Improving the Efficiency of the PC Algorithm by Using Model-Based Conditional Independence Tests

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning causal structure is useful in many areas of artificial intelligence, such as planning, robotics, and explanation. Constraint-based and hybrid structure learning algorithms such as PC use conditional independence (CI) tests to learn a causal structure. Traditionally, constraint-based algorithms perform the CI tests with a preference for smaller-sized conditioning sets, partially because the statistical power of conventional CI tests declines substantially as the size of the conditioning set increases. However, many modern conditional independence tests are *model-based*, and these tests use well-regularized models that can perform well even with very large conditioning sets. This suggests an intriguing new strategy for constraint-based algorithms which may result in a reduction of the total number of CI tests performed: Test variable pairs with *large* conditioning sets *first*, as a pre-processing step that finds some conditional independencies quickly, before moving on to the more conventional strategy of testing with incrementally larger conditioning sets of sizes (beginning with marginal independence tests). We propose such a pre-processing step for the PC algorithm which relies on performing CI tests on a few randomly selected large conditioning sets. We perform an empirical analysis on directed acyclic graphs (DAGs) that correspond to real-world systems and both an empirical and theoretical analysis for Erdős-Renyi DAGs. Our results show that the PC algorithm with our pre-processing step performs far fewer CI tests than the original PC algorithm, between 0.5% and 20%, of the CI tests that the PC algorithm alone performs. The efficiency gains are particularly significant for the DAGs corresponding to real-world systems.

1 Introduction

Causal models are useful for representing and reasoning about causal relationships in many areas of artificial intelligence and machine learning. One way to represent causal models is as a directed graphical model, which consists of a causal structure in the form of directed edges between variables and parameters in the form of a conditional probability distribution corresponding to each variable. The edges in the directed structure convey the existence and direction of causal relationships, and the parameters specify the magnitude of their causal effects. To construct a causal model given a data sample, the first step is to learn a graph structure and the second is to use that graph structure for estimating parameters.

Constraint-based algorithms are a type of structure learning algorithm that infer a structure that corresponds to the conditional independence (CI) relationships within a population. These conditional independence relationships are inferred by performing CI tests on a data sample. In this work, we focus on constraint-based algorithms, improving their efficiency for causal structure learning.

Submitted to the NeurIPS Causal ML for Real-World Impact Workshop (CML4Impact 2022). Do not distribute.

The phase of constraint-based algorithms that applies CI tests typically dominates the runtime of such algorithms, and the number of such tests in the worst case is equal to the cardinality of the power set of the possible conditioning variables. Specifically, constraint-based algorithms run CI tests for each pair of variables and possible conditioning sets, stopping only when a test infers that the pair is conditionally independent or after all feasible CI tests have been run. As the number of variables increases, the number of CI tests these algorithms run increases exponentially.

Most constraint-based algorithms assume that CI tests perform more accurately on small conditioning sets. However, recent model-based CI tests have the potential to work effectively on very large conditioning sets. The models typically used in those tests are well-regularized, and so they can produce accurate results even with very large numbers of conditioning variables. Several experimental evaluations compare their performance with that of other CI tests and find that they perform more accurately than the older CI tests, particularly on larger conditioning sets. Substantial recent work on CI tests suggests that their accuracy and efficiency is likely to improve.

The focus of this paper is on how such current and future tests could be used to improve the efficiency of constraint-based algorithms for structure learning. We assume that accurate and general model-based CI tests exist or will exist soon. We explore how algorithms could make use of model-based CI tests, presenting a pre-processing algorithm for the PC algorithm which relies on randomly selecting a few large conditioning sets and running model-based CI tests on those sets for every pair of variables. The aim of the pre-processing algorithm is to speed up the process of identifying pairs of variables that are conditionally independent. We provide theoretical and empirical evidence that the pre-processing algorithm is extremely successful at this task.

