Constraint Relaxation for Learning the Structure of Bayesian Networks

Andrew Fast and David Jensen
Department of Computer Science, University of Massachusetts Amherst
140 Governors Drive
Amherst, MA 010002

Abstract

This paper introduces constraint relaxation, a new strategy for learning the structure of Bayesian networks. Constraint relaxation identifies and "relaxes" possibly inaccurate independence constraints on the structure of the model. We describe a heuristic algorithm for constraint relaxation that combines greedy search in the space of undirected skeletons with edge orientation based on the constraints. This approach produces significant improvements in the structural accuracy of the learned models compared to four well-known structure learning algorithms in an empirical evaluation using data sampled from both real-world and randomly generated networks.

1 Introduction

Constraint-based algorithms for learning the structure of Bayesian networks (BNs) are designed to recover the underlying structure of the distribution that generated the training data. These algorithms utilize a series of conditional independence tests to identify constraints on the possible model structure. This is in contrast to *search-and-score* algorithms that search for model structures that maximize some scoring metric.

When applied to samples of reasonable size, the conditional independence tests used to identify the constraints are prone to both false positive (type I) and false negative (type II) errors (Spirtes et al., 2000). In the context of Bayesian networks, false positive errors indicate an edge in the learned model that does not appear in the true model, and false negative errors indicate an edge in the true model that does not appear in learned model. These errors result in inaccurate constraints and corresponding reductions in the accuracy of the learned structure.

In this paper, we introduce *constraint relaxation*, a new strategy for improving the structural accuracy of Bayesian network structure learning. The goal of constraint relaxation is to identify potentially inaccurate constraints and "relax" or ignore those constraints when orienting edges. Constraint relaxation searches the space of all possible directed acyclic graphs (DAGs), unlike previous constraint-based and hybrid approaches which only search a subset of the space. By using the complete search space, this approach can address both false negative and false positive errors. This contrasts with previous attempts at improving structural accuracy, which have focused primarily on reducing false positive errors (Li and Wang, 2009; Tsamardinos and Brown, 2008). In addition, constraint relaxation is a general strategy that can be easily combined with these existing approaches.

In Section 3, we describe the RELAX algorithm for constraint relaxation. This algorithm combines a greedy search in the space of skeletons with a novel edge orientation algorithm based on the constraints. Despite considering a larger search space, empirical results on data sampled from both randomly-generated BNs and BNs drawn from real decision support problems demonstrate that constraint relaxation produces significant improvements in structural accuracy over existing structure learning algorithms.

2 Background and Related Work

Structure learning algorithms identify the presence and orientation of edges in a Bayesian network from data. A Bayesian network is a directed, acyclic, graphical model representing a joint probability distribution over a set of variables (Pearl, 1988). The structure of the graph encodes probabilistic independencies among the variables. In this work, we focus on constraint-based algorithms for structure learning. These algorithms typically operate in two phases: *skeleton identification* and *edge orientation*.

2.1 Skeleton Identification

Constraint-based algorithms utilize a series of conditional independence tests to learn a set of constraints on the final model structure. These constraints are typically represented in two parts. The first part of the constraints is an *undirected skeleton* indicating the presence and absence of edges in the model but not their final orientation. The process of learning constraints is called *skeleton identification*. The constraints also contain separating sets, commonly called *sepsets*. For each pair of variables that were found to be independent, the sepset for that pair contains the (possibly empty) set of conditioning variables that were sufficient for proving independence.

Skeleton identification algorithms can differ in both the algorithm used to order the statistical tests and the type of statistical test used. We use the first and most widely used ordering algorithm drawn from steps A and B of the PC algorithm (Spirtes et al., 2000). This is called Fast Adjacency Search (FAS) in the TETRAD IV[1] package. Other ordering algorithms include Max-Min Parents Children (Tsamardinos et al., 2006) and Three-Phase Dependency Analysis (Cheng et al., 2002).

For discrete data, the FAS algorithm uses either the G^2 test or the χ^2 test to determine independence. These tests are known to have low statistical power when run with small samples and large conditioning sets, resulting in errors in the constraints (Spirtes et al., 2000). Since the performance of constraint-based algorithms depends on the quality of the constraints, many attempts have been made to understand and improve the accuracy of the statistical tests used to determine independence. Alternative approaches include Bayesian tests of independence (Dash and Druzdzel, 2003), tests of mutual information (Cheng et al., 2002), and using approaches based on False Discovery Rate (FDR) (Li and Wang, 2009; Tsamardinos and Brown, 2008).

2.2 Edge Orientation

Once constraints have been learned, constraint-based algorithms utilize the skeleton and sepsets to orient the edges appearing in the skeleton. When causal sufficiency can be assumed, an algorithm consisting of Steps C and D in the PC algorithm, which we call PC-Edge, has been proven sound and complete in the sample limit (Spirtes et al., 2000). A corresponding algorithm, called FCI, is sound and complete without the assumption of causal sufficiency and in the presence of selection bias, though we do not address that situation in this work (Spirtes et al., 1999, 2000).

