Algorithms for Markov Equivalence in Maximal Ancestral Graphs (MAGs): A Comparative Evaluation

Nikolas Kougioulis*

CS-583 Project (Graph Algorithms), Fall 2023
Instructor: Prof. Ioannis G. Tollis

Jan 2024
Department of Computer Science
University of Crete

Abstract

Maximal Ancestral Graph (MAG) models, introduced by Thomas Richardson and Peter Spirtes, serve as an extension of causal Bayesian Networks [1], capable of capturing causal relationships with latent confounders, as well as selection bias. Just like in causal graphs modeled as DAGs, when causal discovery is performed on observational and interventional data, different MAGs can encode the same conditional independencies and be indistinguishable given a dataset. These MAGs are said to be *Markov Equivalent* since they represent the same amount of statistical information. Over the years, attention has focused on algorithmic methods for deciding on the Markov Equivalence between MAGs. For DAGs, Verma and Pearl have shown that for two DAGs to be Markov Equivalent they need to (i) have the same adjacencies and (ii) have the same unshielded colliders. For MAGs, Markov Equivalence is not as straightforward as in DAGs, as the Spirtes-Richardson criterion (SRC) [5] shows. As such, attention has focused on developing methods for testing Markov equivalence. For MAGs with n vertices and m edges, the first polynomial time algorithm for Markov equivalence testing introduced by Ali, Richardson & Spirtes in 2009 [6], is bounded by $\mathcal{O}(n \cdot m^4)$ (hence $\mathcal{O}(n^9)$ for dense graphs). In 2020, a method using parametrizing sets was proposed by Hu and Evans [7]. These sets can be generated in $\mathcal{O}(n \cdot m^2)$ ($\mathcal{O}(n^5)$ for dense graphs). In 2022, Wienöbst, Bannach, and Liśkiewicz [8] introduced an algorithm with worst running time $\mathcal{O}(n^3)$. We first make a concise introduction and diligent treatment of the theory behind Maximal Ancestral Graphs. Then, the algorithms are presented and finally a comparative evaluation is made among them using methods for generating synthetic MAGs of variable size, density and sparsity which, to our knowledge, are unique. The Julia programming language [11] is chosen for implementing the algorithms and conducting the experiments, in a high-performance computing environment.

Keywords: Causality, Causal Inference, Maximal Ancestral Graphs, Markov Equivalence, Graph Algorithms, Julia.

*kougioulis@csd.uoc.gr

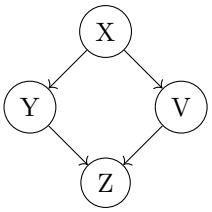


Figure 1: $Ind(Y; V|X)$, $Ind(Z; X|\{Y, V\})$, $Ind(X; Z|\emptyset)$ etc.

1 Introduction

Definition 1. *A Bayesian Network (BN) is a pair (G, Θ) where G is a DAG between random variables $\mathcal{U} = \{X_1, \ldots, X_n\}$ and Θ a set of conditional probabilities $\theta_i = P(X_i|Par(X_i))$ $\forall X_i$. The joint probability on $\mathcal{U}$ decomposes according to the chain rules as*

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|ParX_i)$$

In a DAG, a directed edge denotes dependence, whereas the absence of an edge denotes (conditional) independence. In the causal context, a directed edge $X \to Y$ denotes that X is a **direct** cause of Y (Y is a **direct** effect of X). Denote conditional dependence (independence) of two non-empty sets of variables $\mathbf{X}, \mathbf{Y} \subset \mathcal{V}$ given $\mathbf{Z} \subset \mathcal{V} \setminus \{\mathbf{X}, \mathbf{Y}\}$ (possibly empty) as $Dep(\mathbf{X}; \mathbf{Y}|\mathbf{Z})$ ($Ind(\mathbf{X}; \mathbf{Y}|\mathbf{Z})$).

Definition 2 (Markov Condition)**.** *A node is conditionally independent of its non-descendants on the graph, given its parents.*

Definition 3 (Causal Markov Condition)**.** *A node is conditionally independent of its non-effects on the graph, given its direct causes.*

Definition 4. *A node X of a path p in the DAG is called a **collider** if the previous and next nodes of X in the path are into X.*

Definition 5. *A node X of a path p in the DAG is called an **unshielded collider** if the previous and next nodes Y, Z of X in the path are into X and Y and Z are not adjacent.*

Definition 6. *A path p from X to Y is **blocked** by a set of nodes $\mathbf{Z}$ (possibly empty), if some node on p:*

1. *is a collider and neither it or any of its descendants are in $\mathbf{Z}$, or*
2. *is not a collider and is in $\mathbf{Z}$*

Definition 7. *If a path p is not blocked by a set of nodes $\mathbf{Z}$, it is said to be **active**.*

E.g The path $X \to Y \to Z$ is blocked by $\mathbf{Y}$, $Y \to Z \leftarrow V$ is blocked by $\mathbf{Z} = \emptyset$ etc.

2 D-separation

Definition 8 (d-separation criterion)**.** *Two nodes X and Y are d-separated by $\mathbf{Z}$ iff every path from X to Y is blocked by $\mathbf{Z}$. Otherwise, we say they are **d-connected**.*

When learning causal DAGs from data, **faithfulness** and **causal sufficency** is assumed. For the first one, given a causally sufficient set of variables $\mathcal{U}$ in a population N, *every* conditional independence relation that holds in the density over $\mathcal{U}$ *is entailed by the local directed Markov condition for the causal DAG of N*. Causal sufficiency is the *asumption that there are no unobserved variables*. Unobserved variables are called **latent** or **hidden** variables. The fact that DAGs are unable to adequately capture latent confounders becomes apparent in the example at Figure 2. MAGs are able to drop the causal sufficiency assumption and are closed under marginalizing and conditioning.

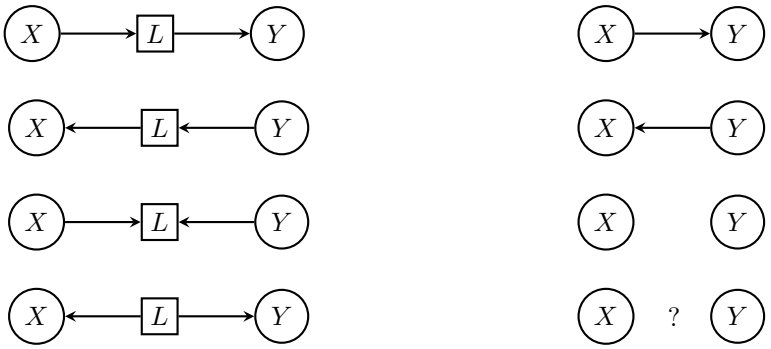


Figure 2: Illustrations for the presence of a hidden confounder L for two variables X, Y. In the first case, where X indirectly causes Y with L acting as a mediator, we observe that $Dep(X, Y)$ so we infer $X \to Y$, and similarly $X \leftarrow Y$ in the second case. In the third case where X and Y share a hidden common effect, we infer $Indep(X, Y)$ and thus discover no edge between X and Y. However, in the case of L being a hidden common cause of X and Y, there exists no DAG that captures the dependencies.

3 Mixed Graphs

Definition 9. *A (directed)* ***mixed graph*** $\mathcal{G}$ *is a graph that may contain two kinds of edges: directed edges (*$\to$*) and bi-directed edges (*$\leftrightarrow$*).*

- Between any two vertices there is at **most one edge**.
- The two ends of an edge we call **marks**.
- There are two kinds of marks: **arrowhead** ($>$) and **tail** ($-$).
- We say an edge is **into** (**out of**) a vertex if the mark of the edge at the vertex is an arrowhead (or tail).

