaxSAT),
 727 and furthermore integrates domain-specific techniques within a hybrid MaxSAT
 728 solver (Saikko et al., 2016a) making use of both SAT and integer programming
 729 solvers based on the so-called implicit hitting set paradigm (Moreno-Centeno
 730 and Karp, 2013; Davies and Bacchus, 2013; Saikko et al., 2016b).

731 Here we compare the performance of *bcause* and *Dseptor* in the general,
 732 unrestricted search space (allowing latent variables and cycles) as well as the
 733 restricted acyclic search space (will allowing latent variables). The results are
 734 shown in Figures 12 and 13, and Table 1. The plot in Figure 12 gives the number
 735 of instances solved (x-axis) by each approach under different per-instance
 736 time limits (y-axis); essentially, the further to the right the line, the better over-
 737 all running time performance an approach exhibits. Evidently the performance
 738 of *bcause* is better in terms of the number of instances solved: within the 1-h
 739 per-instance time limit, *bcause* solved over 110 instance (both in the cyclic and
 740 acyclic case), while *Dseptor* solved less than 70 instance (both in the cyclic and
 741 acyclic case). Interestingly, the model space restriction has essentially no effect
 742 on the running times of *bcause*, while enforcing acyclicity degrades the perform-
 743 ance of *Dseptor* slightly. As further seen in Figure 12, on a clear majority
 744 of the benchmarks *bcause* exhibits noticeably better runtimes than *Dseptor* re-

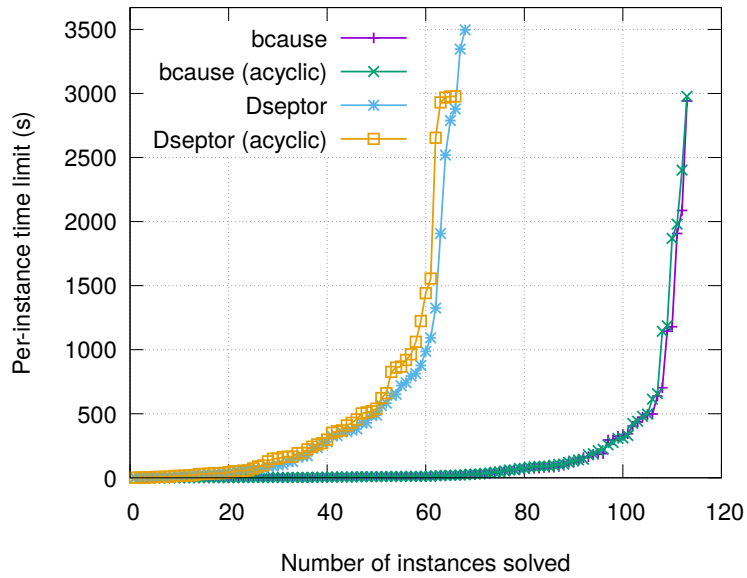


Figure 12: Comparison of *bcause* and *Dseptor*: number of solved instances (*x*-axis) for different per-instance time limits (*y*-axis), for both unrestricted search space and search space restricted to acyclic graphs.

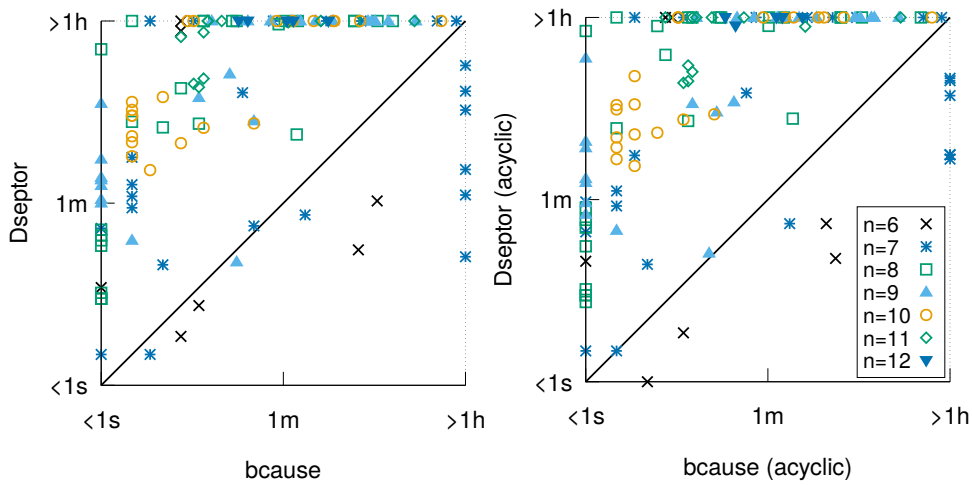


Figure 13: Comparison of *bcause* and *Dseptor*: per-instance running times for both unrestricted search space (left) and search space restricted to acyclic graphs (right).

Table 1: Running times of *bcause* and *Dseptor* over different search spaces.

