UIC CS594 final project report

an evaluation of causal discovery algorithms and exploration on the application on relational data

Chenjie Li

ACM Reference Format:
Chenjie Li. 2024. UIC CS594 final project report: an evaluation of causal discovery algorithms and exploration on the application on relational data. In . ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Causal inference has been a hot topic in recent years. It goes beyond the "association" between A and B that holds statistical significance (e.g., high Pearson correlation coefficient between A and B) but does not actually have any causal effect on the result.

Causal inference has been studied for several decades in different aspects: Pearl's Graphical Causal Model [5], Rubin's Potential Outcome [6] and Causal Model in the database domain [3].

This project is focused on an essential component in applying causal inference with Graphical Causal Model: the causal graph. Many existing causal inference projects have assumed the graphical causal model is given [7, 10]. However, some of the tasks at hand might be a bit too complicated for users to provide a causal graph because of lack of domain knowledge or the nature of the task. In this project I investigated several well-known algorithms on causal discovery on observational data: PC algorithm (Peter-Clark) [8] and FCI (Fast Causal Inference) algorithm [9], which works on single table, and a RCD algorithm [1, 2] which works for relational data. I evaluated the effectiveness of these algorithms using syntactic data.

2 BACKGROUND

Causal discovery has been studied intensively in the past, and in general this line of work can be broken down to a couple of categories: Independence-Based Causal Discovery and Semiparametric Causal Discovery [4]. In this project, I focus mainly on techniques derived from Independence-based causal discovery.

2.1 Markov assumption

One of the main assumption in causal inference is the Markov assumption. It basically describes that if variables are d-seperated in the graph G, then they are independent in the distribution P.

Theorem 1 (Markov). $X \perp\!\!\!\perp_G Y|Z \rightarrow X \perp\!\!\!\perp_P Y|Z$

Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In order to discover the causal graph from the data, however, we need to make "converse" of the Markov assumption, which is known as the faithfulness assumption.

Theorem 2 (Faithfulness). $X \perp\!\!\!\perp_G Y|Z \leftarrow X \perp\!\!\!\perp_P Y|Z$

Note that Theorem 2 is a less robust assumption, as pointed out by [4].

In addition, there is another relevant assumption called *causal sufficiency* which is adapted in PC algorithm and RCD algorithm, but is not assumed to be true in FCI algorithm.

Theorem 3 (Causal Sufficiency). *There are no ubobserved confounders of any of the variables in the graph.*

3 ALGORITHMS

Now based on the theorems introduced in Section 2, I briefly introduce the ideas behind the algorithms evaluated in this project.

3.1 Chain, Fork and Immorality

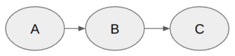

Figure 1: Chain

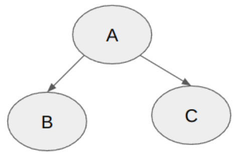

Figure 2: Fork

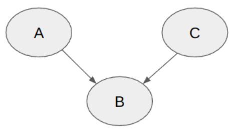

Figure 3: Immorality

There are 3 basic graph "building blocks" in the causal graph, chain, fork and immorality.

Chains and forks share the same set of dependencies. In both structures Figure 1 and Figure 2, A and B are dependent, B and C are dependent. In immorality (Figure 3), A and C are independent.

3.2 Algorithms for single table

These 2 algorithms are designed for discovering causal relations from single relation.

3.2.1 PC algorithm. Originally introduced in [8], PC algorithm starts with a complete undirected graph and then trims down (i.e., removing redundant edges and orient edges as much as possible). it contains 3 steps:

- identify the skeleton
- identify immorality and orient them
- orient qualifying edges that are incident on colliders

In the first step, the skeleton is identified by conducting conditional independence test, i.e., $X \perp\!\!\!\perp Y|Z$ where Z can be as trivial as an empty set. In terms of specific independence tests, any suitable tests can be used such as chi-sqaure tests, fisher-z tests. For each pair of nodes exist in the graph, the algorithm will find the C increasingly, that is starting from an empty set and increase the set size until a certain set pass the conditional independence test.

In the second step, for any results we get from the first step $X - Z - Y$, if X-Z-Y are connected but Z is not in the condition set result, it implies that X->Z<-Y forms an immorality.

In the last step, from step 1 and step 2, we can orient the remaining edges that are connected to the previous immoralities can be determined directly, which can be proven since if it were not the case, they are also immoralities and should've been detected in the previous step.