If $X \to Y$ then X is called a **parent** of Y. If $X \leftarrow Y$ then X is called a **child** of Y. If $X \leftrightarrow Y$ then X is called a **sibling** of Y.

$$\begin{cases} \mathbf{pa}_{\mathcal{G}}(v) = \{w : \ w \to v\} \\ \mathbf{sp}_{\mathcal{G}}(v) = \mathrm{sib}_{\mathcal{G}}(v) = \{w : \ w \leftrightarrow v\} \\ \mathbf{an}_{\mathcal{G}}(v) = \{w : \ w \to \ldots \to v \text{ or } w = v\} \\ \mathbf{de}_{\mathcal{G}}(v) = \{w : v \to \ldots \to w \text{ or } w = v\} \\ \mathbf{dist}_{\mathcal{G}}(v) = \{w : w \leftrightarrow \ldots \leftrightarrow v \text{ or } w = v\} \end{cases} \text{ are the } \begin{cases} \textbf{parents} \\ \textbf{spouses or siblings} \\ \textbf{ancestors} \\ \textbf{descendants} \\ \textbf{districts} \end{cases} \text{ of } v \in \mathcal{G}.$$

Definition 10. *A vertex X is said to be an* ***ancestor*** *of a vertex Y, denoted $X \in \boldsymbol{an}(Y)$, if either there exists a directed path $X \to \cdots \to Y$ from X to Y, or $X = Y$. Similarly, Y is called a* ***descendant*** *of X, denoted as $Y \in \boldsymbol{de}(X)$.*

Definition 11. *A graph $\mathcal{G}$ is called an* ***acyclic directed mixed graph (ADMG)*** *if it is acyclic and contains only directed and bidirected edges.*

Definition 12. *A mixed (directed) graph is an* ***ancestral graph*** *if:*

- *there are no directed cycles;*
- *whenever there is an edge $X \leftrightarrow Y$, then there is no directed path from X to Y or from Y to X (no almost directed cycles), that is, $\forall\ v \in \mathcal{V}, sib_{\mathcal{G}}(v) \cap an_{\mathcal{G}}(v) = \emptyset$.*

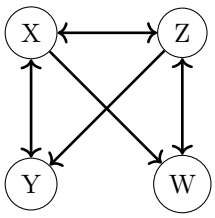


Figure 3: An ancestral graph.

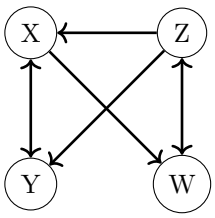


Figure 4: A non-ancestral graph.

Definition 13. *In an ancestral graph, a nonendpoint vertex X on a path is said to be a* ***collider*** *if two arrowheads meet at X (i.e., $\rightarrow X \leftarrow, \leftrightarrow X \leftrightarrow, \leftrightarrow X \leftarrow . \rightarrow X \leftrightarrow$). All other nonendpoint vertices on a path are called* ***noncolliders*** *(i.e., $\rightarrow X \rightarrow, \leftarrow X \leftarrow, \leftarrow X \rightarrow, \leftrightarrow X \rightarrow, \leftarrow X \leftrightarrow$). A path along which every non-endpoint vertex is a collider is called a* ***collider path****.*

The direct generalization of d-separation to MAGs is the m-separation criterion (m-seperation from mixed-separation):

Definition 14 (m-connection)**.** *In an ancestral graph, a path π between vertices X and Y is active or* ***m-connecting*** *relative to a (possibly empty) set of vertices $\mathbf{Z}$, with $X, Y \notin \mathbf{Z}$ if*

- *every non-collider on π is not a member of $\mathbf{Z}$*
- *every collider on π is an ancestor of some member of $\mathbf{Z}$*

Otherwise we say that $\mathbf{Z}$ ***blocks*** *π.*

Example: For the ancestral graph $A \rightarrow B \leftrightarrow C \leftarrow D$: The path $\pi_1 = (A, B, C, D)$ is active relative to $\mathbf{Z} = \{B, C\}$. The path π_1 is not m-connecting relative to $\mathbf{Z} = \emptyset$, $\mathbf{Z} = \{B\}$ or $\mathbf{Z} = \{C\}$.

Definition 15 (m-separation)**.** *X and Y are said to be* ***m-separated by*** *$\mathbf{Z}$ if there are no active paths between X and Y relative to $\mathbf{Z}$, i.e if $\mathbf{Z}$ blocks all paths between X and Y.*

Definition 16 (m-separation)**.** *Two disjoint sets of variables X and Y are m-separated by $\mathbf{Z}$ if every variable in $\mathbf{X}$ is m-separated from every variable in $\mathbf{Y}$ by $\mathbf{Z}$.*

Example: For the ancestral graph $A \rightarrow B \leftarrow C \leftarrow D$, we have that: $\{A\} \perp\!\!\!\perp_m \{D\}$, $\{A\} \perp\!\!\!\perp_m \{D\} \,|\, \{B\}$ and $\{A\} \perp\!\!\!\perp_m \{D\} \,|\, \{C\}$. Since there is no active path relative to $\mathbf{Z} = \emptyset$, $\mathbf{Z} = \{B\}$ and $\mathbf{Z} = \{C\}$ respectively. Moreover, $\{A\} \not\perp\!\!\!\perp_m \{D\} \,|\, \{B, C\}$ because $\pi_1 = (A, B, C, D)$ is active relative to $\mathbf{Z} = \{B, C\}$.

The notion of maximality is that every missing edge corresponds to a conditional independence relation.

Definition 17 (Maximality)**.** *An ancestral graph is* ***maximal*** *if for every pair of nonadjacent vertices (a, b) there exists a set $\mathbf{Z}$ with $a, b \notin \mathbf{Z}$ such that a and b are m-separated conditional on $\mathbf{Z}$ (i.e the pairwise Markov property holds).*

Equivalently:

Definition 18. *An ancestral graph $\mathcal{G}$ is said to be* ***maximal*** *if for every pair of non-adjacent vertices (X, Y) there exists a set of $\mathbf{Z}$ ($X, Y \notin \mathbf{Z}$) such that X and Y are m-separated conditional on $\mathbf{Z}$.*

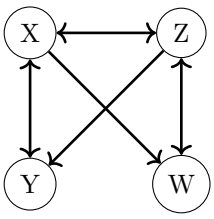


Figure 5: A non-maximal ancestral graph.

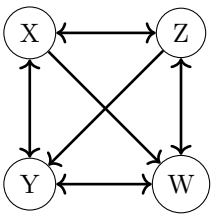


Figure 6: A maximal ancestral graph.

To show that the left AG in Figure 6 is non-maximal, notice that the only pair of non-adjacent vertices is (Y, W). $\mathbf{Z} = \{X\}$, π_1 is *not an active path* since $Z \notin \mathbf{an}(X)$ but $\pi_2 = (Y, Z, W)$ is active since there are no colliders in π_2, Z is a non-collider for π_2 and $Z \notin \mathbf{Z}$. For $\mathbf{Z} = \{X, Z\}$, $\pi_1 = (Y, X, Z, W)$ is a path that m-connects Y and W. For $\mathbf{Z} = \{Z\}$, $\pi_3 = (Y, Z, W)$ is an active path as there are no colliders in π_3, X is a non-collider for π_3 and $X \notin \mathbf{Z}$. For $\mathbf{Z} = \emptyset$, π_1 and π_3 are active paths. The reader should have noticed so far that DAGs are a subset of MAGs (a MAG with no bidirected edges is a MAG).

Theorem 1. *A directed edge $X \to Y$ from some node X into another node Y denotes that Y is not an ancestor of X.*

Proof. Let $\mathcal{M}$ be a MAG and X, Y two nodes on $\mathcal{M}$. Without loss of generality, assume $X \to Y$ is in $\mathcal{M}$ and Y is an ancestor of X. Then there is a directed path from Y to X, but this means that the MAG $\mathcal{G}$ contains a directed cycle, contradiction. Hence if $X \to Y$, Y is not an ancestor of X. □

With a similar proof as before, we can show that a bidirected edge $X \leftrightarrow Y$ between some nodes X and Y denotes that a) X is *not* an ancestor of Y and b) Y is *not* an ancestor of X.

