JMLR Workshop and Conference Proceedings 6:191–202 NIPS 2008 workshop on causality

Discover Local Causal Network around a Target to a Given Depth

You Zhou ZHOUYOU@PKU.EDU.CN
Changzhang Wang CHANGZHANG@PKU.EDU.CN
Jianxin Yin JIANXINYIN@GMAIL.COM
Zhi Geng ZGENG@MATH.PKU.EDU.CN
School of Mathematical Sciences
Peking University
Beijing 100871, China

Editor: Isabelle Guyon, Dominik Janzing and Bernhard Schölkopf

Abstract

For a given target node T and a given depth $k \geq 1$, we propose an algorithm for discovering a local causal network around the target T to depth k. In our algorithm, we find parents, children and some descendants (PCD) of nodes stepwise away from the target T until all edges within the depth k local network cannot be oriented further. Our algorithm extends the PCD-by-PCD algorithm for prediction with intervention presented in Yin et al. (2008). Our algorithm can construct a local network to depth k, has a more efficient stop rule and finds PCDs along some but not all paths starting from the target.

Keywords: Causal network, Local structural learning

1. Introduction

In some applications, we may be interested in discovering a local causal network around a target variable rather than the whole network over all variables. For example, we want to predict the target in the cases with external interventions, or we are interested in direct and indirect causes of a disease and further discriminate direct causes from other indirect causes. There are many algorithms for structural learning, but most of them are for constructing a whole network over all variables, such as Pearl (2000); Spirtes et al. (2000); Heckerman (1999); Tsamardinos et al. (2006); Xie et al. (2006) and Xie and Geng (2008). To discover a local causal network, however, it is inefficient to construct the whole network over a large number of variables. In Causation and Prediction Challenge of IEEE WCCI2008, Yin et al. (2008) proposed local structural learning approaches for prediction with external interventions, in which only edges connecting to the target are discovered and oriented. But it cannot be used to discover a larger local structure or more indirect causes of the target.

In this paper, for a given target node T and a given depth $k \geq 1$, we propose an algorithm for discovering a local causal network around the target T to depth k. Our algorithm extends the PCD-by-PCD algorithm for prediction with intervention presented in Yin et al. (2008). First our

algorithm can construct a depth k local network, and Yin's PCD-by-PCD algorithm is a special case of the depth 1. Second, our algorithm has a more efficient stop rule than Yin's algorithm. In Yin's algorithm, a main stop condition is 'until all edges connecting the target are oriented', but in our algorithm, we make this condition weaker so that our algorithm can stop earlier than Yin's algorithm without loss of validity. Third, our algorithm continues to find PCDs along only some paths away from the target which are necessary to orient the undirected edges within the depth k local network, while Yin's algorithm continues to find PCDs along all paths starting from the target.

In Section 2, we propose the local structural learning algorithm. In Section 3, we theoretically show the correctness of our algorithm. Section 4 gives definitions of scores to be used for evaluation of algorithm performance. In Section 5, we compare our algorithms with other algorithms via simulation. We discuss the challenge task: LOCANET in Section 6. Discussion is given in Section 7. Proof of theorem is shown in Appendix.

2. Learning a local structure around the target to a given depth

Let U denote the full set of all nodes. For a node u, let $PC(u)$ denote the set of all parents and all children of u, and let $PCD(u)$ denote a set which contains $PC(u)$ and may contain some descendants of u. There are several algorithms which can be used to find $PCD(u)$, such as Min Max Parent and Children (MMPC) algorithm (Tsamardinos et al., 2006).

Let T be the target node. Suppose that we are interested in the local network around the target T to a depth k. In our algorithm, we first find parents, children and some descendants (PCD) of the target T to obtain a local skeleton with a radius 1, and then repeatedly find PCDs of nodes in the previous PCDs until the radius of the local skeleton is up to the given depth k. To orient the edges in the local skeleton, we may need to find more PCDs further away from the target T along some but not all paths. We expect to orient all undirected edges within the local network, but some of the undirected edges cannot be oriented essentially from observational data even if we construct a correct global network, which is an equivalence class of causal networks. Thus we propose a stopping rule so that the process of finding PCDs can stop early even if some edges within the local network are unoriented. Our stopping rule is based on the fact that when the unoriented edges are surrounded by directed edges, they cannot be oriented by finding further structures. We theoretically show that our algorithm can correctly obtain the local causal network with the given depth. Our algorithm does not need to construct the global network and thus it can greatly reduce computational complexity of structural learning.

