Recursive Autonomy Identification for Bayesian Network Structure Learning

Raanan Yehezkel and Boaz Lerner
Pattern Analysis and Machine Learning Lab
Electrical and Computer Engineering
Ben-Gurion University, Israel
{*raanany, boaz@ee.bgu.ac.il*}

Abstract

We propose a constraint-based algorithm for Bayesian network structure learning called recursive autonomy identification (RAI). The RAI algorithm learns the structure by recursive application of conditional independence (CI) tests of increasing orders, edge direction and structure decomposition into autonomous sub-structures. In comparison to other constraint-based algorithms d-separating structures and then directing the resulted undirected graph, the RAI algorithm combines the two processes from the outset and along the procedure. Learning using the RAI algorithm renders smaller condition sets thus requires a smaller number of high order CI tests. This reduces complexity and run-time as well as increases accuracy since diminishing the curse-of-dimensionality. When evaluated on synthetic and "real-world" databases as well as the ALARM network, the RAI algorithm shows better structural correctness, run-time reduction along with accuracy improvement compared to popular constraint-based structure learning algorithms. Accuracy improvement is also demonstrated when compared to a common search-and-score structure learning algorithm.

1 INTRODUCTION

Most algorithms for Bayesian network (BN) structure learning are either search-and-score based [Heckerman, 1995; Friedman et al., 1997] in which the structure achieving the highest score given the data is pursued or constraint-based in which the structure is learned from constraints derived from statistical tests of independence between variables combined with causality inference rules [Pearl, 2000; Spirtes et al., 2000]. The main problem of constraint-based algorithms is their inefficiency and inaccuracy (due to the curse-of-dimensionality) in performing conditional independence (CI) tests for large condition sets. Most constraint-based algorithms, such as Inductive Causation (IC) [Pearl, 2000], PC [Spirtes et al., 2000] and Three Phase Dependency Analysis (TPDA), [Cheng et al., 1997], construct a directed acyclic graph (DAG) in two consecutive stages. First is learning associations between variables for constructing an undirected structure. This requires an exponentially growing number of CI tests with the number of nodes, which can be reduced to polynomial by fixing the number of parents (PC algorithm) or using the values computed in the CI test and some strong assumptions (TPDA algorithm). These assumptions however may not be valid in all situations. Another flaw of the TPDA algorithm is ignoring the curse-of-dimensionality in CI tests by not limiting the size of the condition set. The second stage in most constraint-based algorithms is causality inference performed in two consecutive steps: finding and directing V-structures and inductively directing additional edges [Pearl, 2000]. Causality inference, and especially the induction step, is unstable, i.e., small errors in the input to the stage yield large errors at its output [Spirtes et al., 2000]. Thus, the algorithms increase stability by separating the two stages trying in the first stage to minimize erroneous decisions about d-separation caused by invalid threshold selection or poor estimation due to the curse-of-dimensionality.

We propose a constraint-based algorithm that recursively tests conditional independencies with condition sets of increasing orders, directs edges for each order and identifies autonomous sub-structures complying with the Markov property (i.e., the sub-structure includes all node parents). By considering directed rather than undirected

edges, the RAI avoids unnecessary CI tests and performs tests using smaller condition sets. Repeated for autonomies decomposed recursively from the graph both mechanisms reduce computational and time complexities, database queries and errors of subsequent iterations. Using smaller condition sets, the RAI algorithm also improves accuracy since diminishing the curse-of-dimensionality. After providing some preliminaries in Section 2 we introduce the RAI algorithm in Section 3 and present its experimental evaluation in Section 4 before concluding the paper in Section 5.

2 PRELIMINARIES

A BN B(G,Θ) consists of a structure (graph) G and a set of probabilities Θ quantifying the graph. G(**V**,**E**) consists of **V**, a set of nodes representing domain variables, and **E** a set of edges connecting the nodes. $\mathbf{Pa_p}$(X,G), **Adj**(X,G) and **Ch**(X,G) are respectively the sets of potential parents, adjacent nodes and children of node X in a partially directed graph G, $\mathbf{Pa_p}(X,G)=\mathbf{Adj}(X,G)\backslash\mathbf{Ch}(X,G)$. Similarly, **Pa**(X,G) and **Desc**(X,G) are the sets of parents and descendants of X in G. We indicate that X and Y are independent given a set of nodes **S** using $X \perp\!\!\!\perp Y|\mathbf{S}$ and make use of the notion of d-separation [Pearl, 2000]. We also define d-separation resolution evaluating d-separation for different values of the maximal number of nodes in the condition set, an exogenous cause to a graph and an autonomous sub-structure.