[1] `http://www.phil.cmu.edu/projects/tetrad/`

PC-Edge uses the sepsets to orient colliders in the skeleton. A collider is a structure $X \rightarrow Z \leftarrow Y$ such that $X \perp\!\!\!\perp Y$ and $Z \notin Sepset(X, Y)$. These *collider constraints* do not consider the full sepset. This procedure, however, is not stable as small changes in the constraints can lead to large changes in the final structure (Spirtes et al., 2000). Badea (2004) and Steck and Tresp (1996) both describe *post hoc* approaches for revising the learned structure based on the (in)consistency of the constraints.

There are two alternatives to PC-Edge that use a greedy search to produce a final edge orientation. Hybrid algorithms use the skeleton to constrain a greedy search for the final structure (Tsamardinos et al., 2006). Only edges appearing in the skeleton are considered for inclusion in the final model. Abellan et al. (2006) use unconstrained greedy search to refine the model produced by the PC algorithm. This refinement does not consider either the skeleton or the sepsets in determining the final orientation.

2.3 Evaluation of Structural Accuracy

For evaluation purposes, structural accuracy of learned networks can be measured with a variety of different metrics that compare the pattern (essential graph) of both the learned and true models. This requires the structure of the true model to be known *a priori*. The first metric is the accuracy (percent of edges correct) of edges in the model (Badea, 2004; Bromberg and Margaritis, 2009; Spirtes et al., 1999). A closely related metric is the precision and recall of causal structures (Mani and Cooper, 2006). We use precision and recall of compelled edges, where compelled precision is defined as the percentage of compelled edges that are correct in the learned model. A compelled edge is an edge that has the same orientation in every member of the equivalence class of the learned model (Chickering, 2002). For simplicity, we will report a single number, the compelled F-measure, which is the harmonic mean of compelled precision and compelled recall (Croft et al., 2009) and is a general purpose metric of the causal structure.

An alternative metric is structural Hamming distance (SHD) based on the raw counts of errors in the learned model (Tsamardinos et al., 2006). The SHD of a model is a type of graph edit distance and is equal to the number of edge deviations between the model and the true model. It is often expedient to consider decompositions of the SHD, particularly into skeleton errors (false positive and false negative errors) and orientation errors (errors of edge direction). We will also use the number of true positive edges (number of correct edges) for evaluation.

3 Constraint Relaxation

Constraint relaxation is a new strategy for improving the accuracy of constraint-based structure learning. In the following section, we provide additional motivation for constraint relaxation and describe the first algorithm for performing constraint relaxation.

3.1 Motivation

Despite recent advances in the accuracy of the statistical tests used in skeleton identification, learned constraints are still likely to contain errors. In particular approaches bounding the false discovery rate of the skeleton identification procedure provide no bound on the type II (false negative) error rate (Tsamardinos and Brown, 2008). Therefore, alternative approaches, such as constraint relaxation, are still necessary to identify and address possible false positive and false negative errors appearing in the model.

Orienting edges based on constraints is sound in the sample limit and permits model structures that contain a larger number of parameters than would be learned using a penalized likelihood score such as BDeu. Therefore, constraint relaxation is designed to combine the power of constraint-based edge orientation with the relaxation of both independence and dependence constraints.

Our conception of constraint relaxation is inspired by but distinct from constraint relaxation in the partial constraint satisfaction literature (Freuder and Wallace, 1992). In this scenario we are not relaxing constraints to achieve a consistent solution but removing constraints from the knowledge base to improve performance. Additionally, if we believe that some of the constraints are incorrect, then constraint relaxation would be valuable even if there exists a structure which is consistent with the current constraints.

3.2 Greedy Relaxation Algorithm

Here we present a simple greedy algorithm for constraint relaxation, called RELAX (Algorithm 1). We show in Section 5 that this approach for relaxation leads to significant improvements in the structural accuracy of learned models. To learn the constraints, we use the FAS algorithm, but any skeleton algorithm that learns both an undirected skeleton and sepsets could be used instead. After the constraints have been learned, we use a local greedy search over possible relaxations of the constraints. Each time through the while loop, the algorithm chooses the single constraint which, if relaxed, leads to the largest improvement in the score. If the best relaxation produces a model with a higher score than the current best model, then the relaxation is applied to the constraints and the algorithm continues searching for additional relaxations. If no relaxation leads to an improvement, the algorithm halts with the current structure.