In the following algorithm, we separate the process into two parts. Part 1 is to find edges within length $k-1$ from the target T. Part 2 is to find edges at the outer layer k and to orient undirected edges within the local network with depth k. When $k=1$, we only need to run Part 2 but no need to run Part 1.

Algorithm 1: Local structural learning around the target T to depth k

Part 1: Find edges within length $k-1$ from the target T.

1 **Initialization:** Find the PCD of T, $PCD(T)$.
$V = \{T\}$ (V is a set of variables whose PCDs have been obtained)
$Layer(0) = \{T\}$, $Layer(i) = \emptyset$ for $i = 1, \ldots k$ ($Layer(i)$ is the node set on layer i)
$TotalLay = \{T\}$, (The total set of nodes on all layers)
$depth = 1$, (the counter of depth)
$canU(1) = PCD(T)$, $canU(i) = \emptyset$ for $i = 2, \ldots, k$;
($canU(i)$ is an ordinal waiting list for layer i whose PCD will be found.)
Repeat
2 Take X from the head of list $canU(depth)$ out.
3 If $X \notin V$ (i.e., $PCD(X)$ has not been gotten before) then
Find $PCD(X)$, and set $V = V \cup \{X\}$.
For each $Y \in V$, if [$X \in PCD(Y)$ and $Y \in PCD(X)$],
then create an undirected edge (X, Y).
Find v-structures within V including X:
{Within V, find possible v-structures only for the triple of X and other two variables in V if an intermediate node is not in the separator set of two nonadjacent nodes.}
Orient undirected edges within V:
{Orient other edges between nodes in V if each opposite of them creates either a directed cycle or a new v-structure (Meek, 1995).}
End if
4 If $X \notin TotalLay$ and $X \notin Layer(depth)$ and X is adjacent to
a node in $Layer(depth-1)$ then
$Layer(depth) = Layer(depth) \cup \{X\}$,
add $PCD(X) \backslash TotalLay$ to the tail of list $canU(depth+1)$
End if.
5 If $canU(depth) = \emptyset$ then
$TotalLay = TotalLay \cup Layer(depth)$ and $depth = depth + 1$
End if
6 **Until** $canU(depth) = \emptyset$ or $depth \geq k$.

Algorithm 1 (continued)

Part 2: Find edges at layer k and orient undirected edges within the local structure.

Initialization. Set the list of nodes at the layer $k-1$ whose PCDs will further be found:
$canV = \{struct('leaf', v, 'length', 1, 'path', u) : u \in Layer(depth = k-1), v \in PCD(u) \backslash TotalLay\}$
Repeat
Take X from the head of the list $canV$ out.
If all edges on path X.path are undirected then
  If $X.leaf \notin V$ then
    find $PCD(X.leaf)$ and set $V = V \cup \{X.leaf\}$;
    for each $Y \in V$, if $X.leaf \in PCD(Y)$ and $Y \in PCD(X.leaf)$,
      then create an undirected edge (X.leaf,Y);
    find v-structures within V including $X.leaf$;
    orient undirected edges within V.
  End if.
  If there is an undirect edge between X.leaf and the last node u of $X.path$ then
    add $\{struct('leaf', v, 'length', X.length+1, 'path', [X.path, X.leaf])$
    $: v \in PCD(X.leaf) \backslash X.path \backslash TotalLay\}$ to the tail of $canV$
  End if.
End if.
Until $canV = \emptyset$.
Return

Example. Consider the revised ALARM network in Fig. 1 where the arrows $16 \rightarrow 20$ and $25 \rightarrow 20$ in the original ALARM (Beinlich et al., 1989) are reversed as $16 \leftarrow 20$ and $25 \leftarrow 20$ respectively. The revision makes more edges unoriented in its Markov equivalence class and thus it becomes more complicated for structural learning. Suppose that we want to discover a local network around node 20 to depth 2, denoted as $G_2(20)$. Applying Part 1 of our algorithm, we obtain a local network with all edges undirected as shown in Fig. 2. Applying Part 2, we first obtain the local network with $depth = 2$ in Fig. 3. Since there are some edges unoriented, we extend the network along undirected paths to orient these undirected edges. Finally Part 2 returns a local network as shown in Fig. 4, which is larger than $G_2(20)$ and has four nodes 9, 13, 19 and 21 outside $G_2(20)$. Nodes 19 and 21 are used to find two v-structures $19 \rightarrow 15 \leftarrow 18$ and $21 \rightarrow 17 \leftarrow 16$ respectively, and thus they help to orient edges $17 \leftarrow 16$ and $15 \leftarrow 18$ within $G_2(20)$. Nodes 9 and 13 are used to find a v-structure $13 \rightarrow 9 \leftarrow 14$ such that all undirected edges within $G_2(20)$ are surrounded by directed edges, and thus the algorithm stops.