Definition 1: The d-separation resolution between any pair of non-adjacent nodes is the size of the smallest condition set that d-separates the two nodes.

Definition 2: The d-separation resolution of a graph is the highest d-separation resolution in the graph.

Definition 3: Y is an exogenous cause to G(**V**,**E**) if $Y \notin \mathbf{V}$ and $\forall X \in \mathbf{V}$, $Y \in \mathbf{Pa}(X)$ or $Y \notin \mathbf{Adj}(X)$ [Pearl, 2000].

Definition 4: A sub-structure $G^A(\mathbf{V^A},\mathbf{E^A})$ in G(**V**,**E**) s.t $\mathbf{V^A} \subset \mathbf{V}$, $\mathbf{E^A} \subset \mathbf{E}$ is autonomous given a set of exogenous nodes $\mathbf{V_{ex}}$ to G^A if $\forall X \in \mathbf{V^A}$, $\mathbf{Pa}(X,G) \subset \{\mathbf{V^A} \cup \mathbf{V_{ex}}\}$. If $\mathbf{V_{ex}}$ is empty, we say the sub-structure is autonomous.

We define sub-structure autonomy in the sense that the sub-structure holds the Markov property for its nodes. Given a structure G, any two non-adjacent nodes in an autonomous sub-structure G^A are d-separated given nodes either included in the sub-structure or exogenous causes to it. This notion is elaborated in Section 3.3.

3 RECURSIVE AUTONOMY IDENTIFICATION

Starting from a complete graph and proceeding from low to high graph d-separation resolution, the RAI algorithm uncovers the correct pattern (i.e., a family of structures Markov equivalent to the true underlying structure) by recursive (1) test of CI between nodes and removal of edges related to independencies (thinning), (2) edge direction according to inferred causality rules and (3) graph decomposition into autonomous sub-structures.

CI testing of order n between X and Y is performed by thresholding a criterion, such as the χ^2 goodness of fit [Spirtes et al., 2000] or conditional mutual information [Cheng et al., 1997]. The criterion measures dependence conditioned on a set of n nodes from the parents of X or Y determined by the Markov property [Pearl, 2000], e.g., if X is directed into Y only Y's parents are included in the set.

Directing edges is conducted according to causality rules [Pearl, 2000] by identifying intransitive triplets of nodes (V-structures), i.e., non-adjacent parents having a common child, directing the relevant edges, and applying additional rules to further direct edges until no more edges can be directed (the inductive step).

Decomposition into autonomous sub-structures reveals the structure hierarchy and allows performing a fewer CI tests conditioned on a large number of potential parents thereby reducing complexity. The RAI algorithm identifies ancestor and descendant sub-structures, the latter are autonomous given nodes of the former.

3.1 THE RAI ALGORITHM

Iteration of the RAI algorithm starts with knowledge produced in the previous iteration and the current d-separation resolution, n. Previous knowledge includes G_{start}, a structure having d-separation resolution of n-1 and $\mathbf{G_{ex}}$, a set of structures having each possible exogenous causes to G_{start}. In the first iteration, n = 0, $G_{start}(\mathbf{V},\mathbf{E})$ is a complete graph and $\mathbf{G_{ex}}=\varnothing$.

Given a structure G_{start} having d-separation resolution n-1, the RAI algorithm seeks independencies between adjacent nodes conditioned on sets of size n, resulting in a structure having d-separation resolution of n. After directing edges, the algorithm decomposes the structure into ancestor and descendent autonomous sub-structures in order to reduce complexity of successive stages. A descendant sub-structure is established by identifying the lowest topological order nodes (either a single node or a

Main function G_{out} = RAI(n,G_{start},$\mathbf{G_{ex}}$)

Exit condition

If all nodes in G_{start} have less than n-1 potential parents exit.

A. Thinning the link between $\mathbf{G_{ex}}$ and $\mathbf{G_{start}}$ and directing $\mathbf{G_{start}}$

1. For every node Y in G_{start} and its parent X in $\mathbf{G_{ex}}$, if $\exists \mathbf{S} \subset \{\mathbf{Pa_p}(Y,\mathbf{G_{ex}}) \backslash X \cup \mathbf{Pa_p}(Y,G_{start})\}$ and $|\mathbf{S}|$=n s.t $X \perp\!\!\!\perp Y|\mathbf{S}$, then remove the edge between X and Y.
2. Direct the edges using causality inference rules.