Theorem 2. *A directed edge $X \to Y$ between some nodes X and Y denotes that*

1. *X is an ancestor of Y*
2. *Y is not an ancestor of X*
3. *The above does not rule out **possible latent confounding** between X and Y.*

Theorem 3. *A bidirected edge $X \leftrightarrow Y$ between some nodes X and Y denotes that*

1. *X is not an ancestor of Y*
2. *Y is not an ancestor of X*
3. *X and Y are **confounded**.*

Maximal ancestral graphs (MAGs) are maximal in the sense that **no additional edge may be added to the graph without changing the independence model** induced by the MAG.

Theorem 4. *If $\mathcal{M} = (V, E)$ is a maximal ancestral graph and $\mathcal{M}$ is a subgraph of $\mathcal{G}^* = (V, E^*)$, then $\mathbf{I}_m(\mathcal{M}) = \mathbf{I}_m(\mathcal{M}^*)$ implies that $\mathcal{M} = \mathcal{M}^*$.*

Theorem 5. *If $\mathcal{G}$ is an ancestral graph then there exists a **unique** maximal ancestral graph $\mathcal{M}$ formed by adding $\leftrightarrow$ edges to $\mathcal{G}$ such that $\mathbf{I}_m(\mathcal{M}) = \mathbf{I}_m(\mathcal{G})$*

Maximality is closely related to the definition of (primitive) inducing paths, which is used on the algorithms we will be studying.

Definition 19 (inducing paths)**.** *An **inducing path π relative to a set $\mathbf{L}$** between two vertices X and Y in an ancestral graph $\mathcal{G}$, is a path on which every non-endpoint vertex, not in $\mathbf{L}$ is both a collider on π and an ancestor of at least one of the endpoints X and Y.*

From the definition, we notice that any single-edge path is trivially an inducing path relative to any set of vertices. As a simplification of the terminology, inducing paths relative to the empty set will be referred as (primitive) **inducing paths**. Also it may not be a single path but a collection of paths.

For example, in figure 5 the path (Y, Z, W) is an inducing path relative to $\{Z\}$ but not an inducing path relative to the empty set (because Z is not a collider). The path (Y, X, Z, W) is an inducing path relative to the empty set, because both X and Z are colliders on the path, X is an ancestor of W, and Z is an ancestor of Y.

Definition 20. *A mixed graph is called a maximal ancestral graph (MAG) if:*

- *The graph does not contain any directed or almost directed cycles (ancestral) and*
- *there is no inducing path between any two non-adjacent vertices (maximal)*

4 Markov Equivalence

Just like DAGs (which are a subset of MAGs), several MAGs can encode the same conditional independencies via m-separation (as various DAGs can encode the same conditional independencies via d-separation) and are not distinguishable only by correlational patterns.

Definition 21. *Two MAGs $\mathcal{G}_1, \mathcal{G}\_2$ over the same set of vertices are called **Markov equivalent** if for any three disjoint sets of vertices X, Y, Z, X and Y are m-separated by Z in $\mathcal{G}_1$ iff X and Y are m-separated by Z in $\mathcal{G}_2$.*

Definition 22. *In a MAG, a path consisting of a triple of vertices X, Y, Z is said to be **unshielded** if X and Z are not adjacent.*

Definition 23. *A vertex Y is called an **unshielded collider** if the vertices X, Z are into Y and X and Z are not adjacent.*

Definition 24 (discriminating paths)**.** *In a MAG $\mathcal{G}$, a path $\pi = (x, q_1, \ldots, q_p, b, y)$, $p \geq 1$ is called a **discriminating path for** b if:*

- *x is not adjacent to y, and*
- *every vertex q_i, $1 \leq i \leq p$ is a collider on π and a parent of y.*

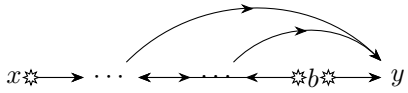


Figure 7: A discriminating path $\pi = (x, q_1, \ldots, q_p, b, y)$, $p \geq 1$ for b.

Notice that a discriminating path is not a single path but potentially a collection of paths (if more than one exist). For Markov Equivalence in MAGs, it is known due to Verma & Pearl [4] that:

Theorem 6 ([4])**.** *Two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent iff*

1. *$\mathcal{G}_1$ and $\mathcal{G}_2$ have the same adjacencies*
2. *$\mathcal{G}_1$ and $\mathcal{G}_2$ have the same unshielded colliders*

For MAGs, although having the same adjacencies and unshielded colliders is necessary, it is not sufficient for Markov equivalence. Consider the MAGs $x \leftrightarrow q \rightarrow y \leftrightarrow b$ with a bidirected edge $q \leftrightarrow b$ and $x \rightarrow q \rightarrow y \leftarrow b$ with a directed edge $q \leftarrow b$. The two MAGs share the same adjacencies and unshielded colliders. But in the first graph, q m-separates x and y but q m-connects x and y in the second graph. The Verma & Pearl criterion on DAGs is extended to MAGs with the Spirtes-Richardson Criterion (SRC) ([5]):

Theorem 7 (Spirtes-Richardson Criterion [5])**.** *Two MAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent iff*

1. *$\mathcal{G}_1$ and $\mathcal{G}_2$ have the same adjacencies*
2. *$\mathcal{G}_1$ and $\mathcal{G}_2$ have the same unshielded colliders and*
3. *if π forms a discriminating path for b in $\mathcal{G}_1$ and $\mathcal{G}_2$, then b is a collider on the path π in $\mathcal{G}_1$ if and only if it is a collider on the path π in $\mathcal{G}_2$.*

Markov Equivalent MAGs form a Markov equivalence class that can be described uniquely by a **partial ancestral graph (PAG)**. A PAG $\mathcal{P}$ has the same adjacencies as any MAG in the Markov equivalence class described by $\mathcal{P}$. We denote all MAGs in the Markov equivalence class described by a PAG $\mathcal{G}$ by $[\mathcal{G}]$. Let *partial mixed graphs* denote the class of graphs containing four types of edges: $\rightarrow$, $\leftarrow$, $\circ\!\!\rightarrow$, $\circ\!\!-\!\!\circ$ and three types of end marks: arrowhead ($>$), tail ($-$) and circle $\circ$.

5 Ali-Spirtes-Richardson (ASR) [6] Algorithm

Discriminating paths, when present in both graphs, lead directly to necessary conditions for Markov equivalence. However, a discriminating path for a given triple may not be present in all graphs within a Markov equivalence class. The authors avoid this by a recursive definition, triples in order, a sub-class of discriminating paths and associated triples (those that are "with order") that are always present, and proving Markov equivalence combined with (i) and (ii) conditions of same adjacencies and unshielded colliders.

Definition 25 (Definition 3.11 in [6]). *Let O_i $(i \geq 0)$ be the set of triples of order i in a MAG $\mathcal{G}$, defined recursively as follows:*

- ***Order 0.*** *A triple $\langle a, b, c\rangle \in O_0$ if a and c are not adjacent.*
- ***Order*** $i+1$***.*** *A triple $\langle a, b, c\rangle \in O_{i+1}$ if*
  1. *for all $j < i+1$, $\langle a, b, c\rangle \notin O_j$, and,*
  2. *there is a discriminating path $\langle x, q_1, \ldots, q_p, b, y\rangle$ for b with either $\langle a, b, c\rangle = \langle q_p, b, y\rangle$ or $\langle a, b, c\rangle = \langle y, b, q_p\rangle$ and the p colliders $\langle x, q_1, q_2\rangle, \ldots, \langle q_{p-1}, q_p, b\rangle \in \bigcup_{j \leq i} O_j$.*

If $\langle a, b, c\rangle \in O_i$ then the triple is said to have order i. If a triple has order i for some i, then we will say that the triple has order. A discriminating path is said to have order i if, excepting $\langle q_p, b, y\rangle$, every collider on the path has order at most $i-1$, and at least one collider has order $i-1$.