The core idea that makes the pre-processing algorithm useful is to perform CI tests on *large* conditioning sets. Large conditioning sets are often avoided when designing algorithms for structure learning due to a concern that such sets could easily include one or more colliders. Conditioning on a collider can induce dependence between a pair of variables that would otherwise be independent. However, we show that the probability of conditioning on a collider in an otherwise unblocked path is very small when the size of the conditioning set is large. This enables conditioning on large variable sets to very quickly identify large numbers of conditional independence relations.

We show that induced dependence due to conditioning on a collider is very rare when conditioning sets are large. Further, we observe that failing to correctly infer every conditional independence in the pre-processing step is not a major problem, because it will be caught in the next phase of the algorithm, and that finding many CIs early greatly reduces the total number of tests. We conduct a theoretical analysis and an experimental evaluation to explore the efficiency gain between the PC algorithm with pre-processing and the PC algorithm without such pre-processing. We conduct the experiments on both random Erdős-Renyi networks and networks that correspond to real-world systems. We find that the PC algorithm with pre-processing typically performs only 0.5% to 20% of the CI tests that the PC algorithm alone performs on networks that correspond to real-world systems.

2 Related Work

A wide variety of work has focused on how to improve the accuracy and efficiency of structure learning algorithms, including work on constraint-based algorithms [1], score-based algorithms, continuous-optimization algorithms [2], and combinations of these approaches [3]. Advances in accuracy and efficiency for continuous-optimization algorithms have been particularly prominent recently [4], turning some structure learning into combinatorial optimization and convex optimization problems [5]. Other advances in efficiency have included analyzing properties of scoring criterions such as BIC and AIC [6].

In contrast, in this paper we focus on improving the efficiency of constraint-based algorithms, particularly the PC algorithm [7]. The PC algorithm operates by performing CI tests in order of increasing size of conditioning sets; i.e. on empty, size 1, size 2, and larger conditioning sets, removing edges from a complete undirected graph once they discover that a conditioning set could make a pair of variables CI, and then orienting edge directions. Several avenues exist for improving the efficiency of the PC algorithm. First, some work has focused on reducing the number of CI tests that need to be performed [8, 9, 10, 11, 12]. Second, some work has focused on improving the PC algorithm in specific applications where the general algorithm performance is weaker, such as for partially faithful distributions in high-dimensional linear models, when the data contains heavy

outliers, and when the data is ordinal [13, 14, 15, 16]. Third, some work has focused on improving the hardware through GPU use and parallelization [17, 18, 19, 9, 20, 21, 22, 23].

However, all of the above techniques for constraint-based algorithms rely on the fact that the PC algorithm must prioritize performing CI tests with small conditioning sets. The literature suggests that the older CI tests do not perform well on larger conditioning sets and that mistakes early on in the PC algorithm severely affect the structure learned by the PC algorithm since early errors propagate forward [7].

Although older CI tests may not be very accurate specifically on larger conditioning sets, new conditional independence tests continue to be developed. Some new tests are model-based [24], which do not suffer from the issue of only performing well on smaller conditioning set sizes. One example is the Classifier Conditional Independence Test (CCIT) which uses the conditioning set such that its variables are predictor variables for a model [25]. In such a case, the CI test using a large conditioning set may not necessarily produce less accurate results than that using a small conditioning set. Further, many CI tests are fast and are continuously improved to be faster. Some very quick CI tests such as the Randomized Conditional Independence Test (RCIT) [26] are improvements to CI tests that were too slow to be applied such as the Kernel-Based Conditional Independence Test (KCIT) [27]. Other CI tests such as CCIT are not almost instantaneous, but also not intractable. Further, interest has even extended to conditional dependence measures, which provide a more accurate estimate of the extent of conditional dependence. Given these new improvements and work, we could assume that even if a CI test does not meet a standard of quickness and accuracy, such a test will be developed in the near future.