B. Thinning, directing and decomposing $\mathbf{G_{start}}$.

1. For every node Y and its potential parent X, both in G_{start}, if $\exists \mathbf{S} \subset \{\mathbf{Pa_p}(Y,\mathbf{G_{ex}}) \cup \mathbf{Pa_p}(Y,G_{start}) \backslash X\}$ and $|\mathbf{S}|$=n s.t $X \perp\!\!\!\perp Y|\mathbf{S}$, then remove the edge between X and Y.
2. Direct the edges using causality inference rules.
3. Group the nodes having the lowest topological order into a descendant sub-structure G_D.
4. Remove G_D from G_{start} temporarily, and define the resulting unconnected structures as ancestor sub-structures $G_{A1},\ldots, G_{Ak}$.

C. Ancestor sub-structure decomposition

for i = 1 to k, call RAI(n+1,G_{Ai},$\mathbf{G_{ex}}$)

D. Descendant sub-structure decomposition

1. Define $\mathbf{G_{D\_ex}}$={$G_{A1},\ldots,G_{Ak},\mathbf{G_{ex}}$} as the exogenous structure to G_D.
2. Call RAI(n+1,G_D,$\mathbf{G_{D\_ex}}$)

Figure 1: The RAI algorithm

several nodes having the same lowest order). This structure is autonomous given ancestor sub-structures composed of nodes of higher order. In order to consider a smaller number of parents for each node of the descendent sub-structure, the algorithm recursively learns ancestor sub-structures and only then their descendant sub-structure. Note that this latter structure may consist of a several non-connected sub-structures. Figures 1-3 show respectively the RAI algorithm, a manifesting example and the algorithm execution order for this example. Figure 2a shows the true underlying structure. Initially, G_{start} is the complete graph and $\mathbf{G_{ex}}$ is empty so stage A is skipped. At stage B1, any pair of nodes is CI tested given an empty condition set (marginal independence) yielding the removal of the edges between node 1 and nodes 3, 4

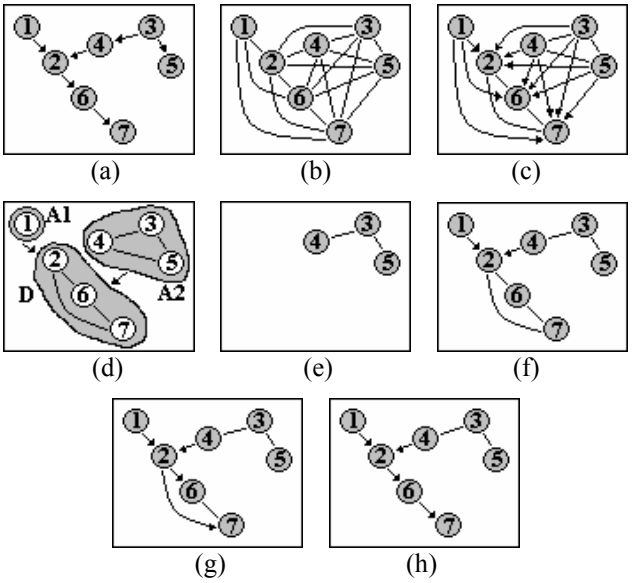

Figure 2: Learning an example structure. a) The true underlying structure and structures learned by the RAI algorithm in stages (see Figure 1) b) B1, c) B2, d) B4, e) C, f) D and A1, g) D and A2 and h) D and B1 (the resulting structure)

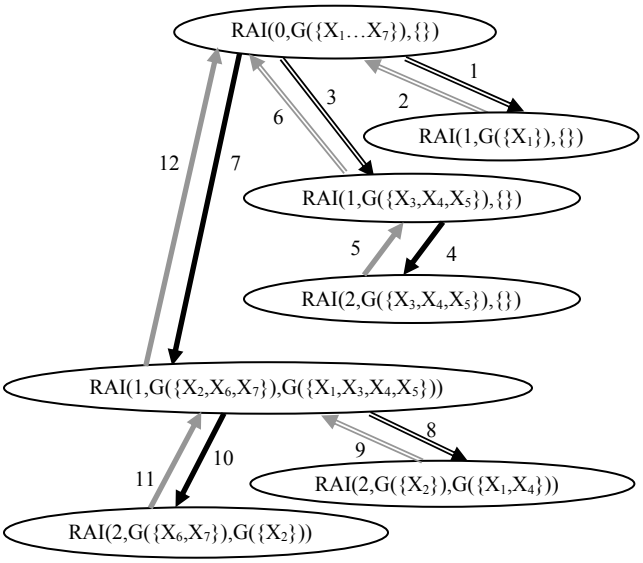

Figure 3: The execution order of the RAI algorithm for the structure of Figure 2. Recursive calls of stages C and D are marked with a double and single arrow, respectively.

