A Core-Guided Approach to Learning Optimal Causal Graphs

Antti Hyttinen and **Paul Saikko** and **Matti Järvisalo**
HIIT, Department of Computer Science, University of Helsinki, Finland

Abstract

Discovery of causal relations is an important part of data analysis. Recent exact Boolean optimization approaches enable tackling very general search spaces of causal graphs with feedback cycles and latent confounders, simultaneously obtaining high accuracy by optimally combining conflicting independence information in sample data. We propose several domain-specific techniques and integrate them into a core-guided maximum satisfiability solver, thereby speeding up current state of the art in *exact* search for causal graphs with cycles and latent confounders on simulated and real-world data.

1 Introduction

Discovery of causal relations from sample data is an important part of data analysis for various application fields, and thus an important area of AI research. Causal relation structures are generally represented as directed graphs, where nodes represent measurements and directed edges denote directed causal relations. It has been shown that many features of these structures can be determined even from passively observed data [Pearl, 2000; Spirtes *et al.*, 2000]. The difficulty lies in determining presence of edges and their directionality, especially when latent confounders (unobserved common causes) and feedback (directed cycles) may be present.

For restricted settings without latent confounders and feedback cycles, various well-performing algorithms for causal discovery have been developed. When aiming for accuracy rather than very high scalability, the most accurate choices are *exact* score-based Bayesian network structure learning algorithms [Malone *et al.*, 2015], which provide *provably globally optimal graphs* as solutions under well-defined (e.g. Bayesian marginal likelihood) scoring functions [Yuan and Malone, 2013; Bartlett and Cussens, 2017; Berg *et al.*, 2014] by employing e.g. constraint optimization methods such as integer programming (IP) and maximum satisfiability (MaxSAT).

However, much less progress has been made for *exact* discovery algorithms when allowing for the presence of latent confounders or feedback. To handle these more general search spaces, the classic constraint-based causal discovery algorithms [Spirtes *et al.*, 2000; Pearl, 2000] use (in)dependence constraints obtained from statistical tests to narrow down the candidate graphs that may have produced the given data. The classic *inexact* algorithms, such as PC, CCD and FCI, scale up by pruning out independence tests based on earlier test results [Spirtes *et al.*, 2000; Richardson, 1996]. Unfortunately, these methods are not very accurate in practice: early mistakes often jeopardize the learning results [Claassen and Heskes, 2012; Hyttinen *et al.*, 2014].

Conflicting independence test results from sample data give rise to a combinatorial optimization problem which, when allowing for latent confounders and cycles, is over a drastically larger search space compared to more restricted settings such as Bayesian network structure learning. The first *exact* approach to this general setting recently developed by Hyttinen *et al.* [2014] is able find guaranteed-optimal causal structures in terms of weighted (in)dependence constraints, and exhibits higher accuracy [Hyttinen *et al.*, 2014; Magliacane *et al.*, 2016; Borboudakis and Tsamardinos, 2016]. The approach is based on applying MaxSAT optimization solvers—or, in this case equivalently, answer set programming [Hyttinen *et al.*, 2014]. However, developing causal discovery algorithms with better running time performance without trading off accuracy for this general setting is a major challenge.

We take on this challenge by proposing several algorithmic techniques for speeding up the approach of Hyttinen *et al.* [2014], who apply a MaxSAT solver as a black box on their encoding of the causal discovery task. This leaves the actual search for a solution to an off-the-shelf MaxSAT solver that has no domain-specific knowledge on the problem domain in question. Instead, we show how to integrate domain-knowledge into a state-of-the-art MaxSAT solver for expediting the search for causal structures. Specifically, we (i) identify a number of generic *cores* describing small sets of conflicting constraints, (ii) develop an *incremental core extraction technique*, (iii) use *dynamic partial encodings* during search without influencing the optimality of found causal structures, and (iv) propose a problem-specific way based on *linear programming relaxations for hardening constraints*. The resulting solver, *Dseptor*, outperforms the approach of Hyttinen *et al.* [2014] in terms of running times on both simulated and real-world data. An implementation and an online appendix are available at the authors' homepages.

2 Exact Constraint-based Causal Discovery

We focus on the problem setting and optimization problem formulation of [Hyttinen *et al.*, 2014]. We consider the class $\mathcal{G}$ of *causal graphs* $G = (V, E)$ with set of nodes V, where the edge relation E is composed of directed causal edges and (symmetric) bi-directed edges (see Figure 1 for an example). Bi-directed edges $X \leftrightarrow Y$ canonically represent *latent confounders*, e.g., structures $X \leftarrow L \rightarrow Y$, where $L \notin V$ is an unmeasured common cause of two observed variables X and Y. A *walk* between X and Y is a sequence of consecutive edges in the graph (allowing for repeated edges and nodes). A node is a *collider* on a walk if both its adjacent edges on the walk point into the node. A walk in graph G is *d-connecting* w.r.t. a conditioning set $S \subseteq V \setminus \{X, Y\}$ if every collider on the walk is in S and no other nodes on the walk are in S. Two nodes are d-connected given a conditioning set S if there is at least one d-connecting walk between them; otherwise they are d-separated. (This is equivalent to Pearl's standard definition [Studený, 1998].) In Figure 1, X and W are d-connected given Y through $X \leftrightarrow Z \rightarrow Y \leftarrow Z \leftarrow W$. Nodes Y and Q are d-separated given W as all walks between them violate the d-connection criterion at node X.