The main theorem of the paper, in which the algorithm is based on, is the following:

Theorem 8 (Theorem 3.7 in [6]). *Two MAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent if and only if they have the same adjacencies and the same colliders with order.*

To present the Markov equivalence algorithm, we introduce the following notation:

- $\text{Adj}(\mathcal{G}) = \{(x, y) \mid x \text{ and } y \text{ are adjacent in } \mathcal{G}\}$,
- $\text{Col}(\mathcal{G}) = \{(x, y, z) \mid x \circ\!\!\rightarrow y \leftarrow\!\!\circ z \text{ in } G\}$,
- $\text{OCol}(\mathcal{G}) = \{(x, y, z) \mid (x, y, z) \in \text{Col}(\mathcal{G}) \text{ and } (x, y, z) \text{ has order}\}$,
- $\text{ICol}(\mathcal{G}) = \bigcap_{\overline{\mathcal{G}} \sim \mathcal{G}} (\overline{\mathcal{G}})$

which are, respectively, the *set of adjacencies, colliders, colliders with order in $\mathcal{G}$ and colliders common to all graphs in the Markov equivalence class containing $\mathcal{G}$*. In general, we have that $\text{Col}(\mathcal{G}) \subset \text{ICol}(\mathcal{G}) \subset \text{Col}(\mathcal{G})$.

The algorithm *Reachable* (Algorithm 3) outputs the vertices connected to the input vertex with a directed path, using depth-first search. The algorithm *Triples* seeks to identify a superset of colliders with order and runs as follows: Line 4 locates candidate triples (a, b, c). Lines 5, 6, and 7 search for collider paths and the sets "vertices" (V) and "edges" (E) in $\mathbf{D}$ correspond to edges and colliders in $\mathcal{G}$ respectively. Lines 8 and 9 search for a vertex satisfying the conditions on x.

The algorithm Reachable$(\mathbb{D}, w)$ runs in $\mathcal{O}(n+e)$ (BFS). Regarding Triples$(\mathcal{G})$ (Algorithm 4): Any triple appearing on a minimal m-connecting path π has order at most n^3: π contains at most n vertices; hence, at most n^2 triples; all of the other discriminating paths involved are sections of π; and unshielded triples (of which there is at least one) are of order 0. Thus, it is always sufficient for Markov equivalence to check that two graphs have triples of order less than n. Hence, the main loop in Triples(G) is of complexity $\mathcal{O}(n)$. Line 4 is executed $\mathcal{O}(e^2)$ times (for each k) and since E is of size $\mathcal{O}(e^2)$, lines 5 to 7 are also of complexity $\mathcal{O}(e^2)$. Finally, line 8 is $\mathcal{O}(e^2)$ (since T_{k-1} is of size $\mathcal{O}(e^2)$) and line 9 is $\mathcal{O}(e)$. Thus overall $\mathcal{O}(ne^4)$.

6 Hu-Evans [7] Algorithm

The second attempt at creating an efficient polynomial algorithm for testing Markov equivalence in MAGs was introduced by Hu and Evans [7] in 2020 at the Department of Statistics at the University of Oxford. The authors make use of parametrizing sets, reducing the complexity of the previous approach to $\mathcal{O}(ne^2)$.

Definition 26 (Induced Subgraph)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an ADMG and $W \subset \mathcal{V}$. Then the **induced subgraph** $\mathcal{G}_W$ is the ADMG with vertex set W and edges in $\mathcal{G}$ with endpoints both in W.*

Definition 27 (Barren subsets)**.** *For a set of vertices $W \subset V$: $barren_{\mathcal{G}}(W) = \{w \in W : de_{\mathcal{G}}(w) \cap W = \{w\}\}$.*

Definition 28 (Heads)**.** *A vertex set H is called a head if (i) $barren_{\mathcal{G}}(H) = H$ and (ii) H is contained in a single district in $\mathcal{G}_{an(H)}$ (the subgraph induced by the ancestors of H).*

For an ADMG $\mathcal{G}$, denote the set of all heads in $\mathcal{G}$ as $\mathcal{H}(\mathcal{G})$.

Definition 29 (Tails)**.** *A tail of a head H is defined as $tail(H) = (dis_{an(H)}(H)) \cup pa_{\mathcal{G}}(dis_{an(H)}(H))$.*

Definition 30 (Parametrizing set)**.** *The parametrizing set $\mathcal{S}(\mathcal{G})$ of $\mathcal{G}$ is*

$$\mathcal{S}(\mathcal{G}) = \{H \cup A : h \in \mathcal{H}(\mathcal{G}) \text{ and } \emptyset \subset A \subset tail(H)\}$$

Definition 31. *For $k \geq 2$ define $\mathcal{S}_k(\mathcal{G}) = \{S \in \mathcal{S}(\mathcal{G}) : 2 \leq |S| \leq k\}$ the parametrizing sets of order k. That is, with cardinality 2 or 3.*

Finally, define the following subset of $\mathcal{S}_k(\mathcal{G})$:

Definition 32. $\tilde{\mathcal{S}}_3(\mathcal{G}) = \{S \in \mathcal{S}_3(\mathcal{G})$ *where there are 1 or 2 adjacencies between the vertices in S}.

In a MAG $\mathcal{M}$, a pair of vertices is in $\mathcal{S}(\mathcal{M})$ iff they are adjacent. Recall condition (iii) from the Spirtes-Richardson Criterion (Theorem 7): If $\pi = (x, q_i, \ldots, q_m, b, y)$ with $m \geq 1$ forms a discriminating path for b, then π is a subgraph of the following collection of paths:

$$x \circ\!\!\to q_1 \leftrightarrow \ldots q_i \to y, \ 1 \leq i \leq m$$

$$x \circ\!\!\to q_1 \leftrightarrow \ldots q_m \leftarrow\!\!\circ\, b \circ\!\!\to y$$

The main takeaway of the paper is that conditional independencies in MAGs (induced from m-separation) correspond to the missing sets of the form $\{a, b\} \cup C'$ where $a \perp\!\!\!\perp_m b | C$ and $C' \subset C$. Hence the Markov equivalence algorithm works on the fact that equivalent graphs should have the same parametrizing sets, with equivalence relationships refined to consider only $\mathcal{S}_3$ or $\tilde{\mathcal{S}}_3$. This results in the following central Theorem:

Theorem 9 (3.2, Hu and Evans, 2020 [7])**.** *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two MAGs. Then $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent iff $\tilde{S}(\mathcal{G}_1) = \tilde{S}(\mathcal{G}_2)$.*

The above shows that in order to show Markov equivalence one has to compute the parametrizing sets of the MAGs, which in turn means to compute the corresponding heads and tails. However since their cardinality is not polynomial to the size of the graph, they reduce to sets of cardinality 2 or 3, and even further, with 1 or 2 adjacencies between the vertices, that is, considering only $\mathcal{S}_3$.

Corollary 1 (3.2.1, Hu and Evans, 2020 [7])**.** *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two MAGs. Then $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent iff $\mathcal{S}_3(\mathcal{G}_1) = \mathcal{S}_3(\mathcal{G}_2) \Leftrightarrow \tilde{\mathcal{S}}_3(\mathcal{G}_1) = \tilde{\mathcal{S}}_3(\mathcal{G}_2)$.*

To prove the Theorem and its corollary hold, the following propositions are used:

Proposition 1 (3.3, Hu and Evans, 2020 [7])**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a MAG. For a set $W \subset \mathcal{V}$, $W \notin \mathcal{S}(\mathcal{G})$ if and only if there exist two vertices $a, b \in W$ such that they can be m-separated by a set C such that $a, b \notin C$ with $W \subset C \cup \{a, b\}$.*

MAG (i)	
Heads	Tails
X	Y
Z	∅
Y	∅
W	Z,Y
X,Z	Y
Z,Y	∅

MAG (ii)	
Heads	Tails
X	∅
Z	∅
Y	∅
W	Z
X,Z	∅
X,Y	∅
Z,Y	∅
Y,W	Z
X,Z,Y	∅
X,Y,W	Z

Figure 10: Heads and tails sets of the MAGs (i) and (ii) in Figures 8 (left) and 9 (right) respectively, which are then used to compute the parametrizing sets of each MAG.