3. Theoretical result for algorithm's correctness

We show below the correctness of the algorithm proposed in the previous section.

Theorem 1 *Suppose that a causal network is faithful to a probability distribution and all conditional independencies are correctly checked by using data. Then the algorithm proposed in the previous section can correctly discover the edges within the depth k local causal network around the target variable T. Further it can obtain the same orientations of these edges as a partially directed graph for the Markov equivalence class of the underlying global causal network.*

The proof of this theorem is given in Appendix. Under the suppositions of the faithfulness and correctness of conditional independence tests, the above result ensures that our algorithm

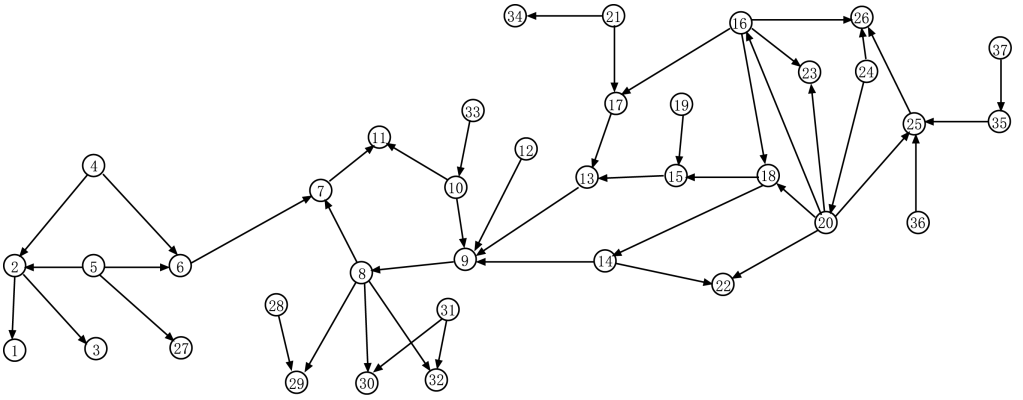

Figure 1: A revised ALARM

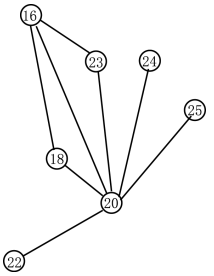

Figure 2: Network by Part 1

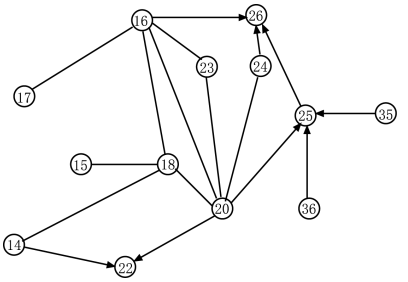

Figure 3: Network to $depth = 2$ by Part 2

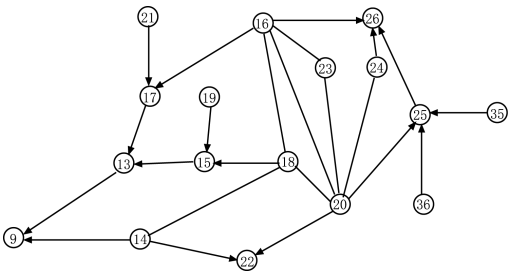

Figure 4: Network returned by Part 2

can return the correct local network. Notice that some edges in the local network may not be oriented. It is because these edges cannot be oriented by using data from observational studies, but it is not because our algorithm does not finish the learning process of the whole network.