Under the commonly used causal Markov and faithfulness assumptions [Spirtes *et al.*, 2000], statistical dependence becomes equivalent to (a type of) reachability in the graph: two random variables are statistically dependent conditional on a set of variables S iff they are d-connected given S in the generating causal structure G.[1] That is, given enough samples from a model with structure in Figure 1, we would expect to find X statistically dependent on W given Y, and Y statistically independent of Q given W. Thus, throughout the rest of the paper we use $X \perp\!\!\!\perp Y|S$ ($X \not\perp\!\!\!\perp Y|S$) to denote statistical independence (dependence) and d-separation (d-connection).

The aim of causal discovery is to recover as many properties of G_t as possible from the data, which we take to be an i.i.d. dataset sampled from a passive observational distribution that is Markov and faithful to an underlying "true" causal graph G_t. The (in)dependence constraints K are obtained by running statistical independence tests on the data. Since the tests produce both false positives and false negatives when run on finite sample data, exact constraint-based causal discovery can be viewed as the following abstract constrained optimization problem [Hyttinen *et al.*, 2014].

INPUT: A set K of conditional (in)dependence constraints over V, and a non-negative weight (cost) $w(k)$ for each $k \in K$.

TASK: Find a causal graph $G^* = (V, E^*)$ s.t.

$$G^* \in \arg\min_{G \in \mathcal{G}} \sum_{k \in K \,:\, G \not\models k} w(k).$$

In words, our goal is to find a single representative graph G^* that minimizes the sum of the weights of the given conditional independence and dependence constraints *not* implied by G^*. The constraints can be weighted according to their reliability, and conflicts among the constraints are well-resolved when the sum of the weights of the constraints not satisfied by the output graph are minimized. Given a representative, properties of its equivalence class can be examined in several ways, see e.g. [Hyttinen *et al.*, 2013].

[1] Spirtes [1995] shows this for the linear cyclic case. Pearl and Dechter [1996] give a proof for the discrete cyclic case, Neil [2000] points out the required additional assumptions. For the acyclic case, see e.g. [Spirtes *et al.*, 2000].

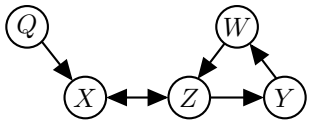

Figure 1: Example graph.

Weights for (in)dependence constraints can be obtained in many ways. Claessen and Heskes [2012] calculate Bayesian probabilities by marginalizing over Bayesian network structures. Ways to turn frequentist p-values into reliability weights have recently been proposed [Magliacane *et al.*, 2016; Triantafillou and Tsamardinos, 2015]. While the algorithms we develop are independent of the choice of weights, we obtain weights from local Bayesian model selection [Hyttinen *et al.*, 2014; Margaritis and Bromberg, 2009] for our experiments.

Note that the optimization problem is computationally challenging. For obtaining good accuracy, a large number of (in)dependence constraints K are needed; we use *all* testable (in)dependence constraints ($\binom{n}{2}2^{n-2}$ for n nodes). The d-separation condition for a solution satisfying a particular (in)dependence constraint is also quite intricate. On the other hand, this separation condition can be relatively naturally encoded declaratively as Boolean constraints. We give here an intuitive overview of the encoding of Hyttinen *et al.* [2014] in terms of MaxSAT to the extent necessary for the rest of this paper. In MaxSAT, problems are encoded using propositional clauses, using hard clauses (which have to be satisfied by all solutions) and weighted soft clauses (which, if not satisfied, incur a cost on a solution equal to their weights). In [Hyttinen *et al.*, 2014] each (in)dependence constraint $k \in K$ is encoded as a unit soft clause over a distinct Boolean variable representing k with weight $w(k)$. Additional Boolean variables are used for representing the solutions searched over, i.e., the edge relation of causal graphs. The d-connecting walks are encoded as hard clauses, linking the edge relation with the (in)dependence constraints.

3 Implicit Hitting Set Approach

We build on the encoding of Hyttinen *et al.* [2014] represented as MaxSAT [Berg *et al.*, 2015], integrating several domain-specific algoritmic improvements into the open-source MaxSAT solver LMHS [Saikko *et al.*, 2016] to further improve the running times of the approach. LMHS implements the generic *implicit hitting set approach* [Moreno-Centeno and Karp, 2013] for MaxSAT [Davies and Bacchus, 2013]. Analogously to MaxSAT, the causal discovery instances (D, K, w) considered here consist of a (hard) d-separability condition D, a set of (soft) (in)dependence constraints K, and a weight function w which characterizes the reliability of constraints in K.

In this context, the implicit hitting set approach of LMHS

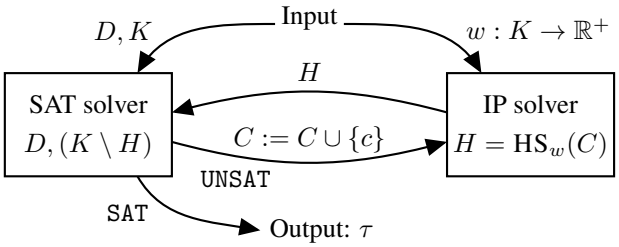

Figure 2: Implicit hitting set approach.

(see Figure 2) works via an alternating sequence of calls to a Boolean satisfiability (SAT) solver and an integer programming (IP) solver. Each is used for a task it is well-suited for: the SAT solver for proving unsatisfiability, and the IP solver for efficiently handling a weighted objective function.