Dataset	<i>n</i>	Running times (s)					
		General		Acyclic		Max-degree 3	
		<i>bcause</i>	<i>Dseptor</i>	<i>bcause</i>	<i>Dseptor</i>	<i>bcause</i>	
Adult	7	1907 (568)	>3600	1980 (380)	>3600	859	(121)
Alarm	9	403 (3)	>3600	425 (3)	>3600	406	(3)
Autos	9	473 (4)	>3600	479 (4)	>3600	447	(3)
Bands	8	296 (3)	>3600	307 (2)	>3600	276	(2)
Epigenetics	7	>3600 (>3600)	18	>3600 (>3600)	165	>3600	(>3600)
Flag	12	75 (30)	>3600	83 (32)	>3600	59	(15)
Heart	12	27 (24)	>3600	29 (25)	2977	16	(13)
Hepatitis	12	65 (27)	>3600	73 (29)	>3600	50	(13)
Horse.23	9	83 (4)	>3600	90 (4)	>3600	82	(3)
Horse	11	11 (7)	>3600	12 (7)	>3600	8	(4)
Image	8	703 (21)	>3600	1869 (46)	>3600	706	(43)
Imports	8	295 (1)	>3600	311 (1)	>3600	352	(1)
Letter	7	>3600 (>3600)	487	>3600 (>3600)	622	>3600	(>3600)
LungCancer	10	184 (10)	>3600	212 (11)	>3600	316	(15)
Meta	7	>3600 (>3600)	128	>3600 (>3600)	149	>3600	(>3600)
Mushroom1000	7	>3600 (>3600)	1325	>3600 (>3600)	868	>3600	(>3600)
Mushroom8124	7	>3600 (>3600)	743	>3600 (>3600)	919	>3600	(>3600)
Parkinsons	7	344 (27)	>3600	257 (20)	>3600	156	(14)
Sensors	7	>3600 (>3600)	72	>3600 (>3600)	164	>3600	(>3600)
Soybean	11	10 (8)	2789	10 (8)	1223	7	(5)
Spectf	11	10 (8)	987	11 (9)	1061	10	(6)
Statlog	7	31 (14)	36	441 (78)	>3600	31	(17)
SteelPlates	6	494 (460)	63	223 (191)	35	51	(25)
Voting	9	637 (7)	>3600	614 (7)	>3600	736	(7)
Water	9	436 (12)	>3600	657 (2)	>3600	486	(7)
Wdbc	8	54 (37)	3346	61 (42)	2977	55	(35)
Wine	9	1179 (9)	>3600	1143 (9)	>3600	1344	(10)
Zoo	7	157 (82)	>3600	145 (69)	>3600	152	(67)
alarm10000	10	>3600 (>3600)	>3600	>3600 (>3600)	>3600	2547	(34)
alarm1000	10	119 (37)	>3600	107 (22)	>3600	108	(7)
alarm100	9	18 (1)	1092	19 (1)	429	20	(1)
asia10000	8	81 (51)	281	105 (70)	370	180	(95)
asia1000	7	98 (<1)	46	98 (<1)	35	94	(29)
asia100	7	3 (<1)	2	2 (<1)	2	3	(<1)
carpo10000	11	139 (65)	>3600	139 (64)	2930	64	(28)
carpo1000	10	2087 (2)	>3600	2402 (2)	>3600	2221	(2)
carpo100	10	52 (1)	>3600	54 (1)	>3600	48	(1)
Diabetes10000	8	<1 (<1)	7	<1 (<1)	6	<1	(<1)
Diabetes1000	8	<1 (<1)	7	<1 (<1)	7	<1	(<1)
Diabetes100	9	88 (<1)	>3600	90 (<1)	>3600	103	(<1)
hailfinder10000	10	2 (2)	429	2 (2)	194	2	(1)
hailfinder1000	9	1 (<1)	65	1 (<1)	43	1	(<1)
hailfinder100	8	498 (<1)	>3600	497 (<1)	>3600	587	(<1)
insurance10000	9	183 (1)	>3600	179 (1)	>3600	206	(1)
insurance100	10	337 (4)	>3600	330 (3)	>3600	378	(3)
Link10000	12	165 (13)	>3600	131 (12)	>3600	187	(14)
Link1000	11	1144 (4)	>3600	1186 (4)	>3600	1362	(4)
Link100	10	189 (3)	>3600	188 (3)	>3600	214	(2)
Mildew10000	10	4 (3)	652	3 (2)	964	3	(2)
Mildew1000	8	1 (<1)	33	1 (<1)	38	1	(<1)
Mildew100	7	2941 (1247)	>3600	2977 (1258)	>3600	2738	(1021)
Pigs10000	10	74 (1)	>3600	88 (1)	>3600	85	(1)
Pigs1000	10	7 (3)	>3600	8 (3)	>3600	6	(2)
Pigs100	8	<1 (<1)	23	<1 (<1)	21	<1	(<1)
Water10000	9	71 (1)	>3600	74 (2)	>3600	78	(1)
Water1000	12	22 (20)	>3600	23 (21)	>3600	19	(17)
Water100	11	15 (8)	>3600	17 (9)	>3600	13	(6)