4. The scores for evaluation

In this section, we introduce the two kinds of evaluation methods that are used in the causal challenge to evaluate the performance for discovering a local causal network (Guyon et al., 2008). The first method uses the average edit distance score. In the causal challenge, the task is to construct a depth 3 causal network around a given target variable. Thus the relationship of a variable to the target variable is encoded as a string of up (u) and down (d) arrows from the target:

- Depth 1 relatives: parents (u) and children (d);
- Depth 2 relatives: spouse (du), grand-children (dd), siblings (ud), grand-parents (uu); and
- Depth 3 relatives: great-grand-parents (uuu), uncles/aunts (uud), nices/nephews (udd), parents of siblings (udu), spouses of children (ddu), parents in law (duu), children of spouses (dud), great-grand-children (ddd).

A confusion matrix $C = \{C_{ij}\}$ is defined to record the number of relatives confused for another type of relative among 14 types of relatives in a depth 3 network. A cost matrix $A = \{A_{ij}\}$ is defined to account for the distance between the true and obtained relatives, as shown in Table 1. The edit distance score is defined as

$$S = \sum_{i,j} A_{ij} C_{ij}.$$

Depth	Desired		1	1	2	2	2	2	3	3	3	3	3	3	3	3	X
Obtained	Relationship		P	C	Sp	GC	Si	GP	GGP	uud	N	PS	SC	IL	CP	GGC	Other
			u	d	du	dd	ud	uu	uuu	uud	udd	udu	ddu	duu	dud	ddd	
1	Parents	u	0	1	1	2	1	1	2	2	2	2	2	2	2	3	4
1	Children	d	1	0	1	1	1	2	3	2	2	2	2	2	2	2	4
2	Spouse	du	1	1	0	1	2	1	2	2	2	1	1	1	1	2	4
2	Gchildren	dd	2	1	1	0	1	2	3	2	1	2	1	2	1	1	4
2	Siblings	ud	1	1	2	1	0	1	2	1	1	1	2	2	1	2	4
2	Gparents	uu	1	2	1	2	1	0	1	1	2	1	2	1	2	3	4
3	Ggparents	uuu	2	3	2	3	2	1	0	1	2	1	2	1	2	3	4
3	Uncles/Aunts	uud	2	2	2	2	1	1	1	0	1	2	3	2	1	2	4
3	Nieces/Nephews	udd	2	2	2	1	1	2	2	1	0	1	2	3	2	1	4
3	ParentsOfSiblings	udu	2	2	1	2	1	1	1	2	1	0	1	2	2	2	4
3	SpousesOfChildren	ddu	2	2	1	1	2	2	2	3	2	1	0	1	2	1	4
3	ParentsInLaw	duu	2	2	1	2	2	1	1	2	3	2	1	0	1	2	4
3	ChildrenOfSpouses	dud	2	2	1	1	1	2	2	1	2	2	2	1	0	1	4
3	GgChildren	ddd	3	2	2	1	2	3	3	2	1	2	1	2	1	0	4
X	Other		4	4	4	4	4	4	4	4	4	4	4	4	4	4	0

Table 1: A cost matrix $A = \{A_{ij}\}$.

The second method uses a score-pair (precision, recall) for each kind of variable subsets: parents, children, Markov blanket, all depth 1 variables, all depth 2 variables, all depth 3 variables. Precision and recall (also called sensitivity) are defined respectively as:

- Precision = # of true positive found/# of found, and
- Recall = # of true positive found/ # of true positive.

In the cases with a 0 denominator, a very small number are added to both the numerator and the denominator.

5. Simulation

In this section, we compare the algorithm proposed in this paper with other algorithms via simulations. Consider again the example in Section 2 and the goal is to get the depth 3 network around node 20. In Table 2, we show the simulation results for the revised ALARM network depicted in Fig. 1. We compare our algorithm (PCD-path) with the PC algorithm, the MMHC algorithm proposed by Tsamardinos et al. (2006) and the recursive algorithm proposed by Xie and Geng (2008). The 'distscore' is the edit distance score defined for the task LOCANET to measure the difference between the obtained local network and the true local network. We consider several cases with different significance levels and different sample sizes. In the simulation, we do 1000 repetitions and obtain average values for each case of different sample size n and significance level α. For each repetition, we draw a training data set from the distribution whose parameters for the unchanged structures are obtained from the FullBNT code package: http://www.cs.ubc.ca/ murphyl/Software/BNT/bnt.html, and parameters for the changed structure are set by chance. All of our computations are performed on a computer with CPU 2.1 GHz$\times$2 and 2 GB RAM. 'CPU time' is the total CPU time of 1000 repetitions for each algorithm. It can be seen from Table 2 that our algorithm takes much less CPU times and it has also less distscores than other three algorithms for every case.