3.2.2 *FCI algorithm.*

FCI algorithm can be thought of an extension of PC algorithm [8]. Unlike PC, FCI doesnt assume *causal sufficiency* and will test for potential confounders. The steps for FCI can be summarized as follows:

The algorithm can be broken down into the following steps:

- skeleton discovery
- orient edges
- handle latent variables and selection bias

The first 2 steps are essentially the same as PC. The difference mainly lies in the third step, where algorithm tests for potential latent confounders or selection bias. Specifically, for each pair of nodes that has a conditional independence set Z, it will perform additional tests for possible latent confounders. If there are colliders in Z, it will also mark the edge as "suspect" of selection bias. Essentially, my understanding of this is that FCI will give you a very conservative answer, and its up to user or domain expert to determine the real causal graph, FCI is merely a "helper" function to give you some guidance.

3.3 Algorithms for relational data

One of the major motivations for selecting this project as my topic is that I am interested in learning about how to detect causal relationships in the domain of relational data. To highlight the difference, the algorithms for single relations are easier since it only needs to test for finite amount of pairs of nodes. In other words, those algorithms can exhaustively examine all the possible conditional sets for all the possible pairs of nodes. However, in relational data where there are X number of tables and can be joined together in different ways, how to conduct such tests are not very straightforward. In some cases, it is impossible to exhaustively conduct such tests since some tables might be able to join with other tables infinite number of times. (for example, mapping tables with 2 other tables can be joined many times). One of the algorithms proposed to work for relational data is called RCD (relational causal discovery) [2]

3.3.1 *RCD algorithm.*

First, we need to introduce some concepts and terminologies in order to properly introduce the RCD algorithm. Here I borrow Neal's *terminology machine gun* [4] to rapid fire some terminologies needed for the RCD algorithm.

(1) *relational schema* describes the entity, relationship, and attribute classes in a domain, as well as cardinality constraints for the number of entity instances involved in a relationship. An example of relational schema can be found in Figure 8
(2) *relational skeleton* a data set of entity and relationship instances. A relational schema serves as a template for relational skeleton. a relational skeleton based on data and relational schema is show in Figure 9.
(3) *relational path* an alternating sequence of entity and relationship classes that follow connected paths in the schema (subject to cardinality constraints). in Figure 8, possible relational paths include [Actor] (a singleton path specifying an actor), [Movie, Stars-In, Actor] (specifying the actors in a movie), or even [Actor, Stars-In, Movie, Stars-In, Actor] (describing costars).
(4) *Relational variable* consist of a relational path and an attribute, and they describe attributes of classes reached via a relational path
(5) *Relational dependencies* consist of a pair of relational variables with a common first item, called the perspective.
(6) *relational model* is a collection of relational dependencies defined over schema.
(7) *ground graph* a model instantiation generated from a relational model paired with a relational skeleton.
(8) *abstract ground graph* An abstract ground graph for relational model, perspective drawn from edge or entity, and a given hop threshold is a directed graph that captures the dependencies among relational variables holding for any possible ground graph. Figure 11 shows abstract ground graph for the model in Figure 8 from the perspective of ACTOR and MOVIE with hop threshold = 4.

The psudocode for RCD algorithm from [2] is shown in Figure 4. Phase I enumerates potential dependencies with relational paths limited by a hop threshold, it then removes them using conditional independence tests with conditioning sets that are increasingly enumerated and tested. Separating sets are recorded, and dependencies are pruned accordingly. Phase II constructs abstract ground graphs for remaining dependencies and iteratively applies orientation rules while searching for additional separating sets. RCD revisits unshielded triples (X-Y-Z where there is no edge between X and Z) lacking separating sets, addressing cases due to alternative perspectives, hop threshold limits, or identical attributes, which are excluded under the acyclic model assumption. For additional details in those steps please refer to [2].

ALGORITHM 1: RCD($schema, depth, hopThreshold, P$)

```
1  $PDs \leftarrow$ getPotentialDeps($schema, hopThreshold$)
2  $N \leftarrow$ initializeNeighbors($schema, hopThreshold$)
3  $S \leftarrow \{\}$
   // Phase I
4  for $d \leftarrow 0$ to $depth$ do
5      for $X \rightarrow Y \in PDs$ do
6          foreach $condSet \in$ powerset($N[Y] \setminus \{X\}$) do
7              if $|condSet| = d$ then
8                  if $X \perp\!\!\!\perp Y \mid condSet$ in $P$ then
9                      $PDs \leftarrow PDs \setminus \{X \rightarrow Y, Y \rightarrow X\}$
10                     $S[X,Y] \leftarrow condSet$
11                     break
   // Phase II
12 $AGGs \leftarrow$ buildAbstractGroundGraph($PDs$)
13 $AGGs, S \leftarrow$ ColliderDetection($AGGs, S$)
14 $AGGs, S \leftarrow$ BivariateOrientation($AGGs, S$)
15 while $changed$ do
16     $AGGs \leftarrow$ KnownNonColliders($AGGs, S$)
17     $AGGs \leftarrow$ CycleAvoidance($AGGs, S$)
18     $AGGs \leftarrow$ MeekRule3($AGGs, S$)
19 return getCanonicalDependencies($AGGs$)
```