and 5 (Figure 2b). The causal relations inferred at stage B2 are shown in Figure 2c. The nodes having the lowest topological order (2, 6, 7) are grouped into a descendant sub-structure G_D (stage B3) while the remaining nodes form two unconnected ancestor sub-structure, G_{A1} and G_{A2} (stage B4) (Figure 2d). At stage C the algorithm is called recursively for each of the ancestor sub-structures with n=1, $G_{start}=G_{Ai}$ and $\mathbf{G_{ex}}=\varnothing$. Since sub-structure G_{A1} contains a single node, the exit condition for the structure is satisfied. While calling $G_{start}=G_{A2}$, stage A is skipped and stage B1 identifies that $X_4 \perp\!\!\!\perp X_5|X_3$ thus removes edge X_4—X_5. No causal relations are identified so the nodes have equal topological order and they are grouped to from a descendant sub-structure. The recursive call for this sub-structure with n=2 is returned immediately since the exit condition is satisfied (Figure 2e). Moving to stage D, the RAI is called with n=1, $G_{start}=G_D$ and $\mathbf{G_{ex}}=\{G_{A1},G_{A2}\}$. Then, in stage A1 relations $(X_1 \perp\!\!\!\perp \{X_6,X_7\}|X_2)$, $(X_4 \perp\!\!\!\perp \{X_6,X_7\}|X_2)$ and $(\{X_3,X_5\} \perp\!\!\!\perp \{X_2,X_6,X_7\}|X_4)$ are identified and the corresponding edges are removed (Figure 2f). At stage A2 node X_2 is identified as a parent of X_6 and X_7 (Figure 2g). Stage B1 identifies the relation $(X_2 \perp\!\!\!\perp X_7|X_6)$ and stage B2 identifies X_6 as a parent of X_7 (Figure 2h). Further recursive calls are returned and the resulting partially directed structure represents a family of Markov equivalent structures of the true structure.

3.2 MINIMALITY, STABILITY & COMPLEXITY

Minimality A structure having a higher d-separation resolution entails a fewer dependencies and thus is simpler and preferred to a structure having a lower d-separation resolution [Pearl, 2000]. By increasing the resolution, the RAI algorithm moves from a complete structure having maximal dependency relations between variables to structures having less (or equal) dependencies than previous structures ending in a structure having no edges between conditionally independent nodes, i.e., a minimal structure.

Stability is measured by the number of errors in the output structure due to CI test errors, which are the only source of errors. CI test errors are the result of unnecessary large condition set leading to the curse-of-dimensionality or choosing an inaccurate condition set due to partial information (e.g., undirected edges). Although as a recursive algorithm the RAI might be unstable, errors are practically less likely to occur since the algorithm utilizes more information (e.g., edge direction and graph decomposition) from previous iterations to choose smaller, informative condition sets for performing the tests.

Complexity CI tests are the major contribution to complexity (run-time) [Cheng and Greiner, 1999]. In the worst case, the RAI algorithm will not direct any edges nor decompose the structure thus identify the entire structure as a descendant sub-structure calling stage D iteratively while skipping most other stages. Then, the execution of the algorithm will be similar to that of the PC algorithm and the complexity will be bounded by that of the PC algorithm. Given the maximal number of possible parents k and the number of nodes n, the number of CI tests is bounded by [Spirtes et al., 2000]

$$2\binom{n}{2}\cdot\sum_{i=0}^{k}\binom{n-1}{i}\leq\frac{n^2(n-1)^{k-1}}{(k-1)!}.$$

This worst case scenario rarely occurs in "real-world" applications requiring structures having colliders.

3.3 CORRECTNESS

Proposition: If the input data to the RAI algorithm is faithful to a DAG, G, having any d-separation resolution, then it yields the correct pattern, G_{out}.

Proof: (by induction, ignoring notions common to the RAI and PC algorithms which are proved in [Spirtes et al., 2000])

Base case: If the input data to the RAI algorithm is faithful to a DAG with d-separation resolution 0, then it yields the correct pattern G_{out}.

Since G_{start} is a complete graph, the algorithm tests in stage B marginal independence between pairs of nodes and then direct edges. Thus, the resulting structure contains only edges between marginally dependent nodes, therefore having d-separation resolution of 0. From the decomposition stages, B3 and B4, based on the topological order identified from the partially directed structure, it follows that every edge from a node X in an ancestor sub-structure to a node Z in the descendant sub-structure is directed, $X \rightarrow Z$. Also, there is no edge connecting one ancestor sub-structure to another ancestor sub-structure. Thus, every ancestor sub-structure contains all the potential parents of its nodes, i.e., it is autonomous.