In Table 3, we give the total (precision, recall) scores in 1000 simulations and the average scores can be obtained by dividing it by 1000. By the (precision, recall) scores, there is no algorithm which always is better than others. The Recursive one seems to be better averagely. In some cases, the PCD-path algorithm seems to be better at 'pa' and 'ch' than the MMHC algorithm, but worse at 'pc' and 'mb'. From Tables 2 and 3, it can be seen that the PCD-path algorithm proposed in this paper runs fastest among the four algorithms without loss of performance. The main advantage of the PCD-path algorithm is to construct a local network around the given target, and this is more important for the cases with a large number of variables.

6. Challenge task LOCANET

We applied the algorithm to three data sets: REGED, CINA and MARTI to find local structures around targets to depth 3. The data set MARTI is preprocessed in the way proposed by Yin et al. (2008), which is available at

$$http://clopinet.com/isabelle/Projects/WCCI2008/MARTI/JY/$$

We summarize our results for the Potluck challenge task LOCANET in Table 4. 'NoNode' denotes the number of nodes for a data set, 'NoNodeLN' denotes the number of nodes in the local network around the target to depth 3, 'NoPcds' denotes the number of PCDs found by our algorithm, and we give CPU times for every data set. Our algorithm takes so much longer on the dataset CINA than other datasets. First, the sample size of CINA is much larger than

n	α	Algorithm	distscore	CPU time (second)
500	0.05	PCD-path	1.0305	881
		Recursive	1.1635	3,293
		MMHC	1.1162	3,405
		PC	1.2213	11,638
	0.10	PCD-path	1.0993	1,175
		Recursive	1.2057	3,404
		MMHC	1.1692	3,515
		PC	1.3150	11,979
	0.15	PCD-path	1.1621	1,489
		Recursive	1.2498	3,509
		MMHC	1.1919	3,949
		PC	1.4083	12,457
1000	0.05	PCD-path	0.8573	993
		Recursive	1.1205	3,739
		MMHC	1.1133	3,864
		PC	1.0804	8,823
	0.10	PCD-path	0.8836	1,204
		Recursive	1.2958	3,889
		MMHC	1.1431	4,326
		PC	1.1241	9,635
	0.15	PCD-path	0.9082	1,415
		Recursive	1.3702	4,008
		MMHC	1.1724	4,823
		PC	1.1508	10,440

Table 2: Comparison of algorithms for the revised ALARM network.

REGED and MARTI. Second, the independence tests of discrete variables for CINA runs much slower than the tests of continuous variables for REGED and MARTI under the assumption of Gaussian distribution. For the data set SIDO, there are 4933 variables, the observed frequencies are very unbalanced, some cells have very small frequencies, and some have very large ones. In this case, the approach for finding parents and children sets is not so efficient as the approach for finding Markov blankets. Thus the recursive algorithm via Markov blankets proposed by Xie and Geng (2008) is used to find a local networks including the target and 400 nodes which are strongly associated with the target.

The (precision, recall) scores and the edit distance scores of our performance on the four datasets of LOCANET are shown in Figure 5. For the (precision, recall) scores, the more the symbols are in the upper right corner, the better the performance is. We have about 7 symbols in the upper right quadrant. Most of the symbols in the lower left corner are for the MARTI dataset which are generated by adding noises to the dataset REGED. The performance for the dataset REGED is quite better, and thus the noises in MARTI may not be filtered throughout in our algorithm. Since there are no known parents in the CINA task, it is not surprising that our result for the parents in the CINA is on the vertical axis (which means we have a recall 1 while precision 0). Thirteen in total 24 symbols are close to the right boundary which presents a high precision, and this means that the results we found are mostly true.