Figure 4: RCD Algorithm

4 EVALUATION

Given the introductions of these 3 algorithms, in this section I evaluate the effectiveness of them using real-world and syntactic datasets.

4.1 Single relation algorithms

for 2 single relation algorithms, I generated a simulated data set in the following way:

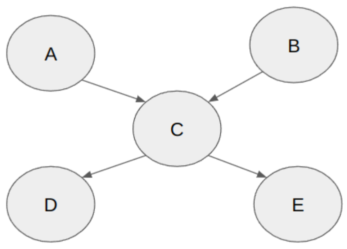

Figure 5: Ground Truth Causal DAG without latent confounder

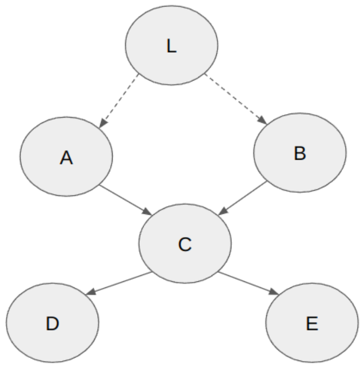

Figure 6: Ground Truth Causal DAG with latent confounder

4.1.1 *causal graph without latent confounders.*

The first syntactic dataset is generated without any latent confounders. The dataset has 5 columns: A,B,C,D,E. The 5 attributes' dependencies are shown in the following:

$$C = 0.5 \times A + 0.3 \times B + \epsilon_0$$
$$D = 0.7 \times C + \epsilon_1$$
$$E = -0.4 \times C + \epsilon_2$$

4.1.2 *causal graph with latent confounders.*

To test the difference between PC and FCI, I also generated a dataset with a latent confounder L, with the flexibility of controlling the coefficients between L and A , L and B:

$$A = Coef_1 \times L + \epsilon_0$$
$$B = Coef_2 \times L + \epsilon_1$$
$$C = A + B$$
$$ssssD = 0.7 \times C + \epsilon_2$$
$$E = -0.4 \times C + \epsilon_3$$

The implementation of PC and FCI are from `Causal-Learn` [11]. Some of the results are shown from Figure 12 to Figure 19. It is clear that when there are no latent confounders present, the results from 2 algorithms are very similar (see Figure 12 vs Figure 13, Figure 14 vs Figure 15) and close to ground truth causal dag. However when there are latent confounders present, both algorithms deviated from the ground truth quite significantly. This experiment shows that when the data is perfect (no latent confounders), PC and FCI can at least return an accurate graph sekeleton, with FCI being a bit more conservative.

4.2 Relational data

for RCD algorithm for relational data, I have also created a syntactic data based on the code implementation from [1] (link here). The schema design is shown in Figure 7. In order to evaluate how good is RCD in recovering ground truth causal dag, I imposed those 2 correlations: PGS.pct<->PGS.points and TGS.tpts<->[TGS,GAME].outcome. The outcome of the RCD algorithm is show in Listing 1. The phase I output is colored in red and the phase II outcome is highlighted in blue. as shown in the results, RCD can successfully detect both of the ground truth relations in phase I. but only succesfully oriented 1 of the 2 ground truth causal relations.