In detail, the SAT solver is used to accumulate a set C of *(unsatisfiable) cores* of the instance, i.e., subsets of constraints $c \subset K$ which (together with D) are not satisfied by any causal graph. A core of an unsatisfiable instance in readily provided by modern SAT solvers without additional overhead once unsatisfiability is established. After each core extraction step, an IP solver is called on C to optimally solve the minimum-cost hitting set problem $\mathrm{HS}_w(C)$ over the cores in C with weights given by w, i.e., to find a set $H \subseteq \bigcup C$ minimizing $\sum_{x \in H} w(x)$ subject to $H \cap c \neq \emptyset$ for each $c \in C$, or in words, a minimum-cost set of constraints which intersects with each of the currently accumulated cores.

After a hitting set H is computed, the constraints in H are removed from the next SAT solver call. Concretely, this next call checks the satisfiability of a CNF formula consisting of the encoding of D and the encoding of the constraints of K not in H. Once the SAT solver finds a solution τ, i.e., finds a causal graph that satisfies $K \setminus H$ and D, the solution given by the SAT solver is guaranteed to be an optimal solution, i.e., the optimal cost is $\sum_{x \in H} w(x)$. The intuition here, as shown for MaxSAT by Davies and Bacchus [2011], is that the optimal causal graphs of (D, K, w) all satisfy $K \setminus H$ and D for some minimum-cost hitting set H over *all* unsatisfiable cores of the instance. In practice, such an H can be (and typically is) found without having to accumulate *every* unsatisfiable core. Further, the solution to each min-cost hitting set instance solved during search provides a lower bound on the cost of optimal causal graphs.

Example. Consider the following unit-weighted constraints:

$$K = \{X \not\perp\!\!\!\perp Z|W;\ Y \not\perp\!\!\!\perp Z|W;\ X \perp\!\!\!\perp Y|W;\ X \perp\!\!\!\perp Y|Z, W;\ X \not\perp\!\!\!\perp Z|Y, W;\ Y \not\perp\!\!\!\perp Z|X, W;\ X \not\perp\!\!\!\perp Y|W, Q;\ Y \perp\!\!\!\perp Q|W\}.$$

The implicit hitting set approach starts by checking whether all of the constraints are simultaneously satisfiable, i.e., whether some causal graph satisfies K and D. No such graph exists, so the SAT solver will return an unsatisfiable core of the instance, for example, $c_1 = \{X \not\perp\!\!\!\perp Z|W;\ Y \not\perp\!\!\!\perp Z|W;\ X \perp\!\!\!\perp Y|W;\ X \perp\!\!\!\perp Y|Z, W\}$. Thus, at least one of the four constraints has to be an error in the statistical tests (under the used assumptions). The search proceeds by solving the minimum-cost hitting set problem over this single core using an IP solver. One solution is $X \perp\!\!\!\perp Y|Z, W$, which gives a *lower bound* of 1 for the cost of optimal solutions. The search then iterates by checking whether a causal graph satisfies constraints other than $X \perp\!\!\!\perp Y|Z, W$. Again no such graph exists, and we obtain a core, say $c_2 = \{X \perp\!\!\!\perp Y|W;\ X \not\perp\!\!\!\perp Y|W, Q;\ Y \perp\!\!\!\perp Q|W\}$. The IP solver can then provide the min-cost hitting set $\{X \perp\!\!\!\perp Y|W\}$ of $\{c_1, c_2\}$. We then check for a graph in which all constraints other than $X \perp\!\!\!\perp Y|W$ are satisfied. Now the SAT solver returns a solution, which can be interpreted as the graph in Figure 1. The cost of this graph is 1, since only one constraint, $X \perp\!\!\!\perp Y|W$, with unit weight is violated in the graph, and the search terminates. Optimality is proven without considering the core $c_3 = \{X \not\perp\!\!\!\perp Z|Y, W;\ Y \not\perp\!\!\!\perp Z|X, W;\ X \perp\!\!\!\perp Y|W;\ X \perp\!\!\!\perp Y|Z, W\}$.

Table 1: Common domain-specific core types.

(i) $\{X \perp\!\!\!\perp Z|S;\ X \perp\!\!\!\perp Y|S;\ X \not\perp\!\!\!\perp Z|Y, S\}$

(ii) $\{X \not\perp\!\!\!\perp Z|S;\ Y \not\perp\!\!\!\perp Z|S;\ X \perp\!\!\!\perp Y|S;\ X \perp\!\!\!\perp Y|Z, S\}$

(iii) $\{X \not\perp\!\!\!\perp Z|Y, S;\ Y \not\perp\!\!\!\perp Z|X, S;\ X \perp\!\!\!\perp Y|S;\ X \perp\!\!\!\perp Y|Z, S\}$

(iv) $\{Y \not\perp\!\!\!\perp Z|S;\ X \not\perp\!\!\!\perp Z|S;\ Z \perp\!\!\!\perp W|X, Y, S;\ X \perp\!\!\!\perp Y|Z, S;\ X \perp\!\!\!\perp Y|W, S\}$

(v) $\{Y \not\perp\!\!\!\perp Z|S;\ X \not\perp\!\!\!\perp Z|S;\ Z \perp\!\!\!\perp W|Y, S;\ X \perp\!\!\!\perp Y|Z, S;\ X \perp\!\!\!\perp Y|W, S\}$

(vi) $\{X \not\perp\!\!\!\perp Y|Z, S;\ Y \not\perp\!\!\!\perp Z|X, W, S;\ W \not\perp\!\!\!\perp Y|Z, S;\ W \perp\!\!\!\perp X|Y, Z, S;\ X \perp\!\!\!\perp Z|W, S\}$

(vii) $\{X \not\perp\!\!\!\perp Y|Z, S;\ Y \not\perp\!\!\!\perp Z|X, W, S;\ W \not\perp\!\!\!\perp Y|S;\ W \perp\!\!\!\perp X|Y, S;\ X \perp\!\!\!\perp Z|W, S\}$

4 Domain-Specific Cores

As the first main contribution of this work, we identify several low-cardinality cores commonly encountered in causal discovery instances. Identifying general forms (or patterns) of such cores can be very beneficial for speeding up LMHS: we can find a number of cores by a simple pattern search through the constraints K (in negligible time), instead of making potentially very time-consuming SAT solver calls for each core. We can thereby use critical domain-specific knowledge in the LMHS search simply by adding the found domain specific cores directly to the core set maintained by LMHS.