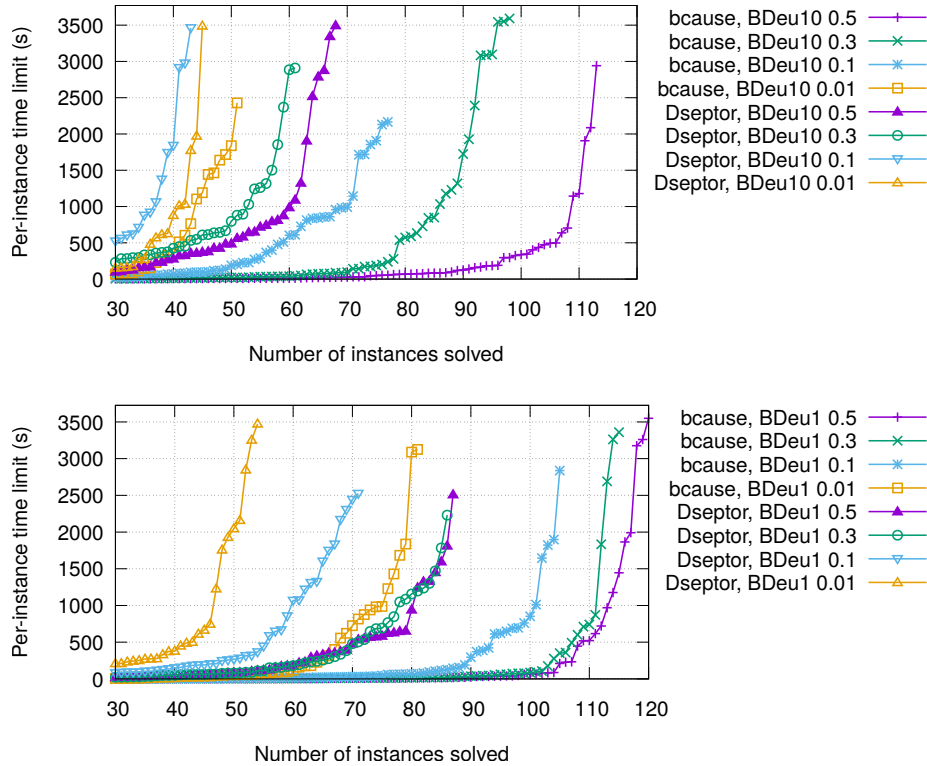


Figure 14: Comparison of *bcause* and *Dseptor*: number of solved instances (x-axis) for different per-instance time limits (y-axis), with constraint weights obtained by BDeu scoring using ESS 1 (top) and 10 (bottom) with prior probabilities of independence (α) as 0.5, 0.3, 0.1 and 0.01.

745 regardless of whether acyclicity is enforced, and times out less frequently, with
 746 7 and 52 timeouts, respectively, without enforcing acyclicity, 7 and 54 time-
 747 outs under acyclicity. Table 1 gives per-instance details for largest n instances,
 748 with the time to reach an optimal solution (without yet proving optimality)
 749 shown for *bcause* in parentheses. Furthermore, the Max-degree 3 column gives
 750 runtimes for *bcause* when enforcing that the maximum node degree is at most
 751 three. The better running time between *bcause* and *Dseptor* for each instance
 752 and search spaces is given in bold. Apart from the fact that *bcause* quite consis-
 753 tently exhibits better running times than *Dseptor*, we also observe that *bcause*
 754 exhibits very good anytime performance in that it reaches an optimal solution
 755 often relatively fast.

756 4.3. Impact of Scoring Function Parameters

757 To further study the scalability of *bcause* and *Dseptor* and the impact of the
 758 scoring function parameters used to generate the causal discovery instances,