3 Pre-processing Algorithm

We propose a pre-processing algorithm for the PC algorithm that leverages the use of CI tests that perform accurately and efficiently on large conditioning sets:

Algorithm 1: Pre-processing algorithm for the PC algorithm

Input : $X = [\mathbf{x}_1, \mathbf{x}_2, , \mathbf{x}_n]$: a dataset where $\mathbf{x}_i$ is a vector containing data samples for the ith variable and n is the number of variables in the dataset; V: the set of variables

Output : $p = [v_1, v_2, \ldots, v_n]$: an $n \times n$ matrix where index $[i, j]$ is 0 if the pair of variables could be made conditionally independent and 1 otherwise

```
Initialize p as an n × n matrix with all 1 values.
Initialize L as a list of sets having length c_1.
for a, b ∈ V do
    if a ⊥ b then
        Set p[a, b] = 0 and p[b, a] = 0; go to the next iteration of the for loop
    for i ∈ {1, ..., c_1} do
        Append a random set of n − c_2 variables not including a and b to list L
    for S ∈ L do
        if a ⊥ b | S then
            Set p[a, b] = 0 and p[b, a] = 0; go to the next iteration of the for loop
Return p
```

The pre-processing algorithm aims to inform the PC algorithm about pairs which could possibly be made conditionally independent, before it runs. Such knowledge decreases the number of CI tests that the PC algorithm performs as indicated by Section 4. For each pair of variables, the algorithm randomly selects c_1 conditioning sets of size $n - c_2$ where c_1 and c_2 are both near-zero integers. For the derivations in Section 4, we set $c_1 = 3$ and $c_2 = 4$. Next, the algorithm aims to perform CI tests on the empty set and on each of the c_1 conditioning sets to determine if we could tell whether a pair could be made conditionally independent.

The pre-processing algorithm we propose relies on tests that are accurate on large conditioning sets (e.g., a model-based CI test). However, after pre-processing, the subsequent application of the PC algorithm still uses traditional CI tests. Although fast model-based CI tests exist, given the current uncertain speed of model-based CI tests in comparison to traditional CI tests, the most efficient algorithm is likely to use model-based CI tests only where necessary, which is when the conditioning set size is large (in the pre-processing algorithm).

4 Theoretical Analysis for Erdős-Renyi Networks

Definitions. We define a *trail* of length ℓ between nodes u_0 and u_ℓ as a sequence of ℓ distinct edges $e_1, \ldots, e_\ell$ where u_0 is an endpoint of e_1, u_ℓ is an endpoint of e_ℓ, and each e_i has one endpoint in common with e_{i-1} and the other endpoint in common with e_{i+1}. Note that the head (or "child" node) of e_{i-1} does not need to equal the tail (or "parent" node) of e_i. The node in common with e_i and e_{i+1} is a *collider* in the trail if it is the head of both the edges e_i and e_{i+1}. A trail is *blocked* by a conditioning set if at least one intermediate node on the trail that is not a collider is in the conditioning set or if at least one intermediate node on the trail that is a collider is not in the conditioning set. A *pair of variables is made CI* if all trails between the pair of variables are blocked by a conditioning set. The goal of the pre-processing algorithm is to detect if a pair of variables could be made CI or not.

Erdős-Renyi networks. For Erdős-Renyi networks where n nodes are labeled in increasing order and an edge exists with probability p only from nodes with a smaller label to nodes with a larger label, we provide indications that one randomly selected conditioning set of size $n-4$ would be able to determine whether a pair of variables could possibly be made CI. We index the nodes $v_1, v_2, \ldots, v_n$ where v_i is the node with the ith smallest label.

If a pair of variables could be detected as CI with respect to a conditioning set of size $n-4$, then all possible trails between the pair of variables are blocked with respect to the conditioning set. We indicate that the probability that all possible trails between a pair of variables are blocked with respect to the conditioning set is low by considering trails of length 7 and more and trails of length 6 and fewer separately.

Statement 1 (Trails of length at least 7). A conditioning set of size $n-4$ will block every trail of length 7 or greater. Note that this will be true for any network, not just Erdős-Renyi networks.