During search, low-cardinality cores especially constrain the hitting set problems (and considerably improve the lower bound); however, their extraction can be time-consuming with a SAT solver. Thus, we ran plain LMHS on the encoding of Hyttinen *et al.* [2014] to search for examples of small cores. We then manually generalized them and proved their validity. Table 1 lists core types over at most four nodes, these core types are valid for any set S disjoint from the nodes mentioned. The induced cores were enough for solving all instances up to four nodes in our simulations, i.e., no further cores needed to be extracted using a SAT solver. The following theorem establishes that the core types (i)–(vii) in Table 1 are *subset-minimal* cores for causal discovery instances.

Theorem 1 *Each of the sets of (in)dependence constraints*

(i)–(vii) in Table 1 is a subset-minimal core: for each of the cores (i)–(vii), it holds that 1) there is no causal graph in $\mathcal{G}$ that satisfies all of the (in)dependence constraints in the core (assuming faithfulness), and 2) the constraints in every proper subset of the core are satisfied by a causal graph.

For example, the intuition behind core (i) is that if conditioning on node Y induces a dependence (d-connection) between X and Z (1st vs 3rd constraint), Y would have to be dependent on X (violating the 2nd constraint). Minimality of core (i) is verified by noting that any two constraints can be trivially simultaneously satisfied.

5 Incremental Core Extraction through Dynamic Partial Encoding

If we adhere exactly to the core loop shown in Figure 2, all constraints in $K \setminus H$ are input at once to the SAT solver in a single SAT call for extracting a core. Here we propose instead finding a core by incrementally adding constraints from $K \setminus H$ one by one until the SAT solver reports unsatisfiability and yields a core; Alg. 1 outlines this approach as FINDCOREINCREMENTAL. This technique has several potential benefits (as also previously pointed out in [Ansótegui *et al.*, 2016] for the related so-called stratified approach in the context of pure SAT-based MaxSAT solvers). (i) We obtain a solution τ from each SAT solver call until the call on which the solver reports unsatisfiability. Each of the satisfiable calls thus gives an upper bound which can improve the currently known best upper bound U; improved upper bounds are important for the bounds-based constraint hardening described in Sect. 6 to have an impact. (ii) The core c obtained is minimal w.r.t. the order in which the soft constraints are added. This can result in smaller cores, which makes the core minimization procedures implemented within LMHS run faster. (iii) Many of the—compared to LMHS, additional—SAT solver calls on D, K' are satisfiable and made on a relatively small set of constraints (K'); hence many of these calls can be expected to be relatively fast. Also, a number of SAT solver calls can be skipped, since the solution obtained during the preceding solver call often already satisfies the next constraint to be added.

Incremental core extraction allows for using domain-specific heuristics for the order in which constraints are added. For example, one can order by decreasing weights, following heuristics from inexact causal discovery algorithms [Triantafillou and Tsamardinos, 2015; Borboudakis and Tsamardinos, 2016] and MaxSAT [Ansótegui *et al.*, 2016], or by increasing conditioning set size, inspired by the FCI and PC algorithms [Spirtes *et al.*, 2000]. However, these heuristics tend to consider a relatively high number of constraints before finding a core. We found that adding constraints one by one in a random order works well here. Intuitively, a random order diversifies the search for cores to different subsets of (in)dependence constraints, which can be very beneficial for proving improved lower bounds.

Dynamic Partial Encoding. We also employ an encoding technique which allows for switching off parts of the (in total large) d-separation encoding that are redundant w.r.t. the current set K' of (in)dependence constraints considered in single

```
1: procedure FINDCOREINCREMENTAL(H, U)
2:     K' = ∅
3:     while K' ≠ K \ H do
4:         Add a new element k from K \ H to K'.
5:         If k is satisfied by previous solution τ: continue.
6:         (R, τ, c) = SATSOLVER(D, K')
7:         If R = SAT: update upper bound U, continue.
8:         If R = UNSAT: return core c.
9:     Return SAT.
```

Algorithm 1: Incremental core extraction in Dseptor.

SAT solver call. While the full MaxSAT encoding of a causal discovery instance is input initially to LMHS, by definition each of the SAT solver calls take into account only a subset of the (in)dependence constraints: the constraints in the latest hitting set H are disregarded in the next SAT solver call. The incremental core extraction technique further decreases the number of enforced (in)dependence constraints per SAT solver call. However, the hard clauses imposing d-separation criterion related to a specific (in)dependence constraint remain active in the SAT solver—although the actual independence is not used or considered at all. The SAT solver may not always realize that such hard clauses are redundant. In hope of making the job of the SAT solver easier, we propose a way of dynamically switching off hard clauses unrelated to a given set of (in)dependence constraints.