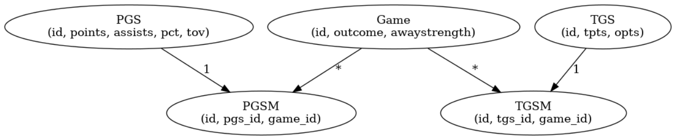

Figure 7: syntactic NBA data: ER diagram

Listing 1: Command-line output of RCD process

```
INFO:RCD:Phase I: identifying undirected
    ↪ dependencies
INFO:RCD:Number of potentialDeps 76
INFO:RCD:Conditioning set size 0
INFO:RCD:Conditioning set size 1
INFO:RCD:Conditioning set size 2
INFO:RCD:Undirected dependencies: [
[PGS].points -> [PGS].pct,
[PGS].pct -> [PGS].points,
[TGS].tpts -> [TGS].opts,
[TGS].opts -> [TGS].tpts,
[Game, TGSM, TGS].tpts -> [Game].outcome,
[TGS, TGSM, Game].outcome -> [TGS].tpts
]
INFO:causality.learning.RCD:Phase II: orienting
    ↪ dependencies
INFO:causality.learning.EdgeOrientation:CD
    ↪ Oriented edge:
[Game, TGSM, TGS].opts->[Game, TGSM, TGS].tpts
INFO:causality.learning.EdgeOrientation:CD
    ↪ Oriented edge:
[Game].outcome->[Game, TGSM, TGS].tpts
INFO:causality.learning.RCD:Oriented dependencies:
{
[TGS, TGSM, Game].outcome -> [TGS].tpts,
[TGS].opts -> [TGS].tpts
}
```

5 CONCLUSION

In this project, i studied 3 different causal discovery algorithms, evaluated their effectiveness in syntactic data. The overall takeaway is that the effectiveness of those algorithms are very sensitive to the quality of the data, that is the data collection process might have a huge impact on the quality of the generated causal graph.

REFERENCES

[1] Ragib Ahsan, David Arbour, and Elena Zheleva. 2023. Learning relational causal models with cycles through relational acyclification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 12164–12171.
[2] Marc Maier, Katerina Marazopoulou, David Arbour, and David Jensen. 2013. A sound and complete algorithm for learning causal models from relational data. *arXiv preprint arXiv:1309.6843* (2013).
[3] Alexandra Meliou, Wolfgang Gatterbauer, Joseph Y Halpern, Christoph Koch, Katherine F Moore, and Dan Suciu. 2010. Causality in databases. *IEEE Data Engineering Bulletin* 33, 3 (2010), 59–67.
[4] Brady Neal. 2020. Introduction to causal inference. *Course Lecture Notes (draft)* (2020).
[5] Judea Pearl. 2009. Causal inference in statistics: An overview. (2009).
[6] Donald B Rubin. 2005. Causal inference using potential outcomes: Design, modeling, decisions. *J. Amer. Statist. Assoc.* 100, 469 (2005), 322–331.
[7] Babak Salimi, Harsh Parikh, Moe Kayali, Lise Getoor, Sudeepa Roy, and Dan Suciu. 2020. Causal relational learning. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*. 241–256.
[8] Peter Spirtes, Clark Glymour, and Richard Scheines. 2001. *Causation, prediction, and search.* MIT press.
[9] Peter L Spirtes, Christopher Meek, and Thomas S Richardson. 2013. Causal inference in the presence of latent variables and selection bias. *arXiv preprint arXiv:1302.4983* (2013).
[10] Brit Youngmann, Michael Cafarella, Amir Gilad, and Sudeepa Roy. 2024. Summarized Causal Explanations For Aggregate Views. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–27.
[11] Yujia Zheng, Biwei Huang, Wei Chen, Joseph Ramsey, Mingming Gong, Ruichu Cai, Shohei Shimizu, Peter Spirtes, and Kun Zhang. 2024. Causal-learn: Causal discovery in python. *Journal of Machine Learning Research* 25, 60 (2024), 1–8.