The reason for this is as follows. First note that for any trail of length at least 7, there are at least 3 intermediate nodes on the trail that are not colliders; this follows because colliders cannot be adjacent to each other and therefore if there are k intermediate nodes, at least $\lfloor k/2 \rfloor$ of them are non-colliders. Next consider a trail of length at least 7 between a pair of nodes a and b and a conditioning set $S \in \{v_1, \ldots, v_n\} \setminus \{a, b\}$ of size $n-4$. Since $\{v_1, \ldots, v_n\} \setminus (S \cup \{a, b\})$ only has size 2, it must be that S contains at least one of the three or more non-colliders on the trail and hence blocks the trail.

Statement 2 (Trails of length at most 6). In an Erdős-Renyi network, the expected number of trails of length ≤ 6 between a specific pair of nodes is at most $p(1 + (pn) + (pn)^2 + \ldots (pn)^6) \leq 7n^6p^7$ where the last inequality assumes $p \geq 1/n$. If $p = \Theta(1)/n$, corresponding to graphs with constant expected degree, this expectation tends to 0 as $n \to \infty$ and hence, by the Markov bound, the probability there exists such a trail tends to 0 as $n \to \infty$.

This statement can be proved as follows. There are $\binom{n-2}{k}$ choices for the set of k intermediate nodes on a trail of length $k+1$, there are $k!$ orderings of these nodes, and the probability there exists edges between each successive pair of nodes is p^{k+1}. Hence, the expected number of trails with k intermediate nodes is $k!\binom{n-2}{k}p^{k+1} < n^k p^{k+1}$. Summing from $k = 0$ to 6 gives the claimed result.

Finally we show that we expect most nodes to not be colliders for any trail. This implies that even when there are short trails, they will likely contain more non-colliders than colliders. And if the conditioning set includes at least one non-collider in a trail, then that trail is blocked.

Statement 3 (Expected number of colliders). The expected number of nodes that are colliders for some trail is

$$n(1-p)^n - (1-p)^n + 2((1-p)^n - 1)/p + n - p^2 + 1 \,.$$

For example, when $p = 1/n$, the expression corresponds to $0.104n$.

This is because the probability that v_i, the node with the ith largest label, is a collider is

$$P(\text{Bin}(i-1, p) \geq 2) = 1 - (1-p)^{i-1} - (i-1)p(1-p)^{i-2} \tag{1}$$

since the only way for a node v_i to be a collider on any trail is if are edges between it and two nodes of more nodes amongst $\{v_1, \ldots v_{i-1}\}$. Using linearity of expectation, the expected number of nodes that are colliders for some trail is $\sum_{i=3}^{n} P(\text{Bin}(i-1, p) \geq 2)$ and this simplifies to Eq. 1.

Takeaway. For a single pair of variables, one unblocked trail will prevent the pre-processing algorithm from correctly determining that the pair could be made CI. Yet, we show that one unblocked trail

of length 7 or more is impossible, that the expected number of trails of length $2, 3, 4, 5$, or 6 is very small, and that the expected number of nodes that are colliders for a trail is very small. Since the expected number of short trails and of colliders is small, the large conditioning sets will very likely catch some pairs that may take many CI tests to be concluded CI using the original PC algorithm.

5 Experiments

We performed experiments to estimate the raw number of CI tests performed by the PC algorithm with our pre-processing algorithm and without it. Our goal was to evaluate the effects of pre-processing on the total number of tests run by the algorithms. Thus, we conducted experiments using known DAG structure and standard rules of d-separation to determine whether variables could be made conditionally independent by a given conditioning set, rather than running conditional independence tests on data generated by that DAG.

The first set of experiments compared the raw number of tests performed by the PC algorithm with and without pre-processing on over 1,000 Erdős-Renyi networks randomly generated in the manner described in Section 4. The second set of experiments compared the same two algorithms on networks in the `bnlearn` repository, which are designed to correspond to real-world systems. For example, the *Water* network is from a technical report on an expert system for controlling waste water treatment [28]. The number of nodes in the Erdős-Renyi networks is smaller than that of the realistic networks in the `bnlearn` repository because we aimed to perform experiments on a large number of ER networks and inferring d-separation is time-consuming on larger, dense networks.