Proposition 2 (3.4, Hu and Evans, 2020 [7]). *For a MAG* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

1. *Any* $a, b \in \mathcal{V}$ *are adjacent iff* $\{a, b\} \in \mathcal{S}(\mathcal{G})$
2. *For any unshielded triple* (a, b, c), $\{a, b, c\} \in \mathcal{S}(\mathcal{G})$ *iff* b *is a collider on that triple*
3. *if* π *forms a discriminating path for* b *with two end vertices* $x, y \in \mathcal{V}$ *then* $\{x, b, y\} \in \mathcal{S}(\mathcal{G})$ *iff* b *is a collider on the path* π.

Proof of Theorem 9. The proofs follow from Propositions 3.3 and 3.4 in [7]: For the direct of Theorem 3.2, proposition 3.3 ensures that sets that do not exist in $\mathcal{S}(\mathcal{G})$ are due to m-separation, hence for MAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ there must hold $\mathcal{S}(\mathcal{G}_1) = \mathcal{S}(\mathcal{G}_2)$. For the inverse, proposition 3.4 implies that any violations in Theorem 3.1 result in different parameterizing sets, hence equality must hold. □

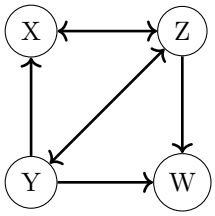


Figure 8: Maximal ancestral graph (i).

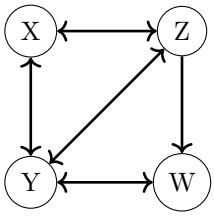


Figure 9: Maximal ancestral graph (ii).

For example, Figures 8 and 9 show two non-Markov equivalent MAGs. The heads and tails sets are computed in Table 6. The parametrizing set for MAG (i) (Figure 8) $\mathcal{S}(\mathcal{G}_1)$ consist of the sets $\{X\}, \{Z\}, \{Y\}, \{W\}, \{X, Z\}, \{X, Y\}, \{Z, Y\}, \{Z, W\}, \{Y, W\}, \{X, Z, Y\}, \{Z, Y, W\}$. Similarly, for the MAG (ii) (Figure 9) we obtain the sets $\{X\}, \{Z\}, \{Y\}, \{W\}, \{X, Z\}, \{X, Y\}, \{Z, Y\}, \{Z, W\}, \{Y, W\}, \{X, Z, Y\}, \{X, Y, W\}, \{Z, Y, W\}, \{X, Y, Z, W\}$. Since they are not equal, the two MAGs are not Markov equivalent (which is also sufficient to show using only the sets $\tilde{\mathcal{S}}_3(\mathcal{G}_1)$ and $\tilde{\mathcal{S}}_3(\mathcal{G}_2)$).

The complexity of Algorithm 6 the following: The loop from lines 2 to 9 runs in $\mathcal{O}(e^2)$ time (in the worst case a vertex has $n-1$ parents, hence $\mathcal{O}(n^2 \cdot n^2) = \mathcal{O}(n^4)$). The loop from line 11 to 24 runs in $\mathcal{O}(e)$ because there are at most e bidirected edges. The set of tails for $\{u, w\}$ runs in $\mathcal{O}(n+e)$ (by running any graph traversal algorithm like BFS). The loop from line 14 to 16 runs in $\mathcal{O}(n-2)$ due to the size of each tail. For line 17 to 24, there are $\mathcal{O}(n-2)$ potential values for z and computing the district runs in $\mathcal{O}(n+e)$ as we have seen before. Hence overall $\mathcal{O}(e^2 + e((n+e) + n + n(n+e))) = \mathcal{O}(e^2 + ne + ne^2 + 2n^2 + en) = \mathcal{O}(ne^2)$.

The authors also show an algorithm for converting and ADMG to a MAG (projection of an ADMG to a MAG) in polynomial time (Algorithm 8) using the definitions of heads and tails. We will be using algorithm in our experiments to convert synthetically generated ADMGs to MAGs, on which we can run the Markov equivalence algorithms. This is based on the following lemma:

Lemma 1 (3.7 in Hu and Evans, 2020 [7]). *Let v, w be two vertices, then*

1. $u \to w \in \mathcal{G}^m$ *iff* $u \in tail_{\mathcal{G}}(w)$
2. $u \leftrightarrow w \in \mathcal{G}^m$ *iff* $\{u, w\} \in \mathcal{H}(\mathcal{G})$

Regarding the time complexity of converting an ADMG to a MAG in Algorithm 8, the outer loop from line 2 to 8 runs worst-case in $\mathcal{O}(n)$, obtaining a district costs $\mathcal{O}(n+e)$ (by performing graph traversal with BFS), hence lines 2 to 8 run in $\mathcal{O}(n(n+e))$. Lines 9 to 14 run in $\mathcal{O}(n^2(n+e))$, similarly to before. Hence overall $\mathcal{O}(n(n+e)+n^2(n+e) = n^2 + ne + n^3 + n^2e) = \mathcal{O}(n^2e)$.

7 Wienobst-Bannach-Liskiwicz [8] Algorithm

In 2022, Wienobst, Bannach and Liskiwicz introduced an algorithm with $\mathcal{O}(n^3)$ complexity based on the Spirtes-Richardson Criterion (SRC), with a Theorem named Constructive SRC. Similarly to Hu et al [7], they introduce the following fact for a discriminating path in a DAG:

Proposition 3 (Fact 4.1 in [8]). *If $\pi = (x, \dots, q, b, y)$ is a discriminating path in a MAG $\mathcal{G}$, then b and y are connected either with $b \leftrightarrow y$ or $b \to y$. In the former case b is a collider on π and a non-collider in the latter.*

This results in the constructive SRC criterion:

Theorem 10 (Constructive SRC, Theorem 4.2 in [8]). *Two MAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent if and only if*

1. *$\mathcal{G}_1$ and $\mathcal{G}_2$ have the same adjacencies,*
2. *$\mathcal{G}_1$ and $\mathcal{G}_2$ have the same unshielded colliders and*
3. *for all bidirected edges $b \leftrightarrow y \in \mathcal{G}_1$ with $Discr_{\mathcal{G}_1}(b, y) \neq \emptyset$ we have $b \to y \notin \mathcal{G}_2$ and vice-versa.*

In terms of parametrizing sets, condition (iii) can be stated as: $\forall(x, b, y)$ defined by a discriminating path $x \circ\!\!\to q_1 \dots \circ\!\!\to q_k \leftrightarrow b \circ\!\!\to y$ the set is parametrizing in both MAGs.

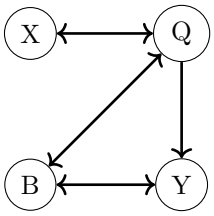


Figure 11: MAG (i)

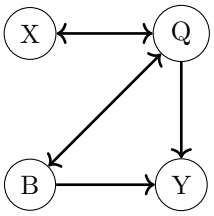


Figure 12: MAG (ii)

For example, the above MAGs are not Markov equivalent as the first one contains a discriminating path from x to y with $b \leftrightarrow y$ while the second the edge $b \to y$ which violates condition (iii).