Algorithm 1 Constraint Relaxation

```
procedure RELAX(Data)
    C = LEARN-CONSTRAINTS(Data)
    B = ORIENT-EDGES(C)
    S = BDEU(B, Data)
    C* = C, B* = B, S* = S
    updated = True
    while updated=True do
        updated = False
        for c ∈ C do
            // Relax the constraint
            C' = C/c
            B' =ORIENT-EDGES(C')
            S' =BDEU(B', Data)
            if S' > S then
                B = B', S = S', C = C'
            end if
        end for
        if S > S* then
            B* = B, S* = S, C* = C
            updated = True
        end if
    end while
    return B*
end procedure
```

Since constraints directly correspond to the absence or presence of an edge, this procedure is a search over undirected skeletons. To relax a constraint we simply toggle the corresponding edge in the current network and update the sepset accordingly. If we toggle a constraint from dependence to independence, the sepset is set to the empty set; if we toggle from independence to dependence, the sepset can be ignored since the edge now exists. We then orient the edges based on the current formulation of constraints. Thus, we use an approach similar to the approach of Steck (2000); however, we choose an orientation at each step to maximize the number of constraints satisfied and not a likelihood score. As detailed in the following section, this approach cannot be used with existing edge orientation algorithms.

4 Edge Orientation as Constraint Optimization

Algorithm 1 requires that the ORIENT-EDGES procedure produce a model that can be scored using BDeu

or other penalized likelihood measure. Unfortunately, it is not possible to use PC-EDGE as the edge orientation algorithm. As currently defined, PC-EDGE often results in networks with bidirected edges. There is currently no known parameterization of these models for discrete data (Spirtes et al., 2000), precluding their use with our constraint relaxation framework.

Therefore, to make a greedy relaxation algorithm possible, we developed a novel edge orientation algorithm, called EDGE-OPT, as a replacement for PC-EDGE (See Algorithm 2). The EDGE-OPT algorithm uses a new formulation of constraints, based on a dependency model, that is sound in the sample limit yet permits search over possible structures. Like Bromberg and Margaritis (2009), our formulation of constraints uses the full separating set. In Section 4.3, we demonstrate that the EDGE-OPT performs no worse than the PC-EDGE algorithm in structural accuracy.

4.1 Constraints as a Dependency Model

Skeleton identification can be viewed as a process of recovering the dependency model M corresponding to the empirical distribution given by the training data. A *dependency model* (Pearl, 1988) is a set of assertions of the form $(X \perp\!\!\!\perp Y|Z)$ indicating that "X is independent of Y given Z." A DAG D is a *Bayesian network* of M (Pearl, 1988) if and only if every independence assertion in M corresponds to a valid d-separation relationship in D; that is, D is a minimal I-map of M. Therefore, if we let our constraints correspond exactly to the independence assertions in the learned dependency model M, then structure learning can be viewed as a search for a DAG D which satisfies all of the constraints. Note that in this formulation, dependence constraints are implied by the absence of an independence.

If we make the assumptions of Spirtes et al. (2000)—no latent variables, the distribution represented by the training data is faithful to a DAG, and the statistical decisions made from the data are correct (e.g., made in the sample limit)—then it follows directly from the definitions of a dependency model and a Bayesian network that our constraint formulation is sound. A DAG which satisfies all of the constraints will be equivalent to the structure of the generating distribution.

In practice, however, the set of learned independence constraints are rarely consistent and cannot be represented by a DAG. Given this situation, it is natural to view edge orientation, and structure learning in general, as a constraint optimization problem where the goal is to identify a DAG which maximizes the number of satisfied independence constraints.

4.2 Edge Orientation Algorithm

Given an undirected skeleton S, indicating the presence of edges in the model, we develop an edge orientation algorithm based on constraint optimization, called EDGE-OPT, with the goal of satisfying as many learned independence constraints as possible. We define an independence constraint $(X \perp\!\!\!\perp Y|Z)$ to be satisfied by a DAG D if Z is a minimal d-separator of X and Y in D. To check whether a constraint holds in D we use Algorithm 1 of Tian et al. (1998).

Ideally, we would like to use a complete search algorithm to find the best orientation of S. However, this is not feasible with reasonably sized networks, as the number of possible orientations grows exponentially with the number of edges appearing in S. Additionally, since d-separation constraints are defined in terms of paths, approaches from traditional constraint satisfaction are not applicable because the resulting constraint graph is fully connected.

Instead, we use greedy heuristic search over possible orientations of the skeleton S, which is fixed during edge orientation. Each state in the search process is a fully directed, acyclic graph D. The initial state is oriented by choosing a random ordering over the variables and orienting the edges to be consistent with that ordering. Each successor state is generated by an application of an original search operator called TOGGLE-COLLIDER. For every triple of variables in D, TOGGLE-COLLIDER either makes the triple into a collider (right to left in Figure 1) or breaks the existing collider in each of three possible ways (left to right in Figure 1). The successor states consist of the set of DAGs that differ from D by at least one collider. Note that changing one collider may introduce one or more additional colliders. Since the successors states contain the same skeleton but a different set of colliders from D, the successor states are not in the same equivalence class as D (though multiple successors may be in the same equivalence class) (Verma and Pearl, 1990). The size of the search space depends on the number of triples in S.