Consider Figure 3, where the two triangles rooted at the (in)dependence constraints k_i and k_j, respectively, represent the set of hard clauses that are part of the d-separation encoding for k_i and k_j, i.e., the clauses in the *cone of influence* of the respective constraints. In our case, this structure is directly given by the encoding proposed in [Hyttinen *et al.*, 2014]. Firstly, we add the logical implication $\neg b_i \rightarrow k_i$ per each (in)dependence constraint k_i introducing a fresh *blocking variable* b_i. Secondly, for clauses in the non-shared part of the cone marked with b_i (similarly for b_j), we augment each clause with the blocking variable b_i: we replace each clause cl by logical implication $\neg b_i \rightarrow cl$. This means that $b_i = 0$ implies that the clauses have to be satisfied, and $b_i = 1$ relaxes (blocks) the clauses. For the shared part of the cones marked with b_{ij}, we again augment the clauses in that part with the blocking variable b_{ij}. We also add the logical implication $(\neg b_i \vee \neg b_j) \rightarrow \neg b_{ij}$ as a hard constraint. Thus $b_i = 0$ (or $b_j = 0$) enforces the clauses in the shared part as well. The idea here resembles the approach proposed for dealing with dependencies between blocking variables in other contexts [Lagniez and Biere, 2013]. Applying this strategy recursively on the d-separation encoding establishes that all clauses in the cone of influence of K' are switched on iff all (in)dependence constraints in K' are enforced to be satisfied during the incremental core extraction SAT calls.

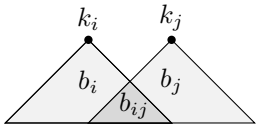

Figure 3: Blocking variables in dynamic partial encoding.

6 Bounds-based Constraint Hardening

During search, we use the current bounds for bounds-based hardening of (in)dependence constraints. Our domain-specific approach is based on the following well-known technique of *variable fixing* for IP solving in operations research literature [Danzig *et al.*, 1954; Crowder *et al.*, 1983].[2] By finding the min-cost hitting set over the current set of accumulated cores, conditioned on including a particular constraint k, we can obtain a conditional lower bound $L_{\neg k}$ for the space of solutions in which k is not satisfied. If $L_{\neg k}$ exceeds the current global upper bound U, k can be enforced as a hard constraint. Instead of solving the IP problem, a good bound $L_{\neg k}$ can be obtained via the corresponding LP relaxation which is faster to solve. For hardening constraints early, this technique benefits crucially from the improved bounds obtained by incremental core extraction. Each hardened constraint simplifies the subsequent SAT and IP calls, and also speeds up incremental core extraction as only constraints that have not yet been hardened are considered there.

As a domain-specific extension, we also use this scheme to harden Boolean variables representing the causal graph solution, i.e., the existence of individual edges in the graph. Specifically, if the solution graph includes *any* type of an edge between nodes X and Y, then it leaves all constraints K'' of the form $X \perp\!\!\!\perp Y|S$ unsatisfied. If there are many sets $S \subseteq V \setminus \{X, Y\}$ such that $X \perp\!\!\!\perp Y|S$, the sum of weights of such constraints can be large. Thus it makes sense to find the lower bound L'' conditional on the existence of an edge between nodes X and Y—we can obtain such L'' from the LP relaxation conditioned on including *each* constraint in K''. If L'' is greater than the current global upper bound U, no edges can then be present between X and Y in an optimal graph and thus we can w.l.o.g. assign all edge variables between X and Y to be false. (Note that, in contrast, the inexact PC algorithm infers the absence of edges between X and Y upon finding a *single* (possibly unreliable) independence $X \perp\!\!\!\perp Y|S$ [Spirtes *et al.*, 2000].) Inferring hard absences of edges directly simplifies the subsequent SAT solver calls by tightening the solution space. The subsequently extracted cores are thereby often smaller, which makes the subsequent hitting set IPs also simpler.

7 Experiments

We implemented *Dseptor* by extending the MaxSAT solver LMHS with the domain-specific techniques proposed in Sects. 4–6; We use MiniSAT [Eén and Sörensson, 2004] and IBM CPLEX as the internal SAT and IP solvers, respectively. We compare the performance of Dseptor to those of state-of-art MaxSAT solvers using the encoding of Hyttinen *et al.* [2014]. The experiments were run on 2.83-GHz Intel Xeon E5440 machines with 32-GB RAM and Debian GNU/Linux.

First we compare the performance of the solvers on synthetic data generated from 7-node cyclic (possibly) linear Gaussian models with correlated disturbances. The edges were sampled at random to obtain average degree 2, coefficients were from $\pm[0.2, 0.8]$. We used BIC-based model selection to obtain the weights for the independence constraints over $N = 1000$ samples. Figure 4 gives a comparison of Dseptor with the MaxSAT solvers LMHS [Saikko *et al.*, 2016], MaxHS [Davies and Bacchus, 2013], Maxino [Alviano *et al.*, 2015], MSCG [Morgado *et al.*, 2015], OpenWBO [Martins *et al.*, 2014], QMaxSAT [Koshimura *et al.*, 2012], and WPM3 [Ansótegui *et al.*, 2016], CPLEX run on a standard IP translation of MaxSAT [Davies and Bacchus, 2013; Ansótegui and Gabàs, 2013], and the ASP solver Clingo (using its default branch-and-bound search) [Gebser *et al.*, 2012] as used in [Hyttinen *et al.*, 2014]. Dseptor outperforms all of the other solvers, solving 96% of the instances under the timeout of 600 seconds. Maxino comes in second (81%) ahead of plain LMHS and QMaxSAT. Clingo and many other solvers are not competitive here. Figure 4 also shows that all techniques proposed in Sects. 4–6 improve the performance: turning off each individual technique in Dseptor increases solving times.