The advantage of the constructive SRC is that it is not necessary to consider discriminating paths with non-colliders b; one only has to look for discriminating paths with a collider b. It also simplifies the notion of a discriminating path between x and y: *a collider path between non-adjacent endpoints x and y for which every vertex but the one before y is a parent of y.* This also results in the following reformulation for the Markov equivalence criterion on MAGs from Verma & Pearl [4]:

Corollary 2 (Corollary 4.3 in [8]). *Two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent iff*

1. *$\mathcal{G}_1$ and $\mathcal{G}_2$ have the same adjacencies,*

2. *If in $\mathcal{G}_1$ there is an unshielded collider $x \to b \leftarrow y$ then $\mathcal{G}_2$ does not contain an edge $b \to y$ and vice-versa.*

Corollary 3 (Corollary 4.4 in [8]). *Two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent iff*

1. *$\mathcal{G}_1$ and $\mathcal{G}_2$ have the same adjacencies,*
2. *if there is a collider path $(x, \dots, b, y)$ between non-adjacent x, y with every vertex but x, b and y being a parent of y in $\mathcal{G}_1$, then $\mathcal{G}_2$ does not contain the edge $b \to y$ and vice-versa.*

Algorithmically, the conditions (i) and (ii) in the constructive-SRC (Theorem 10) are checked in $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ time respectively. For condition (iii), one needs to check for each $b \leftrightarrow y$ in $\mathcal{G}_1$ for which $b \to y$ is an edge in $\mathcal{G}_2$, whether there is a discriminating path for b and y (and vice-versa). This is done by considering every choice of y consecutively and computing for each the *bidirected connected components of its parents (called the parent districts)*:

Definition 33 (Parent districts). *For a MAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a vertex y, the bidirected connected components of $\mathcal{G}[Pa(y)]$ are called the* ***parent discricts of*** $\mathbf{y}$*, denoted $\mathcal{D}(y)$.*

The algorithmic analysis of algorithm 7 is the following: The conditions (i) and (ii) in the constructive-SRC (Theorem 10) are checked in $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ time respectively in line 1 (In fact checking unshielded colliders is $\Omega(n^3)$ so we cannot achieve complexity better than $\mathcal{O}(n^3)$ already). The loop at line 4 is $\mathcal{O}(n)$ as there are n vertices to run through, while computing the parent districts (bidirected connected components) for a single y can be done in $\mathcal{O}(n+e)$ using Tarjan's algorithm. The loop at line 8 over b is at worst $\mathcal{O}(n)$, and checking whether $b \to y$ exists in the other graph is constant $\mathcal{O}(1)$. Hence in total $\mathcal{O}(n^3)$.

8 Experiments

8.1 Implementation and setup

All three algorithms are implemented in the Julia programming language [11] relying on and enhancing the implementation of [8]. The reasons are plenty: Julia combines both high-level expressiveness and low-level performance capabilities. Also, graph manipulation is highly convenient using the Graphs library with preimplemented methods such as connected components using Tarjan's algorithm. The language's support for Unicode characters (such as set operations on set objects) enhances code readability when translating graph algorithms from pseudocode to Julia. Julia's native multi-threading capabilities also contribute to improved runtime performance.

The experiments are carried out as follows: Acyclic Directed Mixed Graphs (ADMGs) are generated for varying number of nodes and edges using modified random DAG generators to also introduce bidirected edges. More specifically, modified Erdős-Rényi random DAGs [9] and Barabási-Albert models [10] to generate ADMGs instead of DAGs, which are then converted into MAGs. (More random DAG algorithms could have been implemented, such as Watts-Strogatz, Geometric random graphs or complete bipartite graphs but the time complexity for each configuration would raise massively, maybe in a future contribution).

In the Erdős-Rényi DAG model, we initialize n vertices with no edges. We then fix a topological ordering of the vertices, and generate fixed number of edges ($e = dn$ where d is the average degree, a modification on the original $\mathcal{G}_{n,p}$ directed random graph) with the following rule: For each pair of vertices v and u in the graph ($v \neq u$) independently add a directed edge from v to u with probability p. If cycles occur, remove edges to ensure the graph is acyclic. Our modification is to also either add a directed edge from v to u with probability p or a bidirected edge with probability $1-p$ and again check for almost directed cycles - that is, bidirected cycles (recall definition of ancestrality), see Algorithm 1.

In the Barabási-Albert model, a graph of n nodes is grown by attaching new nodes each with m edges preferentially to nodes with high degree. n nodes. We first start with a small set of nodes ($m = 3$) that represents a cluster connected a star graph. Then

Algorithm 1 Erdős-Rényi MAG generator

Input: n, p, d
Output: a MAG G

initialize G as an empty directed graph with n nodes
for $i \leftarrow 1$ to n **do**
    Fix a topological ordering on vertex i
end for
initialize *edges* with all possible directed arcs in G
while number of edges in $G \leq dn$ **do**
    $e \leftarrow$ pop(shuffle(*edges*))
    if $p \leq$ random uniform sample in $[0, 1]$ **then**
        $G^{'} \leftarrow G$ ▷ Make a copy of G
        add directed edge e to $G^{'}$
    else
        $G^{'} \leftarrow G$ ▷ Make a copy of G
        add bidirected edge e to $G^{'}$
    end if
    if $G^{'}$ has no directed cycles or almost directed cycles **then**
        $G \leftarrow G^{'}$ ▷ Update G with the new edge
    end if
end while
$G \leftarrow$ ADMG_to_MAG(G)
return G

we add nodes one at a time till we obtain n nodes. When we add a node, we connect it to a small number of existing nodes with probability proportional to the degree of the existing node. As a result, nodes with higher degree (the ones in the earlier iterations) tend to get even higher degree (rich-get-richer effect). The name "scale-free" network stems that the degree distribution is power-law, $P(k) \sim k^{-\gamma}$ where $2 < \gamma < 3$ ($P(k)$ is the ratio of nodes with degree k to the total number of nodes). The second moment of the distribution (scale) is infinite, hence the name scale-free. In this case, we employ the same rationale as in Erdős-Rényi DAGs, where the connected graph is generated with directed edges. We then randomly choose two vertices and add bidirected edges until their sum to a predefined quantity ($e = dn$) with probability p, that is, we also introduce the a probability parameter p to the model. The parameter $0 < p < 1$ is set from $p \in \{0.2, 0.5, 0.8\}$; a low value of p over $1 - p$ favors the insertion of bidirected edges. In this case, the degree distribution of bidirectional edges does not follow a power-law like in the DAG case of an Erdős-Rényi scale-free graph (which can also be modified), see Algorithm 2.

The generated AMDGs are then converted to MAGs using the AG projection algorithm by Hu and Evans (Algorithm 8). During the conversion, we keep track of the amount of directed and bidirected edges. For each configuration, 100 MAGs are generated to obtain a mean empirical execution time and variance, and 10 MAGs for dense graphs due to very high running time. Since causal graphs are in general sparse, we are testing with degree up to a total of $30n$ number of edges ($d = 30$). The combinations for vertices and edges ($e = dn$)are the following: $d \in \{1, 3, 5, 10, 15, 30\}$. For $d = 1$, we generate graphs from 5 to $2 \cdot 10^3$ vertices, for $d = 3$ from 10 to 10^3, for $d = 5$ from 25 to 10^3, for $d = 10$ from 30 to 500 and for $d = 15$ and 30 from 100 to 500.

To ease the implementation, we split each algorithm in a separate Julia module (`asr.jl`, `hu.jl`, `csrc.jl`), a helper module containing various utility functions for Graph manipulation (`Graph_utils.jl`) and our random graph generation algorithms (`Graph_generators.jl`). We use a plethora of Julia libraries, particularly `Graphs` (for graph handling and manipulation, as well as pre-implemented algorithms such as connected components), `DataStructures` (for stack, queue and set objects when implementing the algorithms), `Random` and `Statistics` (for probability distributions, generators and sampling), `ProgressBars` (for dynamically tracking the progress of the experiments on screen) and `Dates` (for handling execution time and logging). We represent Maximal Ancestral Graphs as a *pair of two graphs with the same vertices*, one with *directed edges* and one with *undirected edges* (that represent the bidirected