Lemma 1: If the given data entails $X \perp\!\!\!\perp Y|\mathbf{S}$ and X,Y are members of an autonomous sub-structure $G^A(\mathbf{V^A},\mathbf{E^A})$, then $\exists\mathbf{S'}$ such that $\mathbf{S'}\subset\mathbf{V^A}$ and $X \perp\!\!\!\perp Y|\mathbf{S'}$.

Lemma 2: In a DAG, if X and Y are non-adjacent and X is not a descendant of Y then X and Y are d-separated given **Pa**(Y) (proved in [Spirtes et al., 2000]).

An autonomous sub-structure contains all potential parents (either sub-structure nodes or exogenous causes) of each of its nodes. Thus, from Lemma 2, if X and Y are independent given a set of nodes (i.e., d-separated in the true underlying graph), then they are d-separated given $\mathbf{Pa_P}(X)$ or $\mathbf{Pa_P}(Y)$ which are contained in the autonomous sub-structure. Thus, every ancestor sub-structure can be processed independently by recursive calls of the algorithm. The recursive call of the descendant sub-structure regards the ancestor sub-structure nodes as exogenous causes. The data does not entail any higher order conditional independencies and no more edges are removed.

Inductive case: Suppose that the RAI algorithm yields the correct pattern given data faithful to a DAG having d-separation resolution n. Then, given data faithful to a DAG having d-separation resolution n+1 the RAI algorithm yields the correct pattern.

After achieving d-separation resolution of n in an autonomous sub-structure, $G^{(n)}$, a recursive call with n+1 is made. The exit condition is not satisfied in case an edge exists in $G^{(n)}$ and does not exist in the true structure G_t. Suppose an edge $E_{XY}=(X\rightarrow Y)$ exists, such that $E_{XY}\in G^{(n)}$ and $E_{XY}\notin G_t$, then the smallest condition set required to identify the independency between the nodes is S_{XY}, such that $|S_{XY}| \geq n+1$. Thus, it follows from Lemma 2 that either $|\mathbf{Pa}(X)| \geq n+1$ or $|\mathbf{Pa}(Y)|\geq n+1$ and the exit condition is not satisfied. Every pair of connected nodes is tested for independence in stage B1 using condition sets of size n+1 and the corresponding edges are removed resulting in a sub-structure having d-separation resolution of n+1.

The correctness of edge directing is discussed in [Pearl, 2000; Spirtes et al., 2000].

4 EXPERIMENTS AND RESULTS

The RAI algorithm was experimentally compared to the PC and TPDA algorithms, two popular constraint-based structure learning algorithms reported frequently as having good performance [Ramsey et al., 2002]. For simplicity, no speeding-up heuristic techniques [Spirtes et al., 2000] were applied to either algorithm, and the RAI algorithm employed only V-structure identification deferring the inductive step after forming the structure.

The complexity and prediction accuracy of the RAI algorithm were compared to those of the PC and TPDA algorithms using a synthetic problem and fifteen "real-world" databases of the UCI Repository [Murphy and Aha, 1994]. Interested mainly in classification, the

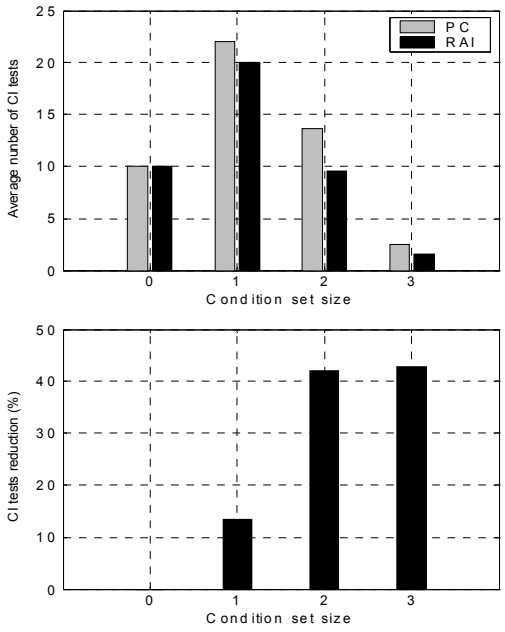

Figure 4: (a) The number of CI tests required by the RAI and PC algorithms for increasing orders averaged over all possible structures having five nodes. (b) CI test reduction by the RAI algorithm compared to the PC algorithm

prediction accuracy is preferred over the likelihood in evaluating performance, as the likelihood ignores the importance of the class variable [Friedman et al., 1997]. Structural correctness was evaluated in recovering the ALARM network in comparison to the TPDA and PC algorithms. BN implementation was aided by the Bayes net toolbox (BNT) [Murphy, 2001] and BNT structure learning package [Leray and Francois, 2004].