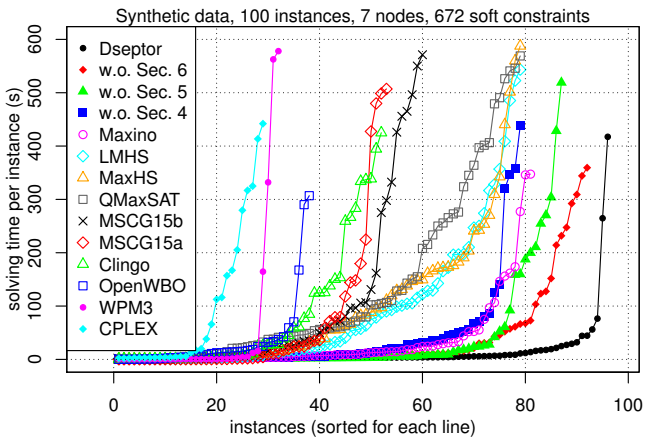

Figure 4: MaxSAT solver and Dseptor running times.

To examine the performances on more realistic sets of (in)dependence constraints, we looked at real-world datasets often used for benchmarking exact Bayesian network structure learning algorithms [Yuan and Malone, 2013; Bartlett and Cussens, 2017]. To comply with the current problem formulation, we considered suitable-sized (n) subsets of the variables in the datasets, the remaining variables becoming thus latent. We employed the BDEU score with equivalent sample size 10 to obtain constraint weights for this discrete data, and used a per-instance timeout of 7200 seconds. The results summarized in Figure 5, and detailed in Table 2 for Dseptor and its best competitors, LMHS (on which Dseptor is based), Maxino, and QMaxSAT, show that Dseptor clearly outperforms the competing approaches on these real-world benchmarks. Table 2 also gives the time taken by Dseptor to find an optimal graph (without proving its optimality yet) in parentheses. On a great majority of instances Dseptor finds an optimal solution very fast; the proof of optimality hence dominates the overall running times. This implies that Dseptor has very good anytime performance, being often able to give very good solutions in short time.

[2] Hardening using costs of residual formulas has also been used in pure SAT-based MaxSAT algorithms [Ansótegui *et al.*, 2013].

Table 2: Running times on real-world datasets.

			Running time (seconds)				
Dataset	n	N	LMHS	Maxino	QMaxSAT	Dseptor	
Adult	6	30162	37	41	41	**11**	(4)
Alarm	7	1000	606	3674	2017	**151**	(4)
Autos	8	159	TO	TO	TO	**2044**	(63)
Bands	6	277	**103**	212	291	3268	(1)
Epigenetics	7	72228	82	98	213	**28**	(23)
Flag	10	194	TO	TO	TO	**888**	(78)
Heart	10	212	TO	TO	6458	**881**	(75)
Hepatitis	10	126	TO	3725	TO	**392**	(72)
Horse.23	7	300	62	10	133	**4**	(0)
Horse	9	300	TO	1235	2878	**337**	(44)
Image	7	2310	3590	723	1267	**686**	(16)
Imports	6	205	**63**	182	243	TO	(7)
Letter	7	20000	2121	1295	1214	**693**	(419)
LungCancer	8	27	1350	1330	4232	**74**	(28)
Meta	7	528	754	350	338	**182**	(119)
Mushroom	7	1000	1026	**777**	1336	1120	(957)
Mushroom	7	8124	252	**205**	305	710	(188)
Parkinsons	6	195	22	12	34	**3**	(0)
Sensors	7	5456	157	**42**	148	82	(53)
Soybean	9	266	TO	990	860	**180**	(0)
Spectf	10	267	TO	3598	449	**261**	(1)
Statlog	7	752	834	1747	7107	**61**	(17)
SteelPlates	6	1941	22	**16**	24	64	(1)
Voting	7	435	252	138	217	**62**	(4)
Water	7	380	181	563	305	**94**	(15)
Wdbc	8	569	TO	TO	TO	**2833**	(2)
Wine	7	178	222	1285	2254	**164**	(3)
Zoo	6	101	29	34	55	**18**	(1)
alarm	9	10000	TO	1382	1230	**26**	(0)
alarm	9	1000	TO	1354	TO	**72**	(14)
alarm	7	100	271	601	301	**68**	(2)
asia	8	10000	TO	TO	TO	**873**	(22)
asia	7	1000	591	1062	738	**92**	(29)
asia	10	100	104	13	21	**4**	(0)
carpo	9	10000	TO	2267	TO	**605**	(76)
carpo	8	1000	TO	TO	TO	**633**	(4)
carpo	8	100	6211	TO	5510	**1877**	(13)
Diabetes	8	10000	TO	TO	171	**8**	(0)
Diabetes	8	1000	TO	4714	373	**9**	(0)
Diabetes	8	100	157	43	33	**11**	(0)
hailfinder	8	10000	3539	1815	**784**	TO	(0)
hailfinder	9	1000	TO	551	1843	**97**	(28)
hailfinder	7	100	**854**	TO	3827	4399	(9)
insurance	8	10000	2255	944	1835	**48**	(4)
insurance	9	100	TO	TO	TO	**853**	(19)
Link	10	10000	TO	TO	TO	**637**	(24)
Link	10	1000	TO	TO	TO	**394**	(21)
Link	9	100	TO	TO	TO	**164**	(11)
Mildew	9	10000	TO	852	1950	**105**	(5)
Mildew	8	1000	2808	TO	6205	**49**	(8)
Mildew	6	100	25	208	164	**10**	(0)
Pigs	8	10000	1899	146	694	**17**	(1)
Pigs	8	1000	3128	2848	1561	**1026**	(60)
Pigs	8	100	3449	TO	4964	**47**	(3)
Water	7	10000	298	202	499	**48**	(0)
Water	10	1000	TO	TO	TO	**1034**	(245)
Water	10	100	TO	5983	TO	**484**	(26)