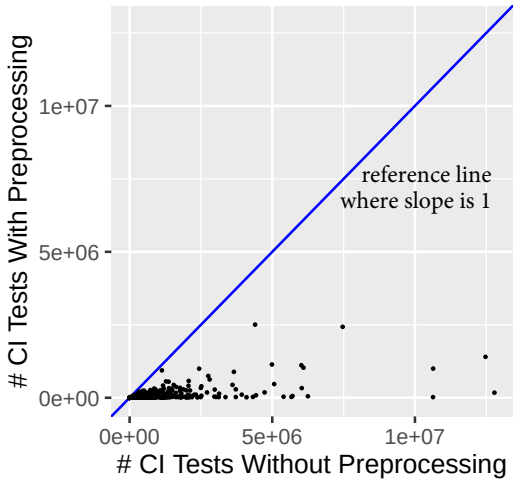

Figure 1: This plot shows the number of tests performed by the PC algorithm with pre-processing including the number of tests performed in the pre-processing step on the y-axis and the number of tests performed by the PC algorithm without pre-processing on the x-axis. Each point in the plot corresponds to a randomly generated ER network and its location depends on the number of tests performed by both versions of the PC algorithm, with and without pre-processing. The blue line is for reference and has a slope of 1.

	Child	Alarm	Mildew	Ecoli	Insurance	Water	Barley	Magic Niab	Mehra	Magic Irri
# Nodes	20	37	35	46	27	32	48	44	24	64
# Edges	25	46	46	70	52	66	84	66	71	102
Prop. Tests	0.361	0.143	0.055	0.285	0.073	0.021	0.006	0.008	0.137	0.005

Table 1: The number of nodes and edges for each DAG from the `bnlearn` repository that corresponds to real-world systems, and the proportion of tests performed by the pre-processing step and the PC algorithm combined to tests performed by the PC algorithm alone for each.

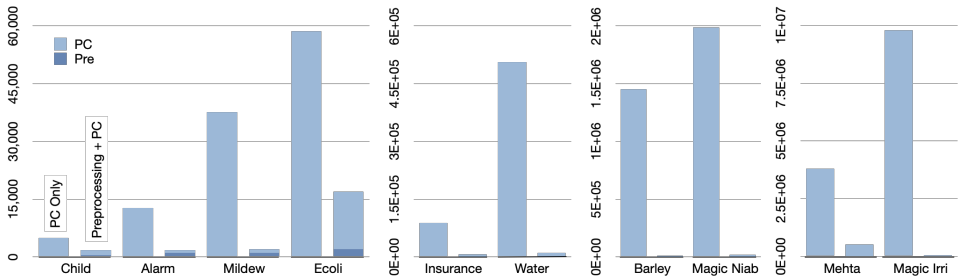

Figure 2: These plots show, for networks that correspond to real-world systems in the `bnlearn` repository, the number of tests that the PC algorithm with pre-processing performs to learn a structure and that the PC algorithm alone performs to learn a structure. The x-axis of the plots are the names of realistic networks, while the y-axis of the plots are the number of tests performed. Note that the y-axis varies for different networks. For all realistic networks, the raw number of tests performed by the PC algorithm with pre-processing is much less than that performed by the PC algorithm alone.

In both the networks that correspond to real-world systems and the Erdős-Renyi networks, the raw number of tests performed by the PC algorithm with our pre-processing step is much fewer than that performed by the PC algorithm alone as shown in Figure 1. In a further analysis, we observed that the efficiency gain is greater as the number of edges increases to be twice the amount of the number of nodes. The difference in performance of the PC algorithm with our pre-processing step and the PC algorithm alone is more noticeable for the networks that correspond to real-world systems, where the number of tests performed by the PC algorithm with pre-processing is so much less than that performed by the PC algorithm alone that the bar representing that number hardly shows up in Figure 2. Specifically, for the networks that correspond to real-world systems, the number of tests performed by the PC algorithm with our pre-processing step is typically between 0.5% and 20% of the number of tests performed by the PC algorithm alone as displayed in Table 1.