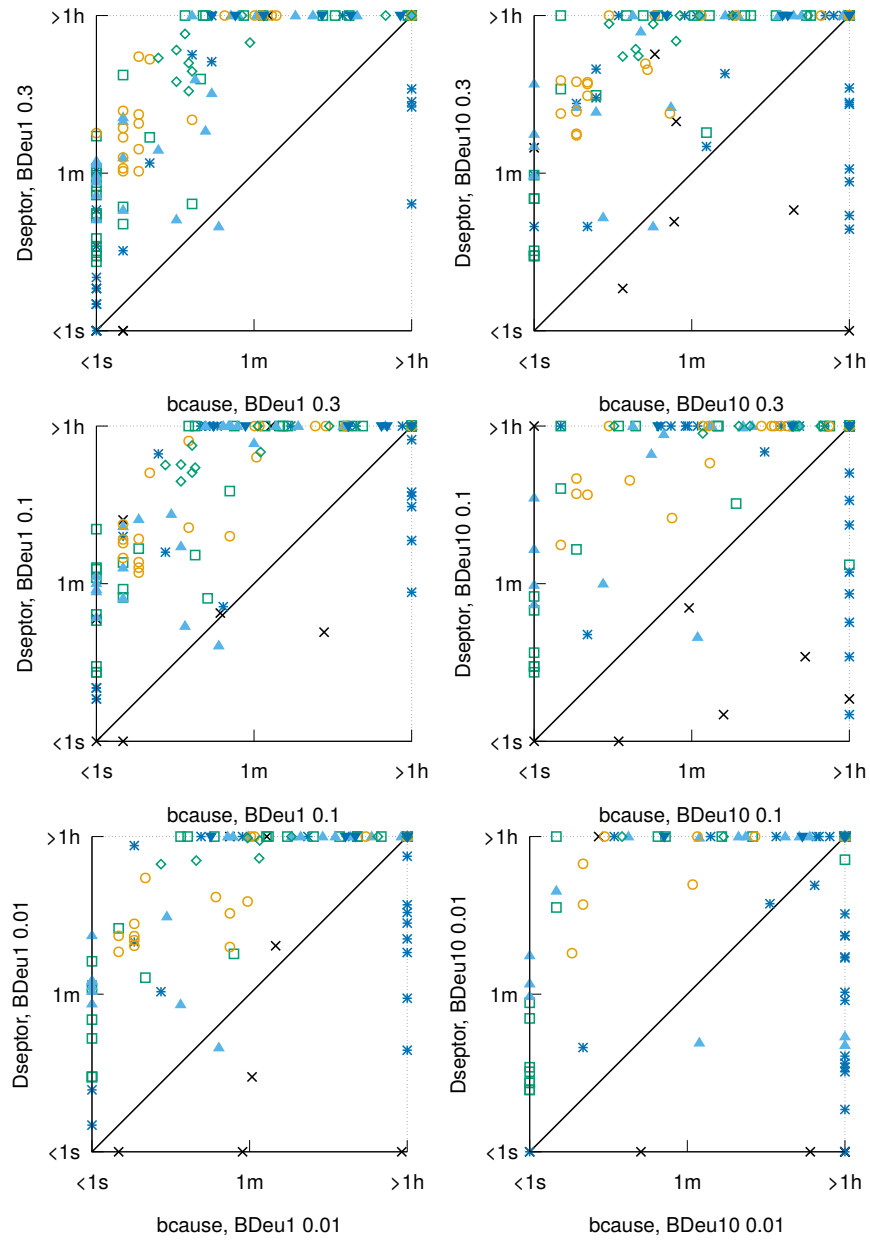


Figure 15: Comparison of *bcause* and *Dseptor*: per-instance running times with constraint weights obtained by BDeu scoring using ESS 1 (left column) and 10 (right column) and prior probabilities of independence as $\alpha = 0.3$ (top row), $\alpha = 0.1$ (middle row) and $\alpha = 0.01$ (bottom row).

759 we generated further problem instances based on the same datasets using BDeu
760 scoring with equivalent sample sizes of 1 and 10, and prior probabilities of
761 independence (α) 0.5, 0.3, 0.1 and 0.01.

762 As seen in Figure 14, for any fixed choice of scoring function parameters,
763 *bcause* is able to solve more instances than *Dseptor*. A per-instance comparison
764 for the individual parameter value pairs is shown in Figure 15. Interestingly,
765 while *bcause* exhibits better overall performance than *Dseptor* on any choice
766 of the parameters, the choice of scoring function parameters has a noticeable
767 impact on the scalability of both *bcause* and *Dseptor*. We hypothesize this
768 to originate from the intuition that lower prior probabilities of independence
769 often result in more dependencies, which in turn may translate to many of the
770 optimal graphs being denser. Considering *bcause*, the increased density may
771 make the problem instances more difficult to solve in two ways. Firstly, the more
772 independences there are, the easier it is for our independence-based heuristic to
773 navigate in the search tree to the optimal solution. The second reason concerns
774 the score-equivalent solutions in the overall search space, i.e., graphs that share
775 the exact same weight. In particular, some causal graphs are able to satisfy the
776 same set of (in)dependence constraints even after some edges are removed or
777 reoriented (see Section 3.3). Intuitively, the more edges we assign within the
778 search, the more equivalent solutions we are likely to encounter, which can to
779 an extent be detrimental to search performance.

780 5. Related Work

781 Declarative Boolean satisfiability (SAT) solvers were first used in (Triantafilou
782 et al., 2010; Triantafilou et al., 2010; Hyttinen et al., 2013) for developing
783 approaches to discovering causal structures with latent confounders from several
784 data sets in constraint-based fashion. Building on these ideas, Hyttinen et al.
785 (2014) proposed the first exact approach to the problem setting considered in
786 this article, in the form of a declarative framework in the language of answer set
787 programming (ASP). This framework was subsequently adapted to formulate a
788 relaxed version with focus on several experimental data sets (Magliacane et al.,
789 2016) and to examine different types of relaxed faithfulness conditions (Zhalama
790 et al., 2017). Furthermore, a different encoding was proposed by Borboudakis
791 and Tsamardinos (2016). Forré and Mooij (2018) apply a different separation
792 condition for non-linear cyclic models. Up until now, the current state of the
793 art to the exact problem setting we consider here is the recent maximum satisfi-
794 ability (MaxSAT) based approach developed by Hyttinen et al. (2017b), where
795 domain-specific techniques were integrated to the extent possible to a MaxSAT
796 solver, still relying on a MaxSAT solver to solve the search problem starting
797 with a declarative encoding of the problem.

798 Several inexact constraint-based algorithm are available for learning causal
799 graphs under restricting assumptions. FCI learns acyclic graphs allowing for
800 latent confounding and selection bias (Spirtes et al., 2000). CCD learns graphs
801 with cycles (Richardson, 1996a). RFCI of Colombo et al. (2012) improves on
802 efficiency of FCI. Claassen et al. (2013) developed a polynomial-time FCI-type

803 algorithm for discovering edge-minimal acyclic graphs with latent variables. Or-
804 der independent versions of FCI and CCD by Colombo and Maathuis (2014) give
805 more stable results in sparse high-dimensional settings.

806 Apart from these methods aiming for scalability, there are approaches focus-
807 ing on accuracy. Conservative FCI of Ramsey et al. (2012) performs additional
808 independence tests to not output orientations due to conflicting constraints.
809 Claassen and Heskes (2012) combine weighted independence constraints with
810 an inexact FCI-type procedure. Ogarrio et al. (2016) use FCI orientation rules
811 to detect latent confounders over a skeleton obtained by a greedy score-based
812 approach. Jabbari et al. (2017) use RFCI to find candidate PAGs fitting data
813 and to generate a set of (in)dependence constraints K . Then an optimal graph
814 is found among the candidates over an objective function consisting of weighted
815 independence constraints in K .

816 For models with parametric restrictions, also score-based algorithms have
817 been proposed. Evans and Richardson (2010) find maximum likelihood param-
818 eters for binary semi-Markovian models and Drton et al. (2009) for linear SEMs
819 with correlated errors (both acyclic). Subsequently high-scoring graphs can be
820 found with a greedy procedure e.g. using a BIC penalty (Evans and Richardson,
821 2010; Tsirlis et al., 2018).