4.1 A SYNTHETIC PROBLEM

All 29,281 possible structures having five nodes were learned by the PC and RAI algorithms. Since the true structure is known, the actual CI relationships could be inputted to the algorithms. Figure 4a shows the complexity, evaluated using the averaged number of CI tests over all possible structures, of the algorithms for increasing orders (condition sets). Figure 4b illustrates the percentage of CI tests reduced by the RAI algorithm in comparison to the PC algorithm.

4.2 "REAL-WORLD" DATA

A several databases of the UCI Repository were employed in order to evaluate prediction accuracy. When needed, continuous variables were discretized using the

Table 1. The average number (percentage) of CI tests reduced by the RAI algorithm compared to the PC algorithm for different orders

Database	CI test order				
	0	1	2	3	4
shuttle (s)	0 (0)	1.4 (0.7)	95.8 (43.8)	117.6 (49.3)	83.6 (56.0)
car	0 (0)	16 (100)	11.2 (100)	3.2 (100)	
corral	0 (0)	22.4 (100)	26 (100)	3.6 (100)	
mofn 3,7,10	0 (0)	17 (100)	4 (100)		
tic-tac-toe	0 (0)	53.2 (27.1)	56.6 (48.6)	1.8 (51.4)	
led7	0 (0)	46.2 (45.7)	105 (100)	140 (100)	105 (100)
breast	0 (0)	107.2 (54.8)	35 (99.1)		
vote	0 (0)	24.2 (21.9)	17.2 (98.1)	6.4 (100)	1 (100)
flare C	0 (0)	16 (39.6)	3 (100)		
wine	0 (0)	25.8 (41.0)	44.2 (67.6)	40.6 (82.4)	19 (96.7)
cmc	0 (0)	10.2 (10.9)	8 (32.5)		
crx	0 (0)	8.8 (49.6)			
zoo	0 (0)	82 (27.8)	365.8 (29.6)	1033.4 (27.7)	1928.6 (25.6)
australian	0 (0)	3.8 (34.4)			
iris	0 (0)	2 (40)			

Table 2. Mean (std) prediction accuracy of the RAI algorithm in comparison to the PC algorithm and "other" classifiers reported in [Friedman et al., 1997] (F) and [Cheng and Greiner, 1999] (TPDA algorithm) (C), as well as the cut (%) of CI test run-time using the RAI algorithm in comparison to the PC algorithm

Database	run-time cut (%)	PC accuracy (%)	RAI accuracy (%)	other
shuttle (s)	**38.94**	98.40	**99.22**	99.17(F)
car	**91.10**	85.07 (1.83)	**92.94** (1.06)	86.11(C)
corral	**87.94**	84.53 (15.45)	**98.52** (3.31)	97.60(F)
mofn 3,7,10	**67.70**	81.45	**93.16**	85.94(F)
tic-tac-toe	**36.52**	74.74 (1.48)	**75.57** (1.93)	
led7	**91.74**	73.31 (1.80)	**73.59** (1.56)	
breast	**71.87**	95.46 (2.04)	96.49 (1.61)	**96.92**(F)
vote	**46.06**	95.64 (1.87)	**95.87** (1.71)	94.94(F) 95.17(C)
flare C	**20.38**	**84.30** (2.54)	**84.30** (2.54)	82.74(F) 82.27(C)
wine	**29.11**	85.44 (7.79)	**87.07** (5.88)	
cmc	**14.22**	50.92 (2.33)	**51.12** (3.16)	
crx	**25.25**	**86.38** (2.63)	**86.38** (2.63)	85.60(F)
zoo	**13.63**	88.95 (8.79)	88.95 (8.79)	
australian	**6.05**	85.51 (0.52)	85.51 (0.52)	**86.23**(F)
iris	**19.10**	**96.00** (4.35)	93.33 (2.36)	94.00(F)

MLC++ library [Kohavi et al., 1994]. Variable A14 of the "shuttle-small (s)" database was ignored by the discretization function of MLC++ and thus omitted from the experiments. "flare1" and "flare2" were merged to form the "flare C" database where the class node is the number of "C-class" flares. All databases were analyzed using a CV5 experiment except the large "shuttle" and "mofn 3-7-10" databases which were analyzed using a hold-out experiment. CI tests were carried out using the χ^2 test [Spirtes et al., 2000] with thresholds chosen for each algorithm and database in order to maximize the prediction accuracy on a validation set selected from the training set. If a several thresholds were suitable, the chosen threshold was that leading to the fewest CI tests. Parameter learning was performed using sequential Bayesian updating with Dirichlet priors of unit hyper-parameters [Heckerman, 1995].