6 Conclusion

We proposed a pre-processing algorithm for the PC algorithm which can dramatically reduce the total number of CI tests that the PC algorithm performs. This implies a new reason to prioritize research on accurate model-based tests of conditional independence, particularly those that perform well with large conditioning sets. Furthermore, we show theoretically why the pre-processing algorithm reduces this number in expectation for Erdős-Renyi graphs. Finally, we perform experiments on networks that correspond to real-world systems, finding that the number of tests performed by the PC algorithm with pre-processing is typically 0.5% to 20% of the number of tests performed by the PC algorithm alone. From the experiments on Erdős-Renyi graphs, we observe that the efficiency gain is also large and increases as the number of edges increases to be twice the number of nodes.

This work addresses one of the primary challenges for routine use of the PC algorithm by greatly reducing runtime. This makes possible improved algorithms such as PC-stable, which repeats the steps in the PC algorithm many times, more tractable [29]. To further increase the efficiency of the pre-processing step, a system could parallelize the tests within the pre-processing algorithm for each pair since the tests for different pairs do not depend on each other.

References

[1] Facundo Bromberg, Dimitris Margaritis, and Vasant G Honavar. Efficient markov network structure discovery using independence tests. *The Journal of Artificial Intelligence Research*, 35 1:449–484, 2006.

[2] Matthew J. Vowels, Necati Cihan Camgoz, and Richard Bowden. D'ya Like DAGs? A Survey on Structure Learning and Causal Discovery. *ACM Computing Surveys*, mar 2022.

[3] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. *Machine Learning*, 65(1):31–78, oct 2006.

[4] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9492–9503, Red Hook, NY, USA, 2018. Curran Associates Inc.

[5] Su in Lee, Varun Ganapathi, and Daphne Koller. Efficient structure learning of Markov networks using L1regularization. In *Proceedings of the Twentieth Conference on Neural Information Processing Systems*, 2006.

[6] Rong Zhu, Andreas Pfadler, Ziniu Wu, Yuxing Han, Xiaoke Yang, Feng Ye, Zhenping Qian, Jingren Zhou, and Bin Cui. Efficient and Scalable Structure Learning for Bayesian Networks: Algorithms and Applications. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2613–2624, 2021.

[7] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*, volume 81. 01 1993.

[8] Ioannis Tsamardinos, Constantin Aliferis, and Alexander Statnikov. Algorithms for Large Scale Markov Blanket Discovery. *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference*, pages 376–381, 01 2003.

[9] Kai Zhang, Chao Tian, Kun Zhang, Todd Johnson, and Xiaoqian Jiang. A Fast PC Algorithm with Reversed-order Pruning and A Parallelization Strategy. arXiv, 2021.

[10] Arjun Sondhi and Ali Shojaie. The Reduced PC-Algorithm: Improved Causal Structure Learning in Large Random Networks. *Journal of Machine Learning Research*, 20(164):1–31, 2019.

[11] Enrico Giudice, Jack Kuipers, and Giusi Moffa. The Dual PC Algorithm for Structure Learning. arXiv, 2021.

[12] Xiaolong Qi, Xiaocong Fan, Huiling Wang, Ling Lin, and Yang Gao. Mutual-information-inspired heuristics for constraint-based causal structure learning. *Information Sciences*, 560:152–167, 2021.

[13] Flaminia Musella. A PC Algorithm Variation for Ordinal Variables. *Computational Statistics*, 28(6):2749–2759, dec 2013.