8 Related Work

There has been recent algorithmic work on causal discovery problems related to the one discussed in this paper. SAT solvers where first used for causal discovery in [Triantafillou *et al.*, 2010; Triantafillou and Tsamardinos, 2015]. Essentially, they greedily add constraints one at a time and check for satisfiability: if the solver returns UNSAT, the constraint is removed, and search is continued. Although this scales up considerably better than the exact counterparts, the accuracy is not as good as that of an exact method [Borboudakis and Tsamardinos, 2016]. Magliacane *et al.* [2016] proposed a relaxation of the optimization problem definition of Hyttinen *et al.* [2014], thereby obtaining better scalability for detecting some features of the graph without losing much accuracy compared to Hyttinen *et al.* [2014]. However, in comparison to the present paper, the most noticeably difference shared by Triantafillou *et al.* [2010], Triantafillou and Tsamardinos [2015], and Magliacane *et al.* [2016] is that (i) they consider a more restricted search space (and hence a different encoding), assuming acyclicity, and (ii) their approaches do not produce guaranteed-optimal solutions in terms of the theoretically appealing objective function considered here. Note also that in our approach acyclicity can be enforced when wanted. Finally, just as Hyttinen *et al.* [2014], we can incorporate background knowledge and straightforwardly generalize to multiple (experimental) data sets.

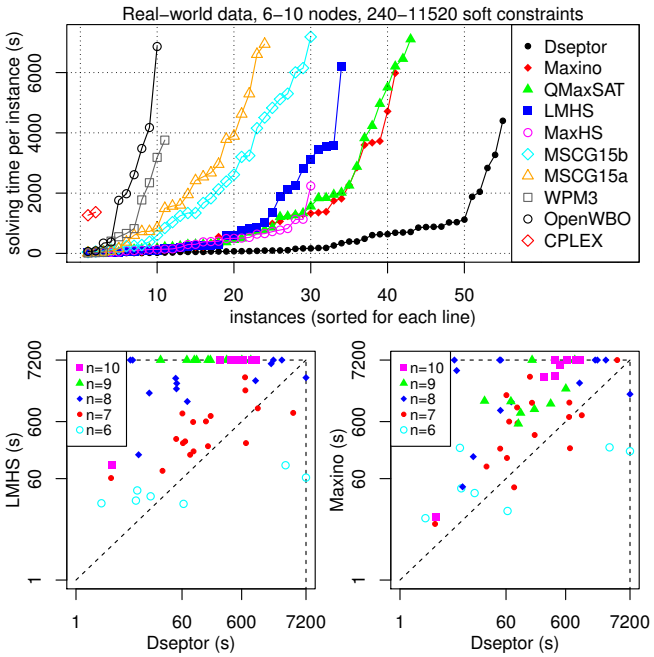

Figure 5: Running time comparison on real-world data.

9 Conclusions

We proposed several domain-specific techniques for improving a recent Boolean optimization approach to exact constraint-based causal discovery over a very general search space. We showed that the resulting causal discovery solver Dseptor has markedly better the runtime performance than other solvers on both synthetic and real-world datasets. We conjecture that enumerating additional (larger but still relatively small) domain-specific minimal cores could improve the runtime performance further.

Acknowledgements

We thank Fahiem Bacchus for pointing out bounds-based hardening via LPs for Section 6. This work was financially supported by Academy of Finland through grants 251170 (COIN), 276412, 284591, and 295673; and DoCS Doctoral School in Computer Science and Research Funds of the University of Helsinki.

References

[Alviano *et al.*, 2015] M. Alviano, C. Dodaro, and F. Ricca. A MaxSAT algorithm using cardinality constraints of bounded size. In *IJCAI*. AAAI Press, 2015.

[Ansótegui and Gabàs, 2013] C. Ansótegui and J. Gabàs. Solving (weighted) partial MaxSAT with ILP. In *CPAIOR*, volume 7874 of *LNCS*, pages 403–409. Springer, 2013.

[Ansótegui *et al.*, 2013] C. Ansótegui, M.L. Bonet, J. Gabàs, and J. Levy. Improving WPM2 for (weighted) partial MaxSAT. In *CP*, volume 8124 of *LNCS*, pages 117–132. Springer, 2013.

[Ansótegui *et al.*, 2016] C. Ansótegui, J. Gabàs, and J. Levy. Exploiting subproblem optimization in SAT-based MaxSAT algorithms. *J. Heuristics*, 22(1):1–53, 2016.

[Bartlett and Cussens, 2017] M. Bartlett and J. Cussens. Integer linear programming for the Bayesian network structure learning problem. *Artif. Intell.*, 244:258–271, 2017.

[Berg *et al.*, 2014] J. Berg, M. Järvisalo, and B. Malone. Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In *AISTATS*, volume 33 of *JMLR W&CP*, pages 86–95. JMLR, 2014.