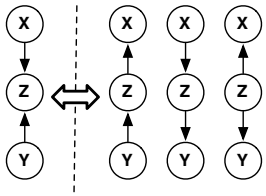

Figure 1: The TOGGLE-COLLIDER operator

To choose among possible successors, we count the number of constraints satisfied by each successor and

choose the successor that satisfies the maximum number of constraints. In the case of ties, we choose the successor with the higher BDeu score. This scoring scheme gives each constraint an implicit unit weight. We considered alternative weighting schemes such as using the statistical power of the test but found no improvement in the performance of our algorithm. In addition, we add a hard (infinite weight) acyclicity constraint to guarantee that the best successor is always a DAG. This is necessary for the overall constraint relaxation approach (see Section 3).

Algorithm 2 Edge Orientation via Constraint Optimization

```
procedure EDGE-OPT(Constraints C, k, numRestarts)
    // Get skeleton and dependency model.
    (S, M) = C
    D* = nil // Global best structure.
    for i = 1 to numRestarts do
        D = RANDOM-ORIENTATION(S)
        while True do
            N = SUCCESSORS(D)
            if |N| == 0 then
                break
            end if
            D' = BEST-SUCCESSOR(N)
            if SCORE(D') > SCORE(D*) then
                D* = D'
            end if
        end while
    end for
    return D*
end procedure

procedure SUCCESSORS(D)
    T = TRIPLES(D)
    N = {}
    for t ∈ T do
        n = TOGGLE-COLLIDER(t,D)
        if SCORE(n) > SCORE(D) then
            N = N ∪ n
        end if
    end for
    return N
end procedure

procedure BEST-SUCCESSOR(N)
    N = SHUFFLE(N)
    size = MAX(1, k * |N|)
    N' = {N_1, ..., N_size}
    return BEST(N')
end procedure
```

We incorporate two strategies for avoiding local maxima: a k-greedy approach (Nielsen et al., 2003) and random restarts. The k-greedy strategy consists of choosing the successor randomly from a fraction of the set of improved successors as defined by $k, k \in (0, 1]$. If $k = 1$, all of the successors are considered and pure greedy search is performed. If k is sufficiently small, then a successor is chosen at random. We found that the choice of k didn't affect the performance and for all of the experiments described in this paper we used $k = 0.5$. We observed that only a small number of random restarts were sufficient to produce structural accuracies that were equivalent or better than PC-EDGE. When we ran EDGE-OPT alone, we used $numRestarts = 25$; for runs as part of constraint relaxation, we used $numRestarts = 3$. Due to space considerations, we have not provided results of the parameter search here.

4.3 Evaluating Constraint Optimization

To evaluate our new constraint optimization approach as a replacement for the PC-EDGE algorithm, we compared the structural accuracy of models produced by both our constraint optimization approach and the PC-EDGE algorithm. Evaluating structural accuracy requires data generated from a known structure. To satisfy this requirement, we trained on data with a range of sample sizes generated from the following networks drawn from real-world domains: Alarm, Insurance, Powerplant[2], and Water[3]. We also trained on data generated from 25 random networks created using the BNGenerator[4] software (Ide and Cozman, 2002). We used the following parameter settings: *nNodes* chosen uniformly between 15 and 25, *minInDegree*=4, *maxOutDegree*=5, *maxVal*=5. The sample sizes we considered for Alarm, Insurance and Water were $n = \{250, 500, 1000, 2000, 5000, 7500, 10000\}$ and for Powerplant and the synthetic networks we considered $n = \{500, 5000, 10000\}$.

We compare the structural accuracy of the PC-EDGE and EDGE-OPT algorithms using the True Positive (TP) metric (Figure 2). The constraint optimization approach performs significantly better ($p = 0.01$) than PC-EDGE on three of the five scenarios we considered (Powerplant, Water, and Synthetic) and is statistically indistinguishable in the other cases. Performance is averaged over five training runs on each real network, and averaged across all 25 structures for the

[2]Available from `http://bndg.cs.aau.dk/Bayesian_networks/powerplant.net`

[3]Alarm, Insurance, and Water are available from the Bayesian Network Repository: `http://compbio.cs.huji.ac.il/Repository/`

[4]`http://www.pmr.poli.usp.br/ltd/Software/BNGenerator/index.html`

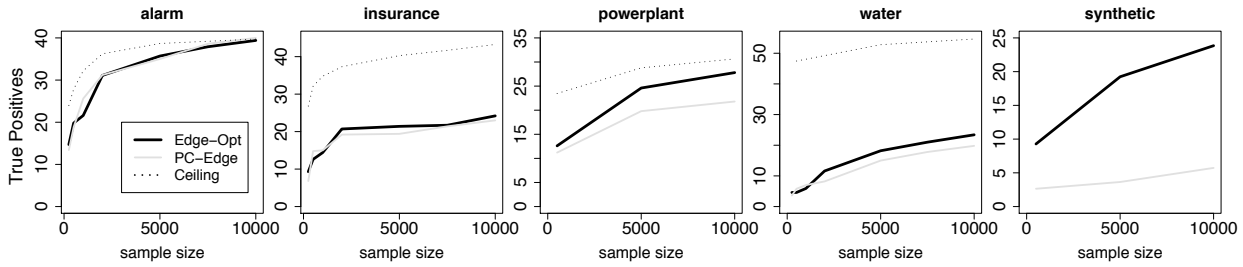

Figure 2: True Positive rates of PC-Edge and Edge-Opt. Differences are significant at $p = 0.01$ on Powerplant, Water, and Synthetic networks. The ceiling is the number of edges correctly included in the skeleton.