[14] Markus Kalisch and Peter Bühlmann. Robustification of the PC-Algorithm for Directed Acyclic Graphs. *Journal of Computational and Graphical Statistics - J COMPUT GRAPH STAT*, 17:773–789, 12 2008.

[15] Marcel Wienöbst and Maciej Liundefinedkiewicz. An Approach to Reduce the Number of Conditional Independence Tests in the PC Algorithm. In *KI 2021: Advances in Artificial Intelligence: 44th German Conference on AI, Virtual Event, September 27 – October 1, 2021, Proceedings*, page 276–288, Berlin, Heidelberg, 2021. Springer-Verlag.

[16] Peter Buhlmann, Markus Kalisch, and Marloes H. Maathuis. Variable selection in high-dimensional linear models: partially faithful distributions and the PC-simple algorithm. *Biometrika*, 97:261–278, 2010.

[17] Christopher Hagedorn and Johannes Huegle. GPU-Accelerated Constraint-Based Causal Structure Learning for Discrete Data. In Carlotta Demeniconi and Ian Davidson, editors, *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, pages 37–45. SIAM, 2021.

[18] Marco Scutari. Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package. *Journal of Statistical Software*, 77, 03 2017.

[19] Thuc le, Tao Hoang, Jiuyong Li, Lin Liu, and Huawen Liu. A Fast PC Algorithm for High Dimensional Causal Discovery with Multi-Core PCs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16, 02 2015.

[20] D Marbach, J C Costello, R Kuffner, N M Vega, R J Prill, D M Camacho, K R Allison, DREAM5 Consortium, M Kellis, J J Collins, and G Stolovitzky. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8):796–804, July 2012.

[21] Anders L. Madsen, Frank Jensen, Antonio Salmerón, Helge Langseth, and Thomas D. Nielsen. Parallelisation of the PC Algorithm. In *Proceedings of the 16th Conference of the Spanish Association for Artificial Intelligence on Advances in Artificial Intelligence - Volume 9422*, page 14–24, Berlin, Heidelberg, 2015. Springer-Verlag.

[22] Anders Madsen, Frank Jensen, Antonio Salmerón, Helge Langseth, and Thomas Nielsen. A Parallel Algorithm for Bayesian Network Structure Learning from Large Data Sets. *Knowledge-Based Systems*, 117, 07 2016.

[23] Christopher Schmidt, Johannes Huegle, and Matthias Uflacker. Order-Independent Constraint-Based Causal Structure Learning for Gaussian Distribution Models Using GPUs. In *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*, SSDBM '18, New York, NY, USA, 2018. Association for Computing Machinery.

[24] Mariusz Kubkowski, Jan Mielniczuk, and Paweł Teisseyre. How to Gain on Power: Novel Conditional Independence Tests Based on Short Expansion of Conditional Mutual Information. *Journal of Machine Learning Research*, 22(1), jan 2021.

[25] Rajat Sen, Ananda Theertha Suresh, Karthikeyan Shanmugam, Alexandres G. Dimakis, and Sanjay Shakkettai. Model-Powered Conditional Independence Test. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 2955–2965, Red Hook, NY, USA, 2017. Curran Associates Inc.

[26] Eric V. Strobl, Kun Zhang, and Shyam Visweswaran. Approximate Kernel-based Conditional Independence Tests for Fast Non-Parametric Causal Discovery. arXiv, 2017.

[27] Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-Based Conditional Independence Test and Application in Causal Discovery. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, page 804–813, Arlington, Virginia, USA, 2011. AUAI Press.

[28] Finn V. Jensen, Uffe Kjærulff, Kristian G. Olesen, and Jan Pedersen. Et forprojekt til et ekspertsystem for drift af spildevandsrensning (an expert system for control of waste water treatment — a pilot project). Technical report, Judex Datasystemer A/S, Aalborg, Denmark, 1989. In Danish.

[29] Diego Colombo and Marloes H. Maathuis. Order Independent Constraint Based Causal Structure Learning. *Journal of Machine Learning Research*, 15(1):3741–3782, January 2014.