822 In contrast to this related work, our procedure tackles a more general search
823 space allowing for cycles and latent confounders and makes no parametric as-
824 sumptions as such; it also offers guaranteed optimality of the solution. Our
825 approach also straightforwardly generalizes to several data sets with partially
826 overlapping variable sets (e.g. by simply combining weights for the constraints
827 testable in several data sets) (Tillman and Spirtes, 2011; Tillman et al., 2008;
828 Triantafillou et al., 2010; Hyttinen et al., 2014). Finally, we note that differ-
829 ent branch-and-bound style algorithms have also been proposed for learning the
830 structure of Bayesian networks (Suzuki, 1996; Tian, 2000; Malone and Yuan,
831 2014; van Beek and Hoffmann, 2015; Suzuki and Kawahara, 2017) and chordal
832 Markov networks (Rantanen et al., 2017).

833 6. Conclusions

834 We developed a novel branch-and-bound approach to learning provably-
835 optimal causal graphs in general search spaces. In contrast to the earlier ap-
836 proaches heavily relying on declarative optimization solvers, our approach is
837 a specialized algorithm for the problem, and integrates knowledge about the
838 problem domain for speeding up search via problem-specific branching heuris-
839 tics; optimized algorithms for evaluating the objective function of the problem
840 during search and inference rules for detecting which edges are irrelevant in
841 terms of d -connectivity under a current partial solution; as well as integrating
842 linear programming relaxation computations for lower bounding applicable dur-
843 ing search based on problem-specific unsatisfiable cores. As we explained, the
844 approach also allows for integrating search space restrictions, such as acyclicity
845 or a degree bound, to the search. Through an extensive empirical evaluation, we
846 showed that our open-source implementation *because* of the approach improves

847 on current state of the art in exact approaches to learning optimal causal graphs
848 in terms of running times on real-world datasets. We foresee various direction
849 for further work. For example, the approach could be modified to use different
850 separation criteria, to account for phenomena such as selection bias, measure-
851 ment noise and also data recorded in multiple different contexts. For runtime
852 improvements, the approach currently does not make use of problem-specific
853 symmetries or parallel computations. The potential of further search heuristics,
854 including lookahead, could also be studied. Furthermore, the impact of dataset
855 properties on the relative runtime performance of *Dseptor* and *bcause* could
856 yield further insights into which types of search techniques result in improved
857 performance on individual datasets.

858 Acknowledgments

859 Work supported by Academy of Finland (grants 276412, 295673, 312662)
860 and the Research Funds of the University of Helsinki.

861 References

- 862 Abellán, J., Gómez-Olmedo, M., Moral, S., 2006. Some variations on the PC
863 algorithm, in: Studený, M., Vomlel, J. (Eds.), Third European Workshop
864 on Probabilistic Graphical Models, 12-15 September 2006, Prague, Czech
865 Republic. Electronic Proceedings., pp. 1–8.
- 866 Bartlett, M., Cussens, J., 2017. Integer linear programming for the Bayesian
867 network structure learning problem. *Artificial Intelligence* 244, 258–271.
- 868 van Beek, P., Hoffmann, H., 2015. Machine learning of Bayesian networks
869 using constraint programming, in: Pesant, G. (Ed.), Principles and Practice
870 of Constraint Programming - 21st International Conference, CP 2015, Cork,
871 Ireland, August 31 - September 4, 2015, Proceedings, Springer. pp. 429–445.
- 872 Biere, A., Heule, M., van Maaren, H., Walsh, T. (Eds.), 2009. Handbook of Sat-
873 isfiability. volume 185 of *Frontiers in Artificial Intelligence and Applications*,
874 IOS Press.
- 875 Bollen, K.A., 1989. Structural Equations with Latent Variables. John Wiley &
876 Sons.
- 877 Borboudakis, G., Tsamardinos, I., 2016. Towards robust and versatile causal
878 discovery for business applications, in: Krishnapuram, B., Shah, M., Smola,
879 A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (Eds.), Proceedings of the 22nd
880 ACM SIGKDD International Conference on Knowledge Discovery and Data
881 Mining, San Francisco, CA, USA, August 13-17, 2016, ACM. pp. 1435–1444.
- 882 Buntine, W.L., 1991. Theory refinement on Bayesian networks, in: D’Ambrosio,
883 B., Smets, P. (Eds.), UAI ’91: Proceedings of the Seventh Annual Confer-
884 ence on Uncertainty in Artificial Intelligence, University of California at Los