synthetic data. To test the significance of the differences between the learning curves, we used the randomized ANOVA approach developed by Piater et al. (1998). These results indicate that constraint optimization is a suitable replacement for the PC-Edge algorithm within our constraint relaxation algorithm which requires an edge orientation algorithm that is guaranteed to produce a DAG.

5 Experimental Evaluation

We evaluated the Relax algorithm on data generated from the 4 real-world Bayesian networks and 25 synthetic networks described in Section 4.3. We also evaluated widely used algorithms from four different classes of structure learning algorithms: (1) the constraint-based PC algorithm (Spirtes et al., 2000), (2) a hybrid algorithm (Hybrid) (Tsamardinos et al., 2006), (3) unconstrained greedy hill-climbing (GS), and (4) Greedy Equivalence Search (GES), which searches in the space of equivalence classes. The Relax, PC, and Hybrid algorithms all use constraints learned using the FAS algorithm. GES, while not a constraint-based algorithm, has been proven correct in the sample limit. We used our own implementation of each of the algorithms with the exception of GES, where we used the implementation in the TETRAD package. Note that the GES algorithm did not terminate successfully on Water. Our software is available at the first author's webpage[5].

A summary of results on the real data (shown in Table 1) shows that Relax outperforms the comparison algorithms across all metrics with few exceptions. We highlight the compelled F-measure score and the structural Hamming distance in Figure 3 as they are good summary measures of the causal interpretability of the model and the overall structural accuracy, respectively. Learning curves for the additional metrics can be viewed in the supplementary materials. On the four real-world datasets, Relax produces models with the highest compelled F-measure[6] (Figure 3). At small sample sizes, GS produces models with better compelled F-measure but also has significantly worse structural Hamming distance when compared to the Relax algorithm. To test whether these improvements were significant, we again used the randomized ANOVA test of Piater et al. (1998). This test is designed to determine the significance of the differences between two learning curves such as the ones shown in Figure 3. The p-values of the Relax algorithm compared to the other algorithms are shown in Table 2. These p-values indicate the strength of the main effect as measured using 1000 randomization trials.

Table 1: Proportion of runs on Alarm, Insurance, Powerplant, and Water where Relax equals or exceeds the performance of the other algorithms.

Measure	PC	Hybrid	GS	GES	Edge-Opt
TP	0.929	0.858	0.642	0.746	0.916
Prec.	0.831	0.575	0.858	0.881	0.716
Recall	0.871	0.825	0.425	0.701	0.920
F	0.849	0.708	0.617	0.701	0.818
SHD	0.551	0.433	0.967	0.985	0.449
Skel.	0.147	0.167	1.000	0.985	0.147
Orient	0.876	0.733	0.883	0.761	0.836
BDeu	–	0.867	0.017	0.896	1.000
LogLL	–	0.825	0.050	0.896	1.000

The results on the randomly generated networks are markedly different from the results on the real networks. The Relax algorithm performs significantly better than PC-Edge but significantly worse than both greedy search and Hybrid. An examination of the parameters of the true networks shows that distributions of the parameters vary significantly between the real and synthetic networks (Figure 4). The parameters on the real networks depend on the nature of the

[5]URL withheld

[6]Due to time constraints, the Powerplant results show only a limited number of relaxations, however, permitting more relaxations should only improve the score