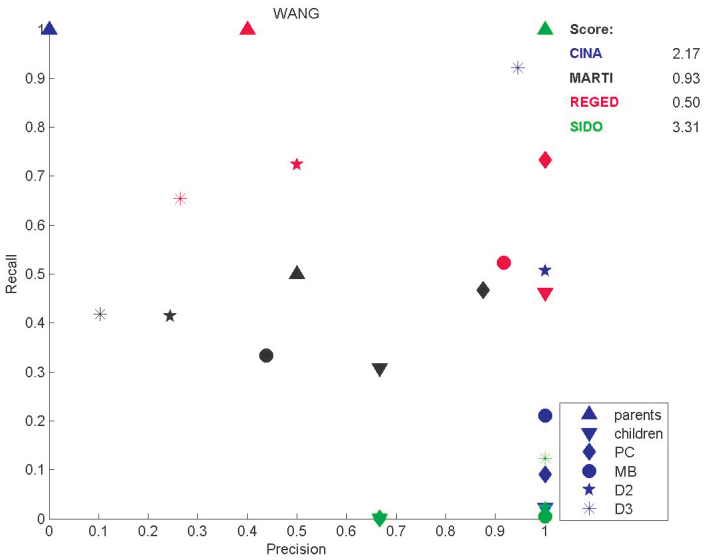

Figure 5: (Precision, Recall) scores and edit-distance scores for four datasets of LOCANET.

n	α	Algorithm	precision						recall					
			pa	ch	pc	mb	D2	D3	pa	ch	pc	mb	D2	D3
500	0.05	PCD-path	41	883	904	873	755	669	93	364	713	640	686	402
		Recursive	282	747	971	824	751	692	971	705	780	876	788	573
		MMHC	50	768	944	933	833	777	21	652	928	887	909	656
		PC	41	932	836	822	823	901	222	441	861	839	713	377
	0.10	PCD-path	145	836	894	853	812	780	307	472	836	751	782	502
		Recursive	262	750	967	822	750	692	934	717	789	880	802	597
		MMHC	56	749	943	935	829	738	28	651	945	900	918	688
		PC	74	882	815	818	823	891	426	393	921	876	728	435
	0.15	PCD-path	174	821	879	839	816	774	409	519	882	796	809	542
		Recursive	246	750	962	816	742	676	893	725	794	880	806	605
		MMHC	64	739	941	933	818	707	35	657	955	908	925	703
		PC	80	848	804	808	806	846	485	378	949	893	735	469
1000	0.05	PCD-path	263	968	990	894	847	657	68	652	643	754	814	576
		Recursive	196	782	961	819	862	840	812	866	894	969	883	681
		MMHC	55	740	986	958	909	804	23	574	911	873	932	769
		PC	49	952	990	810	931	822	190	701	767	817	861	534
	0.10	PCD-path	315	911	983	900	895	780	160	722	776	824	905	688
		Recursive	169	765	935	815	845	813	755	872	904	965	863	689
		MMHC	71	748	980	953	904	801	30	617	943	914	956	804
		PC	88	906	980	802	943	882	359	759	847	879	911	627
	0.15	PCD-path	318	893	974	899	898	811	174	745	826	853	930	737
		Recursive	158	753	917	806	822	774	737	877	911	962	850	693
		MMHC	73	751	976	950	892	785	32	639	959	936	965	814
		PC	99	884	970	796	942	907	422	781	890	906	925	670

Table 3: Precision and recall of algorithms for the revised ALARM network.

Data set	NoNodes	NoNodeLN	NoPcds	CPU time
REGED	1000	136	212	10 minutes
CINA	133	108	116	4 hours
MARTI	1025	224	309	10 minutes

Table 4: Results for the challenge task LOCANET.

7. Conclusion

We proposed an algorithm for local structural learning of a causal network around a given target node to depth k. Our algorithm finds PCDs stepwise starting from the target node and stops the process when the local structure is obtained, and thus it can reduce the computational complexity. We theoretically show its correctness. The algorithm can be used for prediction with external interventions and for local causal discovery.

Acknowledgments

We would like to thank the four reviewers for their valuable comments and suggestions. We would appreciate I. Guyon and the competition committee for their encouragement and support

to our work. This research was supported by NSFC (10771007, 10721403), 863 Project of China (2007AA01Z437), MSRA and MOE-Microsoft Key Laboratory of Statistics and Information Technology of Peking University.

References

I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the 2nd European Conference in Artificial Intelligence in Medicine*, pages 247–256, Germany, 1989. Springer-Verlag.