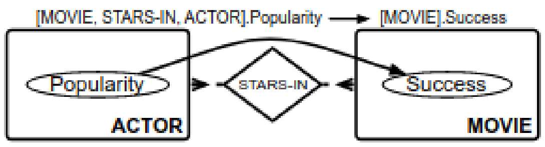

Figure 8: Relational Model

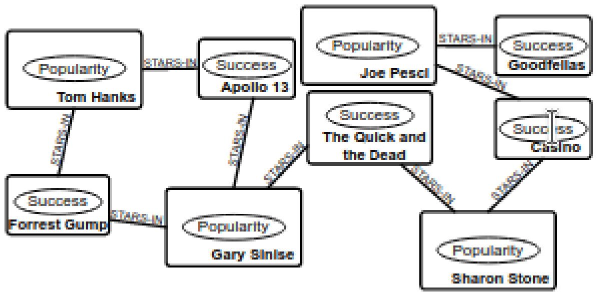

Figure 9: Relational Skeleton

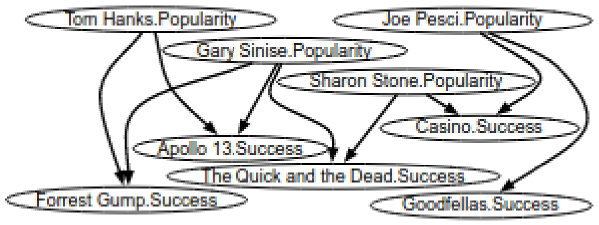

Figure 10: Ground Graph

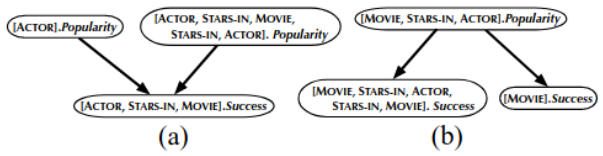
[ACTOR].Popularity
[ACTOR, STARS-IN, MOVIE, STARS-IN, ACTOR]. Popularity
[ACTOR, STARS-IN, MOVIE].Success
(a)
[MOVIE, STARS-IN, ACTOR].Popularity
[MOVIE, STARS-IN, ACTOR, STARS-IN, MOVIE]. Success
[MOVIE].Success
(b)

Figure 11: Abstract ground graph

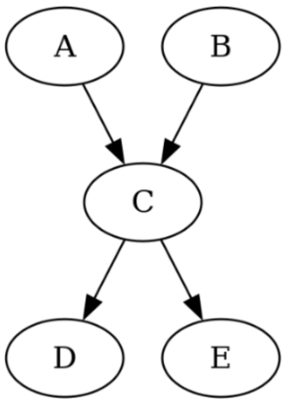

Figure 12: PC: no latent confounder

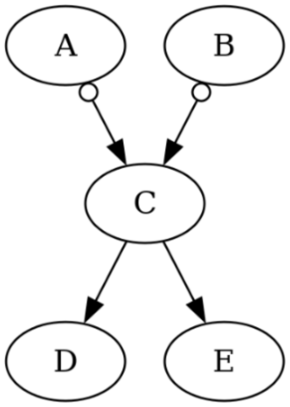

Figure 13: FCI: no latent confounder

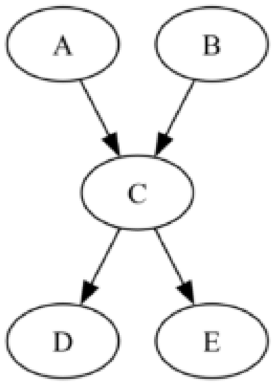

Figure 14: PC: with latent variable: $Coef_1 = 0.9, Coef_2 = 0$

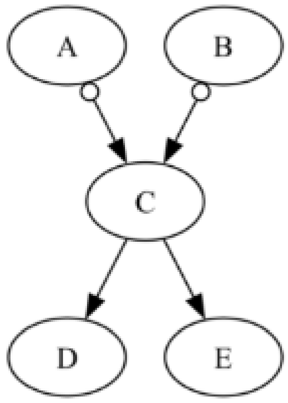

Figure 15: FCI: with latent variable: $Coef_1 = 0.9, Coef_2 = 0$

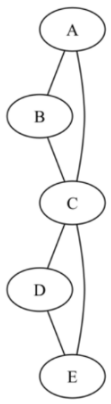

Figure 16: PC: with latent variable: $Coef_1 = 0.9, Coef_2 = 0.1$

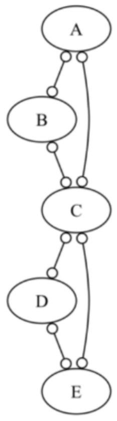

Figure 17: FCI: with latent variable: $Coef_1 = 0.9, Coef_2 = 0.1$

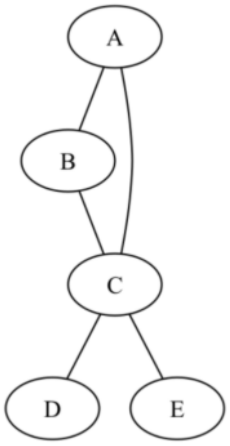

Figure 18: PC: with latent variable: $Coef_1 = 0.3, Coef_2 = 0.3$

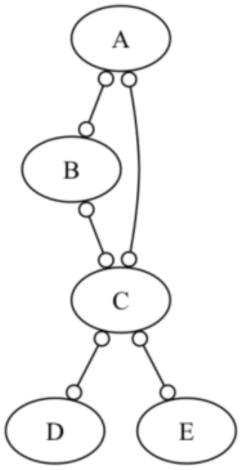

Figure 19: FCI: with latent variable: $Coef_1 = 0.3, Coef_2 = 0.3$