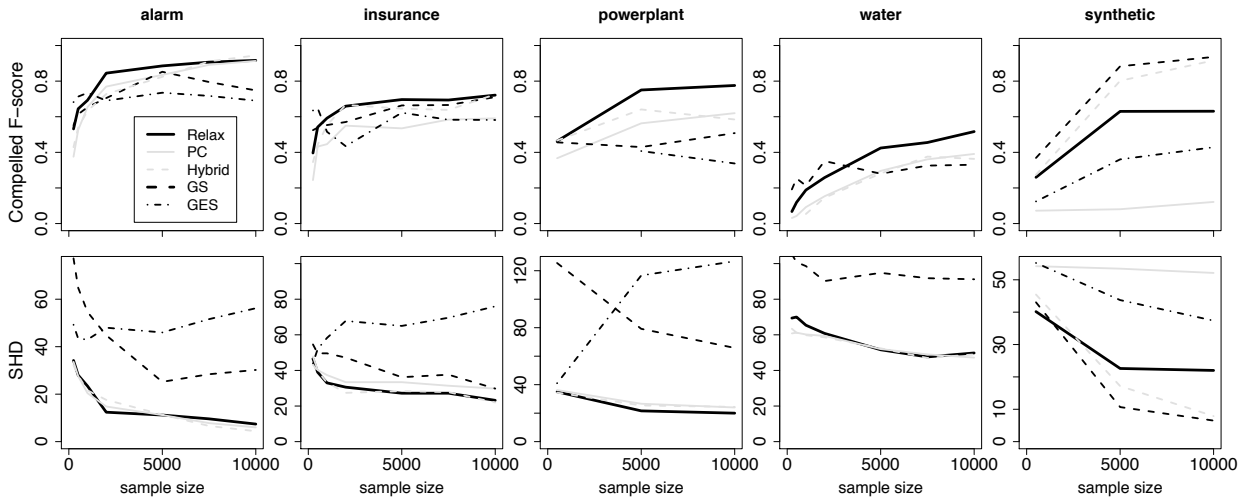

Figure 3: Evaluation of structural accuracy using compelled F-measure and structural Hamming distance.

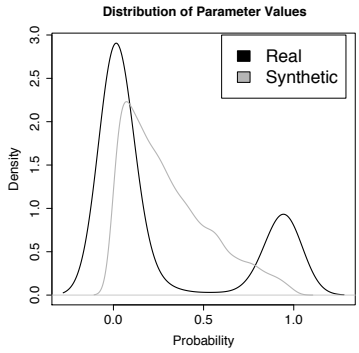

Figure 4: Distribution of the parameters of the true networks.

Table 2: P-values of the differences in compelled F-measure and structural Hamming distance (SHD) between RELAX and the baseline algorithms. *Italics* indicate that RELAX has worse performance.

	Network	PC	HYBRID	GS	GES	EDGE-OPT
F-measure	Alarm	0.006	0.006	0.027	0.01	0.035
	Insurance	0.01	0.135	0.617	0.052	0.007
	Powerplant	0.006	0.087	0.01	0.009	0.488
	Water	0.00	0.018	0.094	–	0.157
	Synthetic	0.00	*0.000*	*0.000*	0.00	0.00
SHD	Alarm	0.541	0.523	0.007	0.008	0.451
	Insurance	0.00	0.733	0.006	0.00	0.53
	Powerplant	0.006	0.021	0.00	0.00	0.472
	Water	0.00	0.00	0.00	–	0.00
	Synthetic	0.00	*0.122*	*0.02*	0.00	*0.009*

domain, whereas the BNGenerator software generates the parameters of the synthetic networks uniformly at random (Ide and Cozman, 2002).

Comparing the results from RELAX and EDGE-OPT indicates that using constraint relaxation leads to consistent improvements over edge orientation alone. The one exception is skeleton errors. Since RELAX is permitted to search a larger space of models, it tends to incur a small number of additional skeleton errors. These errors, however, are more than compensated for by the corresponding decrease in orientation errors that relaxation permits.

6 Discussion and Future Work

Constraint relaxation is a new approach for learning accurate structure of Bayesian networks. We show that a simple greedy algorithm for constraint relaxation produces models with significantly higher structural accuracy across a wide range of metrics. In particular, we highlight the improvements in causal interpretability as measured by the compelled F-measure. Constraint relaxation performs comparably to existing algorithms on structural Hamming distance.

The primary challenge of using this approach is the computational overhead. The EDGE-OPT algorithm requires running the minimal d-separation computation many times in the evaluation of each successor. We are exploring opportunities for caching and reusing d-separator computations to improve performance.

We are exploring applying constraint relaxation in con-

cert with alternative approaches such as identifying inaccurate constraints using the recently proposed approach based on the logic of argumentation (Bromberg and Margaritis, 2009) and techniques for controlling the false discovery rate during skeleton identification (Li and Wang, 2009; Tsamardinos and Brown, 2008).

References

J. Abellan, M. Gomez-Olmedo, and S. Moral. Some variations on the PC algorithm. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models*, 2006.

L. Badea. Determining the direction of causal influence in large probabilistic networks: A constraint-based approach. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 263–267, 2004.

F. Bromberg and D. Margaritis. Improving the reliability of causal discovery from small data sets using argumentation. *Journal of Machine Learning Research*, 10:301–340, February 2009.

J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1-2):43–90, 2002.

D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, November 2002.

W. B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley, 2009.

D. Dash and M. Druzdzel. Robust independence testing for constraint-based learning of causal structure. In *Proc. of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 167–174, 2003.

E. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58(1-3):21–70, 1992.

J. Ide and F. Cozman. Random generation of Bayesian networks. In *Proceedings of the 16th Brazilian Sympoisum on Artificial Intelligence*, pages 366–375. Springer, 2002.