I. Guyon, A. Statnikov, and C. Aliferis. Pot-luck challenge: FACT SHEET. Technical Report.

D. Heckerman. A tutorial on learning with bayesian networks. In M. Jordan, editor, *Learning in Graphical Models*, Cambridge, MA, 1999. MIT Press.

C. Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 403–41. Morgan Kaufmann, 1995.

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

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

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

X. Xie and Z. Geng. A recursive method for structural learning of directed acyclic graphs. *Journal of Machine Learning Research*, 9:459–483, 2008.

X. Xie, Z. Geng, and Q. Zhao. Decomposition of structural learning about directed acyclic graphs. *Artificial Intelligence*, 170(4):422–439, 2006.

J. Yin, Y. Zhou, C. Wang, P. He, C. Zheng, and Z. Geng. Partial orientation and local structural learning of causal networks for prediction. In *JMLR: Workshop and Conference Proceedings*, volume 3, pages 93–104, WCCI2008 workshop on causality, Hong Kong, June 3-4 2008.

Appendix

In this appendix we prove Theorem 1 presented in Section 3.

Proof We first show the correctness of Part 1. Step 1 is initialization. We take X from list $canU(depth)$ to find PCD(X) at step 2. At Step 3, we obtain an undirect edge $X-Z$ if and only if both $Z \in PCD(X)$ and $X \in PCD(Z)$, and thus all edges can be created correctly if independence tests for finding PCD are correctly performed. After finding a new edge connecting node X newly taken at Step 2, we can determine whether there is a v-structures with X as one node and other two nodes in V, such as $X-Y-Z$ with X and Z nonadjacent and all X, Y and Z in the set V. It is because X, Y and Z are all in the set V, and edges between them have been correctly determined. We can correctly find a v-structure $X \to Y \leftarrow Z$ if Y is not in the separator X and Z (that is, $X \perp\!\!\!\perp Z|S$ and $Y \notin S$). Note that the separator S has been obtained during finding $PCD(X)$

if $Z \notin PCD(X)$ or during finding $PCD(Z)$ if $X \notin PCD(Z)$. After finding a new v-structure or adding a new undirected edge, we need to orient again undirected edges within V using Meek's rules.

At Step 4, we add X to $Layer(depth)$ because X is adjacent to a node in $Layer(depth-1)$ and not in the previous layers. Thus $Layer(depth)$ can correctly be formed if $Layer(depth-1)$ was correctly formed. Nodes in $PCD(X) \setminus TotalLay$ are added to the list $canU(depth+1)$ as candidate nodes in the next layer. Thus we can make sure all nodes which have length $depth+1$ from T in $canU(depth+1)$ if $Layer(depth)$ are correct. Inductive, we showed the correctness of $Layer(i)$ for all i since at the initiation step, $Layer(0) = T$ is correctly set.

At step 5, we obtain the final $Layer(depth)$, add it to $TotalLay$ and add 1 to $depth$ after we treated all nodes in $canU(depth)$.

Finally, we stop Part 1 if (1) all nodes having a path to T have a distance shorter than k or (2) the first $k-1$ Layers have been obtained.

Next we show the correctness of Part 2. In Part 2 of the algorithm, we sequentially search nodes outside $TotalLay$ along an undirected path starting from a node in $Layer(k-1)$ through finding PCDs of the terminal node of the path until a directed edge is found. By Part 2, we obtain a network G which covers the local network $G_k(T)$ we want to find. The network G has mixed types of directed and undirected edges and has directed edges as its boundary. Define A as a set which contains all undirected edges and the first $k-1$ layers in the local network finally obtained by the algorithm, that is, $A = \{u \in V : u$ has an undirected path starting from a node in $Layer(k-1)\} \cup TotalLay$. Then any edge (u,v) which connects a node $u \in A$ and a node $v \notin A$ must be a directed edge otherwise v should be contained in A. Define B as a set of nodes which surrounds A, that is, $B = \{v \in PCD(u) \setminus A : u \in A\}$. Define E as a set of edges each of which has at least one node in A, that is, $E = \{(u,v) : u \in A,\ v \in A \cup B\}$. We can have that all undirected edges within E cannot be oriented even if the global network is obtained. It is because any undirected (u,v) in E must have both of its two nodes u and v contained in A and all undirected edges in E must be surrounded by directed edges. Thus, if these undirected edges cannot be oriented by applying Meek's rules to E, then they cannot still be oriented by finding more edges outside E. ■