- 885 Angeles, Los Angeles, CA, USA, July 13-15, 1991, Morgan Kaufmann. pp.
886 52–60.
- 887 Claassen, T., Heskes, T., 2012. A Bayesian approach to constraint based causal
888 inference, in: de Freitas, N., Murphy, K.P. (Eds.), Proceedings of the Twenty-
889 Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island,
890 CA, USA, August 14-18, 2012, AUAI Press. pp. 207–216.
- 891 Claassen, T., Mooij, J.M., Heskes, T., 2013. Learning sparse causal models is
892 not NP-hard, in: Nicholson, A., Smyth, P. (Eds.), Proceedings of the Twenty-
893 Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Belle-
894 vue, WA, USA, August 11-15, 2013, AUAI Press. pp. 172–181.
- 895 Colombo, D., Maathuis, M.H., 2014. Order-independent constraint-based causal
896 structure learning. *Journal of Machine Learning Research* 15, 3741–3782.
- 897 Colombo, D., Maathuis, M.H., Kalisch, M., Richardson, T.S., 2012. Learning
898 high-dimensional directed acyclic graphs with latent and selection variables.
899 *Annals of Statistics* 40, 294–321.
- 900 Cooper, G.F., 1997. A simple constraint-based algorithm for efficiently mining
901 observational databases for causal relationships. *Data Min. Knowl. Discov.*
902 1, 203–224.
- 903 Cooper, G.F., Herskovits, E., 1992. A Bayesian method for the induction of
904 probabilistic networks from data. *Machine Learning* 9, 309–347.
- 905 Davies, J., Bacchus, F., 2013. Exploiting the power of MIP solvers in MAXSAT,
906 in: Järvisalo, M., Gelder, A.V. (Eds.), Theory and Applications of Satisfiability
907 Testing - SAT 2013 - 16th International Conference, Helsinki, Finland,
908 July 8-12, 2013. Proceedings, Springer. pp. 166–181.
- 909 Drton, M., Eichler, M., Richardson, T.S., 2009. Computing maximum likelihood
910 estimates in recursive linear models with correlated errors. *Journal of Machine
911 Learning Research* 10, 2329–2348.
- 912 Evans, R.J., Richardson, T.S., 2010. Maximum likelihood fitting of acyclic di-
913 rected mixed graphs to binary data, in: Grünwald, P., Spirtes, P. (Eds.), UAI
914 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial
915 Intelligence, Catalina Island, CA, USA, July 8-11, 2010, AUAI Press. pp.
916 177–184.
- 917 Forré, P., Mooij, J.M., 2017. Markov properties for graphical models with cycles
918 and latent variables. arXiv.org preprint arXiv:1710.08775 [math.ST].
- 919 Forré, P., Mooij, J.M., 2018. Constraint-based causal discovery for non-linear
920 structural causal models with cycles and latent confounders, in: Globerson,
921 A., Silva, R. (Eds.), Proceedings of the Thirty-Fourth Conference on Uncer-
922 tainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August
923 6-10, 2018, AUAI Press. pp. 269–278.

- 924 Geiger, D., Heckerman, D., 2002. Parameter priors for directed acyclic graphical
925 models and the characterization of several probability distributions. *The*
926 *Annals of Statistics* 30, 1412–1440.
- 927 Heckerman, D., Geiger, D., Chickering, D.M., 1995. Learning Bayesian net-
928 works: The combination of knowledge and statistical data. *Machine Learning*
929 20, 197–243.
- 930 Hyttinen, A., Eberhardt, F., Hoyer, P.O., 2012. Learning linear cyclic causal
931 models with latent variables. *Journal of Machine Learning Research* 13, 3387–
932 3439.
- 933 Hyttinen, A., Eberhardt, F., Järvisalo, M., 2014. Constraint-based causal dis-
934 covery: Conflict resolution with answer set programming, in: Zhang, N.L.,
935 Tian, J. (Eds.), *Proceedings of the Thirtieth Conference on Uncertainty in*
936 *Artificial Intelligence, UAI 2014, Quebec City, Quebec, Canada, July 23-27,*
937 *2014, AUAI Press.* pp. 340–349.
- 938 Hyttinen, A., Hoyer, P.O., Eberhardt, F., Järvisalo, M., 2013. Discovering
939 cyclic causal models with latent variables: A general sat-based procedure, in:
940 Nicholson, A., Smyth, P. (Eds.), *Proceedings of the Twenty-Ninth Confer-*
941 *ence on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA,*
942 *August 11-15, 2013, AUAI Press.* pp. 301–310.
- 943 Hyttinen, A., Plis, S.M., Järvisalo, M., Eberhardt, F., Danks, D., 2017a. A
944 constraint optimization approach to causal discovery from subsampled time
945 series data. *International Journal of Approximate Reasoning* 90, 208–225.
- 946 Hyttinen, A., Saikko, P., Järvisalo, M., 2017b. A core-guided approach to learn-
947 ing optimal causal graphs, in: Sierra, C. (Ed.), *Proceedings of the Twenty-*
948 *Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017,*
949 *Melbourne, Australia, August 19-25, 2017, ijcai.org.* pp. 645–651.
- 950 Jabbari, F., Ramsey, J., Spirtes, P., Cooper, G.F., 2017. Discovery of causal
951 models that contain latent variables through Bayesian scoring of independence
952 constraints, in: *Machine Learning and Knowledge Discovery in Databases -*
953 *European Conference, ECML PKDD 2017, Skopje, Macedonia, September*
954 *18-22, 2017, Proceedings, Part II, Springer.* pp. 142–157.
- 955 Lacerda, G., Spirtes, P., Ramsey, J., Hoyer, P.O., 2008. Discovering cyclic
956 causal models by independent components analysis, in: McAllester, D.A.,
957 Myllymäki, P. (Eds.), *UAI 2008, Proceedings of the 24th Conference in Un-*
958 *certainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008, AUAI*
959 *Press.* pp. 366–374.
- 960 Magliacane, S., Claassen, T., Mooij, J.M., 2016. Ancestral causal inference,
961 in: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R.
962 (Eds.), *Advances in Neural Information Processing Systems 29: Annual Con-*
963 *ference on Neural Information Processing Systems 2016, December 5-10, 2016,*
964 *Barcelona, Spain, Curran Associates, Inc..* pp. 4466–4474.