Algorithm 2 Barabási-Albert random MAG generator

Input: n, m, m_0, d, p
Output: a MAG G

for $i \leftarrow 1$ to n **do**
    Fix a topological ordering on vertex i
end for
$G = (V_G, E_G)$ a random star graph with m_0 nodes
$G \leftarrow \text{ToDag}(G)$ ▷ Convert the graph to a DAG
$rp \leftarrow [\,]$ ▷ List of potential attachment points
for v in V_G **do**
    append v to rp as many times as its degree in G
end for
while $|E_G| \leq dn$ **do**
    $targets \leftarrow \text{RandomSubset}(rp, n)$
    $t_1 \leftarrow \text{RandomNode}(V_G)$
    for t_2 in $targets$ **do**
        if $p \leq$ random uniform sample in $[0, 1]$ **then**
            $E_G \leftarrow E_G \cup (t_1, t_2)$
        else
            if graph $(V_G, E_G \cup (t_1, t_2))$ is almost acyclic **then**
                $E_G \leftarrow E_G \cup (t_1 \leftrightarrow t_2)$
            end if
        end if
    end for
    append $targets$ to rp
    append t_1 to rp m times
end while
$G \leftarrow \text{ADMG\_to\_MAG}(G)$
return G

edges). In order to run the experiments, one simply executes `julia experiments.jl`, with the results available in a dedicated logs folder. Due to computational complexity, the experiments are run locally on an AMD Ryzen Threadripper 16-core workstation for several weeks, with 128GB of memory. Visualization on the terminal window regarding the progress of the experiments is done using the `tqdm` library. For an illustration of the running script see Figure **??** in the Appendix. Complete code is available[1]

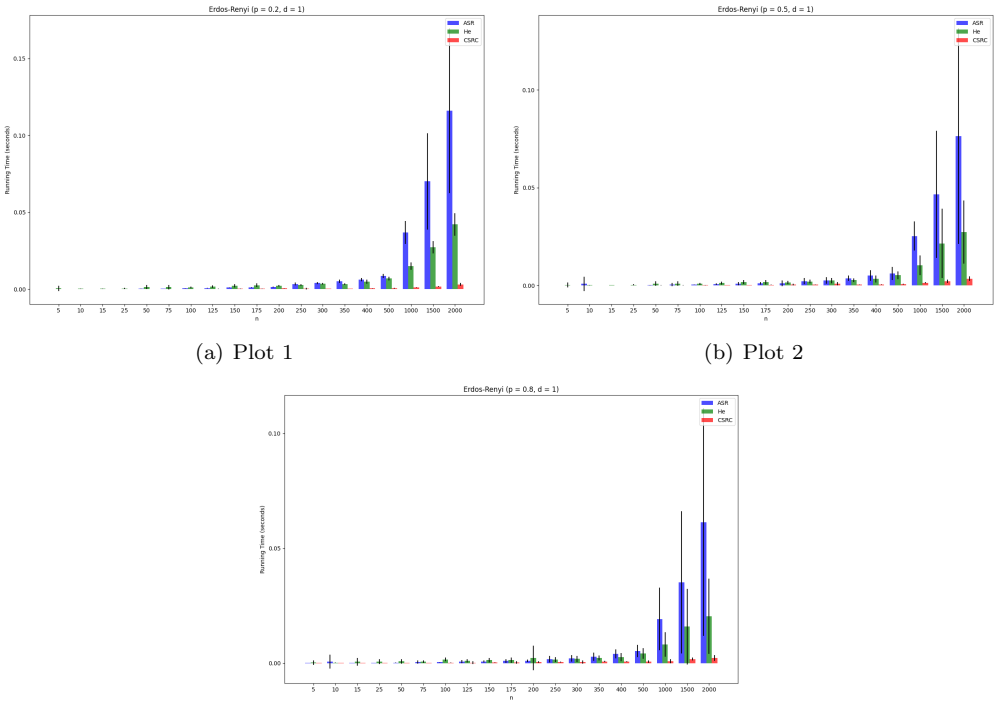
8.2 Experimental Results

8.2.1 Modified Erdős-Rényi model

In the following pages, we report the average running times and standard deviations of each configuration for MAGs based on the Erdős-Rényi model. We illustrate the plots for sparse graphs ($d = 1$) and dense graphs ($d = 30$). All plots for $d = 1, 3, 5, 15, 30$ are available in the appendix.

In the Erdős-Rényi model, big differences are observed in running time even for very sparse graphs between the three algorithms, and even more prominently for dense graphs. We notice that for larger values of p, the average running times are lower for all algorithms as well as their standard deviation (a value of p close to 1 will highly favor directed over bidirected edges). That is, the more latent confounders are assumed to exist between variables, the higher the running complexity.

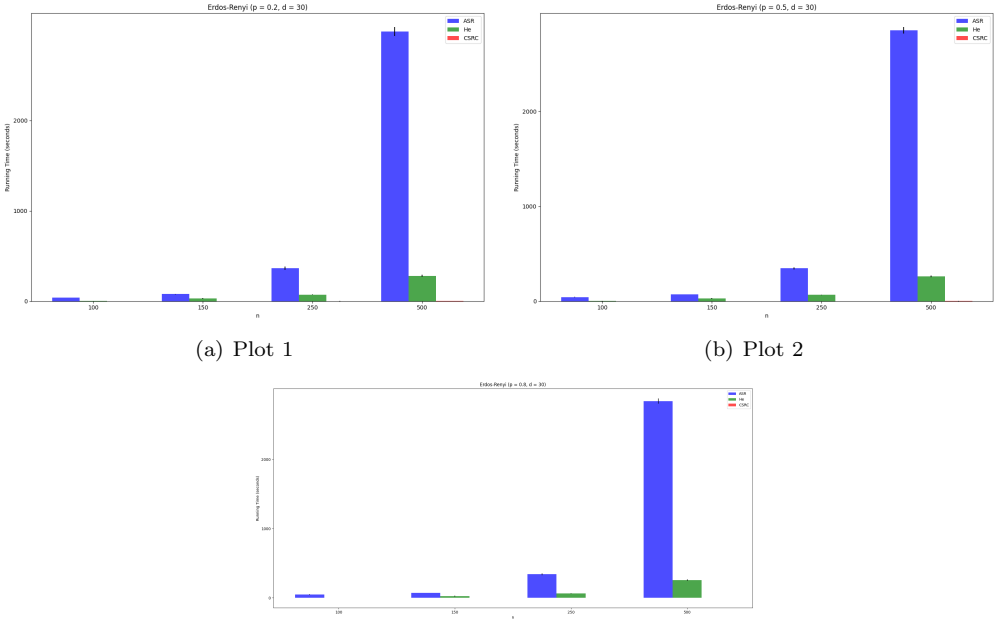
[1] https://github.com/kougioulis/CS-583.



(a) Plot 1

(b) Plot 2

Figure 13: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for sparse Erdős-Rényi MAGs ($d = 1$) and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).

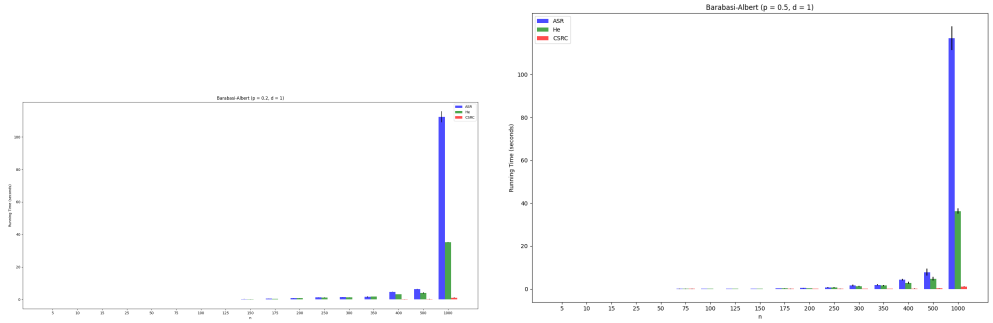


(a) Plot 1

(b) Plot 2

Figure 14: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for dense Erdős-Rényi MAGs ($d = 30$) and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).