Complexity was measured by the number of CI tests employed and the corresponding run-time. Table 1 shows the average number and percentage of CI tests reduced by the RAI algorithm compared to the PC algorithm for

different orders. A 100% cut in CI tests for a specific order means that the RAI does not need any CI tests for this order. Empty cells mean that no CI tests of this order are required. Both Table 1 and Table 2, depicting the cut in run-time due to the RAI algorithm, demonstrate that the RAI algorithm outperforms the PC algorithm in all cases.

Prediction Accuracies of the RAI and PC algorithms for the experimented databases are summarized in Table 2. On ten of the fifteen databases the RAI algorithm improves accuracy on the PC algorithm, on four keeps accuracy intact and on the remaining "iris" database deteriorates accuracy. Examination of the "iris" database reveals discrepancy between the results of CI tests of orders 0 and 1 violating the Markov property. Three nodes are found marginally dependent on each other whereas nodes of each pair of this triplet are found independent given the third node. The prediction accuracy is also compared in Table 2 to that of the TPDA algorithm [Cheng and Greiner, 1999] and a BN learned by a search-and-score method using the minimum description length criterion [Friedman et al., 1997].

4.3 LEARNING THE ALARM NETWORK

Recovering the correct structure was evaluated using the ALARM network [Beinlich et al., 1989], which is widely accepted as a benchmark for evaluating structure learning algorithms. The RAI algorithm was compared to the PC and TPDA (PowerConstructor [Cheng, 1998]) algorithms using ten randomly generated databases each containing 10,000 cases. Since the TPDA algorithm had used the conditional mutual information CI test, we employed this test also here. For comparison, we selected the TPDA threshold of 0.003 [Cheng et al., 1997] for testing also the RAI algorithm and a threshold of 0.002 for the PC algorithm providing better accuracy for this algorithm than using a threshold of 0.003.

Structural correctness for the algorithms was evaluated using two types of errors due to extra edges (EE; commission) and missing edges (ME; omission) (Table 3). The PC and RAI algorithms achieved the smallest errors of extra and missing edges, respectively. The total structural error (Table 4) accounting for both errors was evaluated using

$$\text{Error}_T = \sqrt{\text{EE}^2 + \text{ME}^2}\ .$$

The RAI algorithm yielded structures with the smallest total structural error of all algorithms which was validated using a t-test with 1% significance level. Others structural errors (e.g., edge reversal) were not recorded though we expect the RAI algorithm to dominate both algorithms due to its enhanced mechanism of directing edges.

Table 3. Extra edge (EE) and missing edge (ME) errors (%) when learning the ALARM network in 10 trials using the TPDA, PC and RAI algorithms

Trial	TPDA		PC		RAI	
	EE	ME	EE	ME	EE	ME
1	0.48	8.70	0.16	2.17	0.97	0
2	0.32	4.35	0	6.52	0.65	2.17
3	0.32	4.35	0	4.35	0.65	2.17
4	0.32	6.52	0.16	4.35	0.32	0
5	0.48	8.70	0	2.17	0.65	0
6	0.48	8.70	0.16	4.35	0.48	0
7	0.48	8.70	0.32	0	0.65	0
8	0.16	2.17	0.16	2.17	0.81	2.17
9	0.16	2.17	0.16	4.35	0.81	2.17
10	0.48	8.70	0.32	4.35	0.65	0
mean (std)	0.37 (0.13)	6.30 (2.80)	**0.15** (0.12)	3.48 (1.83)	0.66 (0.18)	**0.87** (1.12)

Table 4. The total structural error (%) in 10 trials of the ALARM network learned using the TPDA, PC and RAI algorithms

Trial	TPDA	PC	RAI
1	8.71	2.18	0.97
2	4.36	6.52	2.27
3	4.36	4.35	2.27
4	6.53	4.35	0.32
5	8.71	2.17	0.65
6	8.71	4.35	0.48
7	8.71	0.32	0.65
8	2.18	2.17	2.32
9	2.18	4.35	2.32
10	8.71	4.36	0.65
mean (std)	6.32 (2.80)	3.51 (1.77)	**1.29** (0.88)