- 965 Malone, B., Järvisalo, M., Myllymäki, P., 2015. Impact of learning strategies
966 on the quality of Bayesian networks: An empirical evaluation, in: Meila,
967 M., Heskes, T. (Eds.), Proceedings of the Thirty-First Conference on Uncer-
968 tainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, The
969 Netherlands, AUAI Press. pp. 562–571.
- 970 Malone, B.M., Yuan, C., 2014. A depth-first branch and bound algorithm for
971 learning optimal Bayesian networks, in: Croitoru, M., Rudolph, S., Woltran,
972 S., Gonzales, C. (Eds.), Graph Structures for Knowledge Representation and
973 Reasoning - Third International Workshop, GKR 2013, Beijing, China, Au-
974 gust 3, 2013. Revised Selected Papers, Springer. pp. 111–122.
- 975 Margaritis, D., Bromberg, F., 2009. Efficient Markov network discovery using
976 particle filters. *Computational Intelligence* 25, 367–394.
- 977 Moreno-Centeno, E., Karp, R.M., 2013. The implicit hitting set ap-
978 proach to solve combinatorial optimization problems with an application to
979 multigenome alignment. *Operations Research* 61, 453–468.
- 980 Natori, K., Uto, M., Ueno, M., 2017. Consistent learning bayesian networks with
981 thousands of variables, in: Hyttinen, A., Suzuki, J., Malone, B.M. (Eds.),
982 Proceedings of the 3rd Workshop on Advanced Methodologies for Bayesian
983 Networks, AMBN 2017, Kyoto, Japan, September 20-22, 2017, PMLR. pp.
984 57–68.
- 985 Neal, R., 2000. On deducing conditional independence from d-separation in
986 causal graphs with feedback. *Journal of Artificial Intelligence Research* 12,
987 87–91.
- 988 Ogarrio, J.M., Spirtes, P., Ramsey, J., 2016. A hybrid causal search algo-
989 rithm for latent variable models, in: Antonucci, A., Corani, G., de Campos,
990 C.P. (Eds.), Probabilistic Graphical Models - Eighth International Confer-
991 ence, PGM 2016, Lugano, Switzerland, September 6-9, 2016. Proceedings,
992 JMLR.org. pp. 368–379.
- 993 Pearl, J., 2000. *Causality: Models, Reasoning, and Inference*. Cambridge Uni-
994 versity Press.
- 995 Pearl, J., Dechter, R., 1996. Identifying independencies in causal graphs with
996 feedback, in: Horvitz, E., Jensen, F.V. (Eds.), UAI '96: Proceedings of the
997 Twelfth Annual Conference on Uncertainty in Artificial Intelligence, Reed
998 College, Portland, Oregon, USA, August 1-4, 1996, Morgan Kaufmann. pp.
999 420–426.
- 1000 Ramsey, J., Zhang, J., Spirtes, P., 2012. Adjacency-faithfulness and conservative
1001 causal inference. *CoRR* abs/1206.6843.
- 1002 Rantanen, K., Hyttinen, A., Järvisalo, M., 2017. Learning chordal markov
1003 networks via branch and bound, in: Guyon, I., von Luxburg, U., Bengio,

- 1004 S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (Eds.),
1005 Advances in Neural Information Processing Systems 30: Annual Conference
1006 on Neural Information Processing Systems 2017, 4-9 December 2017, Long
1007 Beach, CA, USA, pp. 1845–1855.
- 1008 Rantanen, K., Hyttinen, A., Järvisalo, M., 2018. Learning optimal causal graphs
1009 with exact search, in: Studený, M., Kratochvíl, V. (Eds.), International Con-
1010 ference on Probabilistic Graphical Models, PGM 2018, 11-14 September 2018,
1011 Prague, Czech Republic, PMLR. pp. 344–355.
- 1012 Richardson, T., 1996a. A discovery algorithm for directed cyclic graphs, in:
1013 Horvitz, E., Jensen, F.V. (Eds.), UAI '96: Proceedings of the Twelfth Annual
1014 Conference on Uncertainty in Artificial Intelligence, Reed College, Portland,
1015 Oregon, USA, August 1-4, 1996, Morgan Kaufmann. pp. 454–461.
- 1016 Richardson, T.S., 1996b. Feedback Models: Interpretation and Discovery. Ph.D.
1017 thesis. Carnegie Mellon University.
- 1018 Saikko, P., Berg, J., Järvisalo, M., 2016a. LMHS: A SAT-IP hybrid MaxSAT
1019 solver, in: Creignou, N., Berre, D.L. (Eds.), Theory and Applications of
1020 Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux,
1021 France, July 5-8, 2016, Proceedings, Springer. pp. 539–546.
- 1022 Saikko, P., Wallner, J.P., Järvisalo, M., 2016b. Implicit hitting set algorithms
1023 for reasoning beyond NP, in: Baral, C., Delgrande, J.P., Wolter, F. (Eds.),
1024 Principles of Knowledge Representation and Reasoning: Proceedings of the
1025 Fifteenth International Conference, KR 2016, Cape Town, South Africa, April
1026 25-29, 2016., AAAI Press. pp. 104–113.
- 1027 Shachter, R.D., 1998. Bayes-Ball: The rational pastime (for determining irrel-
1028 evance and requisite information in belief networks and influence diagrams),
1029 in: Cooper, G.F., Moral, S. (Eds.), UAI '98: Proceedings of the Fourteenth
1030 Conference on Uncertainty in Artificial Intelligence, University of Wisconsin
1031 Business School, Madison, Wisconsin, USA, July 24-26, 1998, Morgan Kauf-
1032 mann. pp. 480–487.
- 1033 Spirtes, P., 1995. Directed cyclic graphical representations of feedback models,
1034 in: Besnard, P., Hanks, S. (Eds.), UAI '95: Proceedings of the Eleventh An-
1035 nual Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec,
1036 Canada, August 18-20, 1995, Morgan Kaufmann. pp. 491–498.
- 1037 Spirtes, P., Glymour, C., Scheines, R., 2000. Causation, Prediction, and Search.
1038 MIT Press.
- 1039 Steck, H., Jaakkola, T.S., 2002. On the dirichlet prior and bayesian regulariza-
1040 tion, in: Becker, S., Thrun, S., Obermayer, K. (Eds.), Advances in Neural
1041 Information Processing Systems 15 [Neural Information Processing Systems,
1042 NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada],
1043 MIT Press. pp. 697–704.