J. Li and Z. J. Wang. Controlling the false discovery rate of the association/causality structure learned with the pc algorithm. *Journal of Machine Learning Research*, 10:475–514, February 2009.

S. Mani and G. F. Cooper. Causal discovery algorithms based on y structures. In *NIPS 2006 Workshop on Causality and Feature Selection*, 2006.

J. Nielsen, T. Kocka, and J. Pena. On local optima in learning Bayesian networks. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 435–442, 2003.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kauffman, 1988.

J. H. Piater, P. R. Cohen, X. Zhang, and M. Atighetchi. A randomized ANOVA procedure for comparing performance curves. In *Proceedings of the 15th International Conference on Machine Learning*, pages 430–438, 1998.

P. Spirtes, C. Meek, and T. Richardson. An algorithm for causal inference in the presence of latent variables and selection bias. In C. Glymour and G. F. Cooper, editors, *Computation, Causation and Discovery*, pages 211–252. MIT Press Cambridge, MA, USA, 1999.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. MIT Press, 2nd edition, 2000.

H. Steck. On the use of skeletons when learning in Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 558–565, 2000.

H. Steck and V. Tresp. Bayesian belief networks for data mining. *Proceedings of the 2nd Workshop on Data Mining und Data Warehousing als Grundlage Moderner Entscheidungsunterstutzender Systeme*, pages 145–154, 1996.

J. Tian, A. Paz, and J. Pearl. Finding minimal d-separators. Technical Report R-254, UCLA Computer Science Department, February 1998.

I. Tsamardinos and L. E. Brown. Bounding the false discovery rate in local Bayesian network learning. In *Proceedings of the 23rd AAAI Conference on Artifical Intelligence*, 2008.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, March 2006.

T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, 1990.

Supplementary Material Included below are the learning curves for all metrics considered in "Constraint Relaxation for Learning the Structure of Bayesian Networks". Results are on training data sampled from four real-world Bayesian networks and 25 randomly generated networks.

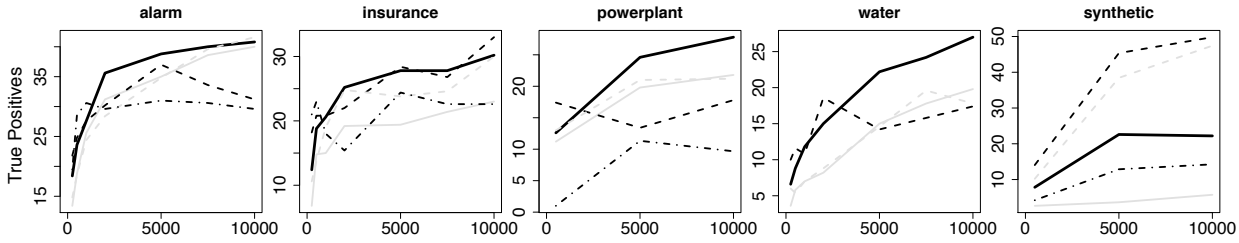

Figure 1: True Positives: Number of edges correct. (higher is better)

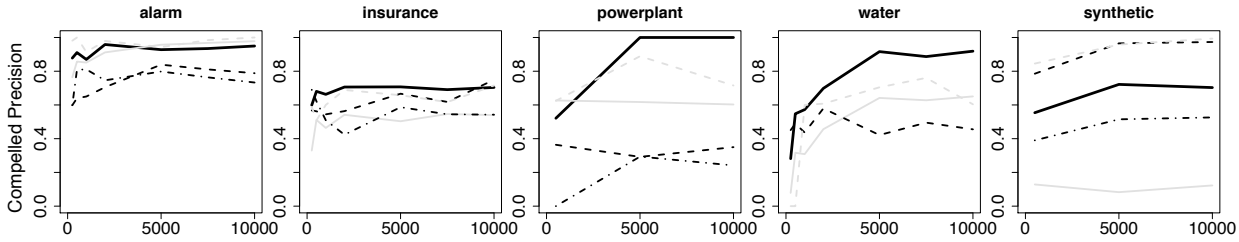

Figure 2: Compelled Precision: Percentage of learned compelled edges also appearing in the true model. (higher is better)

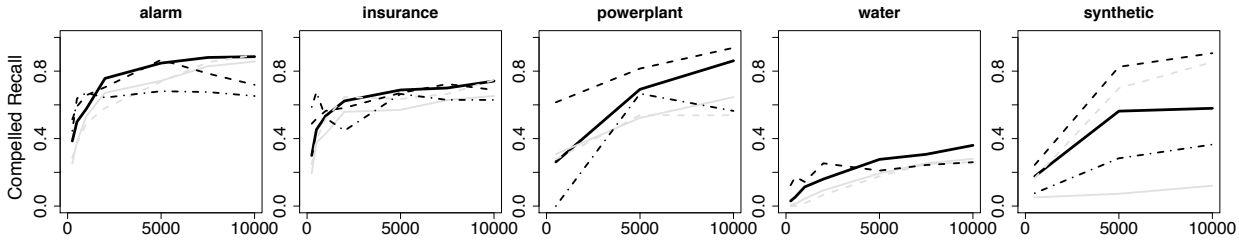

Figure 3: Compelled Recall: Percentage of true compelled edges also appearing in the learned model. (higher is better)