[Berg *et al.*, 2015] J. Berg, A. Hyttinen, and M. Järvisalo. Applications of MaxSAT in data analysis. In *PoS*, 2015.

[Borboudakis and Tsamardinos, 2016] G. Borboudakis and I. Tsamardinos. Towards robust and versatile causal discovery for business applications. In *KDD*. ACM, 2016.

[Claassen and Heskes, 2012] T. Claassen and T. Heskes. A Bayesian approach to constraint based causal inference. In *UAI*, pages 207–216. AUAI Press, 2012.

[Crowder *et al.*, 1983] H. Crowder, E.L. Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Oper. Res.*, 31(5):803–834, 1983.

[Danzig *et al.*, 1954] G.B. Danzig, D.R. Fulkerson, and S.M. Johnson. Solution of a large-scale traveling-salesman problem. *Oper. Res.*, 2:393–410, 1954.

[Davies and Bacchus, 2011] J. Davies and F. Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In *CP*, volume 6876 of *LNCS*, pages 225–239. Springer, 2011.

[Davies and Bacchus, 2013] J. Davies and F. Bacchus. Exploiting the power of MIP solvers in MAXSAT. In *SAT*, volume 7962 of *LNCS*, pages 166–181. Springer, 2013.

[Eén and Sörensson, 2004] N. Eén and N. Sörensson. An extensible SAT-solver. In *SAT 2003*, volume 2919 of *LNCS*, pages 502–518. Springer, 2004.

[Gebser *et al.*, 2012] M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving: From theory to practice. *Artif. Intell.*, 187:52–89, 2012.

[Hyttinen *et al.*, 2013] A. Hyttinen, P. O. Hoyer, F. Eberhardt, and M. Järvisalo. Discovering cyclic causal models with latent variables: A general SAT-based procedure. In *UAI*, pages 301–310. AUAI Press, 2013.

[Hyttinen *et al.*, 2014] A. Hyttinen, F. Eberhardt, and M. Järvisalo. Constraint-based causal discovery: Conflict resolution with answer set programming. In *UAI*, pages 340–349. AUAI Press, 2014.

[Koshimura *et al.*, 2012] M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa. QMaxSAT: A partial Max-SAT solver. *JSAT*, 8(1/2):95–100, 2012.

[Lagniez and Biere, 2013] J.-M. Lagniez and A. Biere. Factoring out assumptions to speed up MUS extraction. In *SAT*, volume 7962 of *LNCS*, pages 276–292. Springer, 2013.

[Magliacane *et al.*, 2016] S. Magliacane, T. Claassen, and J.M. Mooij. Ancestral causal inference. In *NIPS*, pages 4466–4474, 2016.

[Malone *et al.*, 2015] B. Malone, M. Järvisalo, and P. Myllymäki. Impact of learning strategies on the quality of Bayesian networks: An empirical evaluation. In *UAI*, pages 562–571. AUAI Press, 2015.

[Margaritis and Bromberg, 2009] D. Margaritis and F. Bromberg. Efficient Markov network discovery using particle filters. *Comput. Intell.*, 25(4):367–394, 2009.

[Martins *et al.*, 2014] R. Martins, V.M. Manquinho, and I. Lynce. Open-WBO: A modular MaxSAT solver. In *SAT*, volume 8561 of *LNCS*, pages 438–445. Springer, 2014.

[Moreno-Centeno and Karp, 2013] E. Moreno-Centeno and R.M. Karp. The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment. *Oper. Res.*, 61(2):453–468, 2013.

[Morgado *et al.*, 2015] A. Morgado, A. Ignatiev, and J. Marques-Silva. MSCG: Robust core-guided MaxSAT solving. *JSAT*, 9:129–134, 2015.

[Neal, 2000] R. Neal. On deducing conditional independence from d-separation in causal graphs with feedback. *J. Artif. Intell. Res.*, 12:87–91, 2000.

[Pearl and Dechter, 1996] J. Pearl and R. Dechter. Identifying independencies in causal graphs with feedback. In *UAI*, pages 420–426. Morgan Kaufmann, 1996.

[Pearl, 2000] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[Richardson, 1996] T. Richardson. A discovery algorithm for directed cyclic graphs. In *UAI*, pages 454–461. Morgan Kaufmann, 1996.

[Saikko *et al.*, 2016] P. Saikko, J. Berg, and M. Järvisalo. LMHS: A SAT-IP hybrid MaxSAT solver. In *SAT*, volume 9710 of *LNCS*, pages 539–546. Springer, 2016.

[Spirtes *et al.*, 2000] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, 2000.

[Spirtes, 1995] P. Spirtes. Directed cyclic graphical representation of feedback models. In *UAI*, pages 491–498. Morgan Kaufmann, 1995.

[Studený, 1998] M. Studený. Bayesian networks from the point of view of chain graphs. In *UAI*, pages 496–503. Morgan Kaufmann, 1998.

[Triantafillou and Tsamardinos, 2015] S. Triantafillou and I. Tsamardinos. Constraint-based causal discovery from multiple interventions over overlapping variable sets. *J. Mach. Learn. Res.*, 16:2147–2205, 2015.

[Triantafillou *et al.*, 2010] S. Triantafillou, I. Tsamardinos, and I. G. Tollis. Learning causal structure from overlapping variable sets. In *AISTATS*. JMLR, 2010.

[Yuan and Malone, 2013] C. Yuan and B. Malone. Learning optimal Bayesian networks: A shortest path perspective. *J. Artif. Intell. Res.*, 48:23–65, 2013.