- 1044 Studený, M., 1998. Bayesian networks from the point of view of chain graphs,
1045 in: Cooper, G.F., Moral, S. (Eds.), UAI '98: Proceedings of the Fourteenth
1046 Conference on Uncertainty in Artificial Intelligence, University of Wisconsin
1047 Business School, Madison, Wisconsin, USA, July 24-26, 1998, Morgan Kauf-
1048 mann. pp. 496–503.
- 1049 Suzuki, J., 1996. Learning Bayesian belief networks based on the minimum de-
1050 scription length principle: An efficient algorithm using the B & B technique,
1051 in: Saitta, L. (Ed.), Machine Learning, Proceedings of the Thirteenth Interna-
1052 tional Conference (ICML '96), Bari, Italy, July 3-6, 1996, Morgan Kaufmann.
1053 pp. 462–470.
- 1054 Suzuki, J., Kawahara, J., 2017. Branch and bound for regular Bayesian network
1055 structure learning, in: Elidan, G., Kersting, K., Ihler, A.T. (Eds.), Proceed-
1056 ings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence,
1057 UAI 2017, Sydney, Australia, August 11-15, 2017, AUAI Press.
- 1058 Tian, J., 2000. A branch-and-bound algorithm for MDL learning Bayesian net-
1059 works, in: Boutilier, C., Goldszmidt, M. (Eds.), UAI '00: Proceedings of the
1060 16th Conference in Uncertainty in Artificial Intelligence, Stanford University,
1061 Stanford, California, USA, June 30 - July 3, 2000, Morgan Kaufmann. pp.
1062 580–588.
- 1063 Tillman, R.E., Danks, D., Glymour, C., 2008. Integrating locally learned causal
1064 structures with overlapping variables, in: Koller, D., Schuurmans, D., Bengio,
1065 Y., Bottou, L. (Eds.), Advances in Neural Information Processing Systems 21,
1066 Proceedings of the Twenty-Second Annual Conference on Neural Information
1067 Processing Systems, Vancouver, British Columbia, Canada, December 8-11,
1068 2008, Curran Associates, Inc.. pp. 1665–1672.
- 1069 Tillman, R.E., Spirtes, P., 2011. Learning equivalence classes of acyclic models
1070 with latent and selection variables from multiple datasets with overlapping
1071 variables, in: Gordon, G.J., Dunson, D.B., Dudík, M. (Eds.), Proceedings of
1072 the Fourteenth International Conference on Artificial Intelligence and Statis-
1073 tics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011, JMLR.org.
1074 pp. 3–15.
- 1075 Triantafillou, S., Tsamardinos, I., Tollis, I.G., 2010. Learning causal structure
1076 from overlapping variable sets, in: AISTATS, JMLR. pp. 860–867.
- 1077 Triantafillou, S., Tsamardinos, I., Tollis, I.G., 2010. Learning causal structure
1078 from overlapping variable sets, in: Teh, Y.W., Titterton, D.M. (Eds.),
1079 Proceedings of the Thirteenth International Conference on Artificial Intelli-
1080 gence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy,
1081 May 13-15, 2010, JMLR.org. pp. 860–867.
- 1082 Tsirlis, K., Lagani, V., Triantafillou, S., Tsamardinos, I., 2018. On scoring max-
1083 imal ancestral graphs with the max-min hill climbing algorithm. *International*
1084 *Journal on Approximate Reasoning* 102, 74–85.

- 1085 Verma, T., Pearl, J., 1990. Equivalence and synthesis of causal models, in:
1086 Bonissone, P.P., Henrion, M., Kanal, L.N., Lemmer, J.F. (Eds.), UAI '90:
1087 Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intel-
1088 ligence, MIT, Cambridge, MA, USA, July 27-29, 1990, Elsevier. pp. 255–270.
- 1089 Wright, S., 1934. The method of path coefficients. *Annals of Mathematical*
1090 *Statistics* 5, 161–215.
- 1091 Yuan, C., Malone, B., 2013. Learning optimal Bayesian networks: A shortest
1092 path perspective. *Journal of Artificial Intelligence Research* 48, 23–65.
- 1093 Zhalama, Zhang, J., Eberhardt, F., Mayer, W., 2017. SAT-based causal dis-
1094 covery under weaker assumptions, in: Elidan, G., Kersting, K., Ihler, A.T.
1095 (Eds.), *Proceedings of the Thirty-Third Conference on Uncertainty in Arti-*
1096 *ficial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*, AUAI
1097 Press.