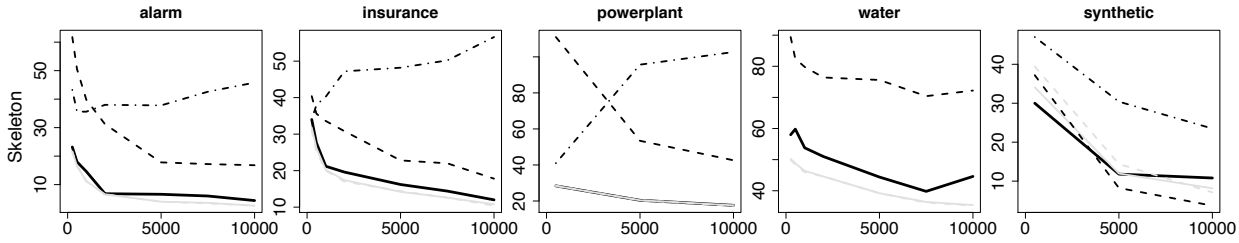

Figure 4: Skeleton Errors: Incorrect addition or subtraction of edges in the learned model. (lower is better)

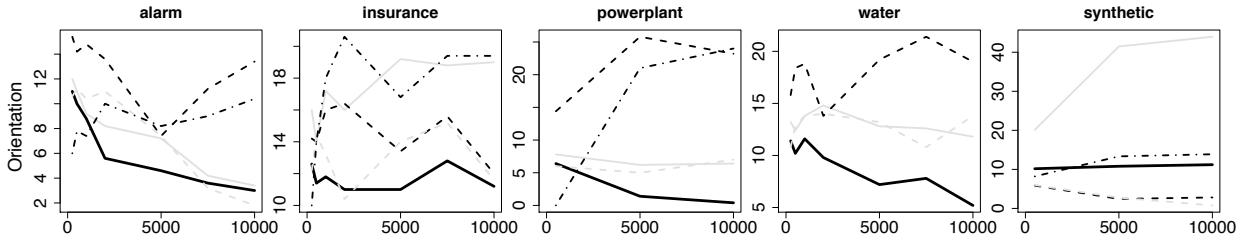

Figure 5: Orientation Errors: Edges where the orientation differs between the learned model and the true model. (lower is better)

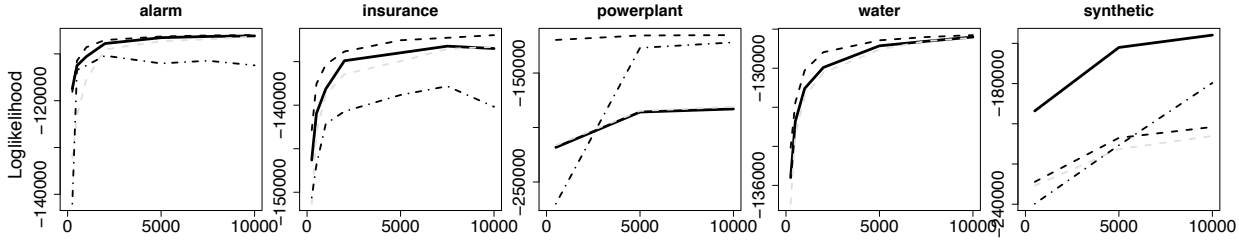

Figure 6: Loglikelihood: Likelihood of the learned model on a held-out test set of 10000 instances sampled from the true model. (higher is better)

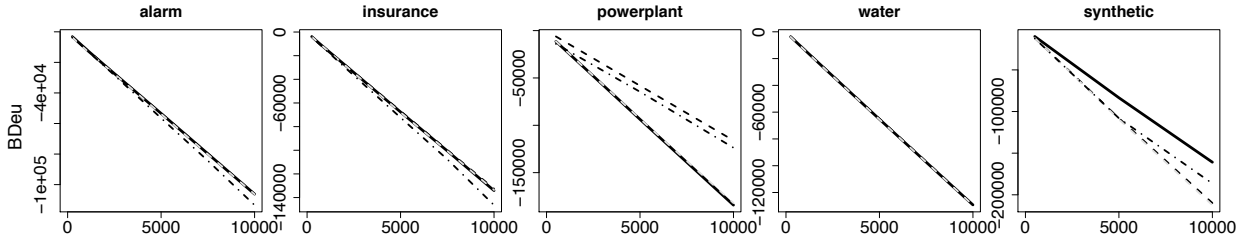

Figure 7: BDeu: BDeu score of the learned model on a held-out test set of 10000 instances sampled from the true model. (higher is better)