Complexity The average reduction in CI tests achieved by the RAI algorithm compared to the PC algorithm for the ALARM network is presented in Figure 5. The RAI algorithm avoids completely the use of CI tests of order 4 and 5 and almost completely CI tests of order 3, and it reduces the use of CI tests of order 2 by more than 83%. However, there is almost no reduction in CI tests of order 1 which are most of the tests. The total CI test run-time

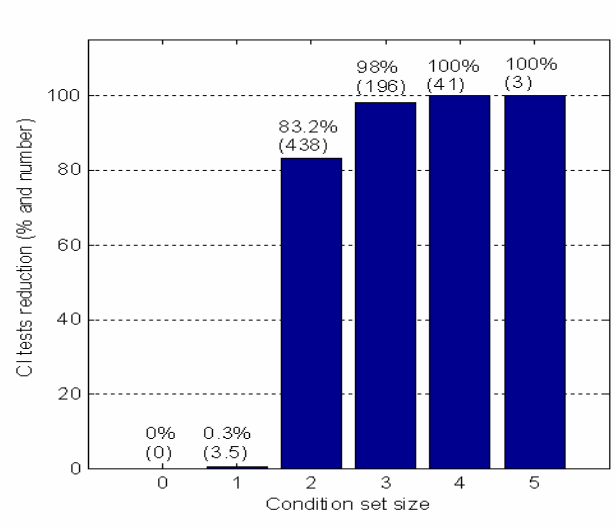

Figure 5: Average percentage (number) of CI tests reduced due to the RAI algorithm compared to the PC algorithm for increasing orders and the ALARM network

reduced by the RAI algorithm compared to the PC algorithm is 38%.

5 DISCUSSION

The performance of constraint-based algorithms of BN structure learning depends on the size of the condition set used for testing conditional independence. The larger the condition set is, the more CI tests (especially of high order) have to be performed and the less is their accuracy.

We propose the constraint-based RAI algorithm that learns BN structures recursively by performing 1) CI tests of increasing orders, along with 2) directing edges employing causality inference rules and 3) decomposing the structure into autonomous sub-structures. These mechanisms provide smaller condition sets enabling the performance of fewer CI tests of higher order thus reducing the algorithm run-time and increasing its accuracy. Other constraint-based algorithms directing edges after accomplishing the undirected graph using all orders, rather than continuously through learning, are expensive and more sensitive to errors accumulated along the procedure.

We demonstrate on a synthetic problem, fifteen real-world databases and the ALARM network that the RAI algorithm significantly reduces the number of CI tests required for structure learning and yields more accurate structures as well as higher prediction accuracy compared to other constraint-based algorithms.

Acknowledgement

This work was supported in part by the Paul Ivanier Center for Robotics and Production Management, Ben-Gurion University, Beer-Sheva, Israel.

References

Beinlich, I. A., Suermondt, H. J., Chavez, R. M. & Cooper, G. F. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Second European Conf. on Artificial Intelligence in Medicine*, pages 246-256, 1989.

Cheng, J. PowerConstructor system, 1998. http://www.cs.ualberta.ca/~jcheng/bnpc.htm.

Cheng, J., Bell, D. & Liu, W. Learning Bayesian networks from data: an efficient approach based on information theory. *Sixth ACM Int. Conf. on Information and Knowledge Management*, pages 325-331, 1997.

Cheng, J. & Greiner, R. Comparing Bayesian network classifiers, *Fifteenth Conf. on Uncertainty in Artificial Intelligence*, pages 101-107, 1999.

Friedman, N., Geiger, D. & Goldszmidt, M. Bayesian network classifiers. *Machine Learning,* 29:131-161, 1997.

Heckerman, D. A tutorial on learning with Bayesian networks. MS TR-95-06, March 1995.

Kohavi, R., John, G., Long, R., Manley D. & Pfleger, K. MLC++: A machine learning library in C++, *Sixth Int. Conf. on Tools with AI*, pages 740-743, 1994.

Leray, P. & Francois, O. BNT structure learning package: documentation and experiments. PSI TR, 2004.

Murphy, K. Bayes net toolbox for Matlab. Computing Science & Statistics, 33, 2001.

Murphy, P. M. & Aha, D. W. UCI Repository of machine learning databases, 1994. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Pearl, J. *Causality: Models, Reasoning, and Inference*. Cambridge. 2000.

Ramsey, J., Gazis, P., Roush, T., Spirtes, P. & Glymour, C. Automated remote sensing with near infrared reflectance spectra: Carbonate recognition. *Data Mining & Knowledge Discovery*, pages 277-293, 2002.

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