8.2.2 Modified Barabási-Albert model



(a) Plot 1

(b) Plot 2

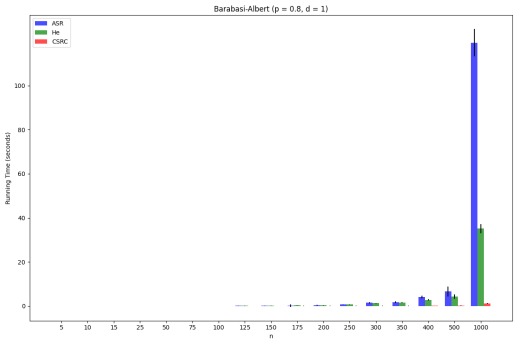
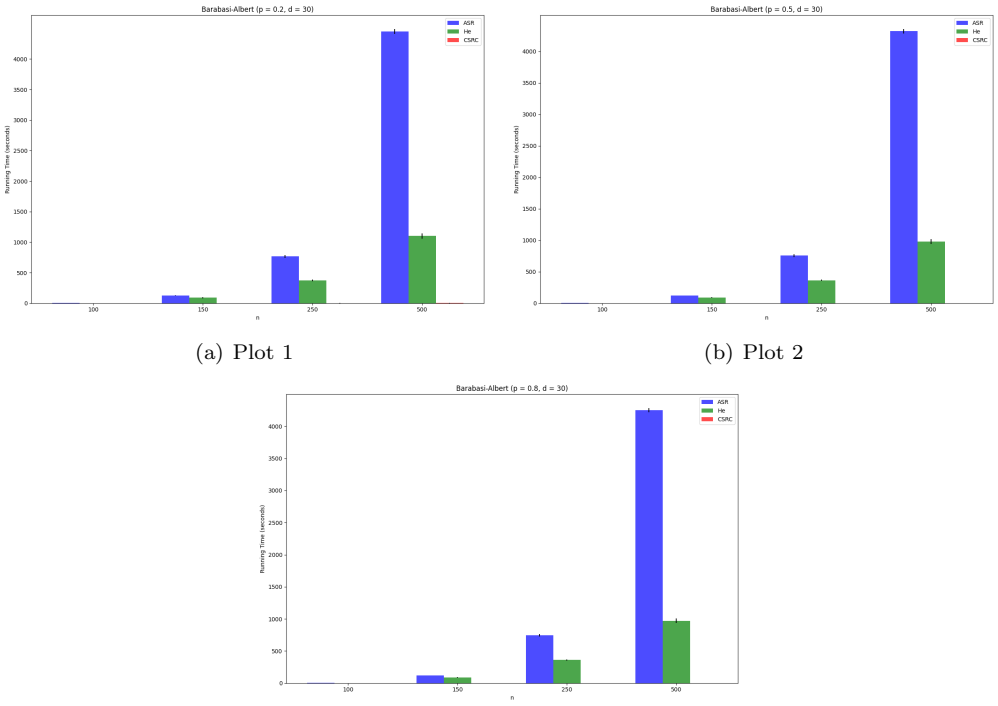


Figure 15: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for sparse Barabasi-Albert MAGs ($d = 1$) and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).

Regarding MAGs based on scale-free graphs using the Barabási-Albert model, very similar characteristics are observed between the algorithms like in the Erdős-Rényi model. The running times are much higher, especially for the Ali-Spirtes-Richardson and Hu-Evans algorithms, due to the rich-get richer effect concentrating a high amount of edges in clusters.



(a) Plot 1

(b) Plot 2

Figure 16: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for dense Barabasi-Albert MAGs ($d = 30$) and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).

8.3 Conclusion

We have presented maximal ancestral graphs (MAGs), a class of graphical models viewed as an extension to causal DAGs that drop the causal sufficiency condition and are able to model latent confounders and selection bias. When learning MAGs from observational data, like DAGs, multiple MAGs may encode the same statistical (in)dependencies. Such MAGs are called Markov equivalent. Is it of importance in the field of causal discovery to test whether two MAGs are Markov equivalent. Although checking Markov equivalence in DAGs is straightforward, this is not transferred directly to MAGs. Over the years three algorithms have been developed using the Spirtes-Richardson criterion to check for Markov equivalence. The Ali-Spirtes-Richardson algorithm works by constructing triples in order in a recursive manner, level-by-level. The Hu-Evans algorithm constructs the so-called parametrizing sets of a MAG and checks for Markov-equivalence if both MAGs have the same parametrizing sets. The Wienbost-Bannach algorithm, called Constructive-SRC, is a constructive method based on the Spirtes-Richardson criterion using the parent districts of the vertices. We modify known random DAG generators to generate ADMGs and then convert them to MAGs, called modified Erdős-Rényi and modified Barabasi-Albert MAG generators and set up experiments in the Julia programming language. For sparse graphs, which are prominent in causal discovery, the Ali-Spirtes-Richardson and Hu-Evans algorithms have similar performance. On denser graphs, where the running time of the first two algorithms is much higher than in smaller graphs, the performance advantage of the constructive-SRC algorithm is highly prominent. Very similar but amplified results are shown in Barabasi-Albert graphs, due to their structure. More results can be obtained for other types of graphs, such as geometric random and bipartite random graphs.

As a final note, causal discovery libraries such as `causaldag` implement Markov equivalence on MAGs using the Ali-Spirtes-Richardson algorithm[2]. We are looking forward to implement the faster algorithms of Hu-Evans and Wienbost-Bannach-Liskiwicz

[2] https://causaldag.readthedocs.io/en/latest/classes/generated/causaldag.classes.ancestral_graph.AncestralGraph.markov_equivalent.html.

to the package.

References

[1] Pearl, J. (2009). *Causality.* Cambridge University Press.

[2] Spirtes, P., Glymour, C. N., & Scheines, R. (2000). *Causation, Prediction, and Search.* MIT press.

[3] Richardson, T., & Spirtes, P. (2002). *Ancestral graph Markov models.* The Annals of Statistics, 30(4), 962.

[4] Verma, T., & Pearl, J. (1990). *Equivalence and Synthesis of Causal Models.* In Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence (pp. 255-270).

[5] Spirtes, P., & Richardson, T. (1996). *A polynomial time algorithm for determining DAG equivalence in the presence of latent variables and selection bias.* In Proceedings of the 6th International Workshop on Artificial Intelligence and Statistics (pp. 489-500).

[6] R Ayesha Ali, Richardson T., & Spirtes, P. (2009). *Markov equivalence for Ancestral Graphs.* The Annals of Statistics, 37(5B):2808–2837.

[7] Hu, Z., & Evans, R. (2020). *Faster algorithms for Markov Equivalence.* In Conference on Uncertainty in Artificial Intelligence (pp. 739-748). PMLR.

[8] Wienöbst, M., Bannach, M., & Liśkiewicz, M. (2022). *A new constructive criterion for Markov equivalence of MAGs.* In Uncertainty in Artificial Intelligence (pp. 2107-2116). PMLR.

[9] Erdős, P., & Rényi, A. (1959). *On random graphs I.* Publ. Math. debrecen, 6(290-297), 18.

[10] Barabási, A. L., & Albert, R. (1999). *Emergence of scaling in random networks.* Science, 286(5439), 509-512.

[11] Bezanson, J., Karpinski, S., Shah, V. B., & Edelman, A. (2012). *Julia: A fast dynamic language for technical computing.* arXiv preprint arXiv:1209.5145.

A Appendix

A.1 Ali-Spirtes-Richardson Algorithm

Algorithm 3 The algorithm Reachable($\mathbb{D}, \mathbf{w}$)

Input: A directed graph $\mathbb{D}(\mathbb{V}, \mathbb{E})$; an element $\mathbf{w} \in \mathbb{V}$
Output: A set $\mathbb{S}$ of elements connected to $\mathbf{w}$ in $\mathbb{D}$

$\mathbb{S}_0 = \emptyset$; $\mathbb{S}_1 = \{\mathbf{w}\}$; $p = 1$
repeat
  $\mathbb{S}_{p+1} = \mathbb{S}_p \cup \{\mathbf{w}_2 \mid \mathbf{w}_1 \in \mathbb{S}_p \setminus \mathbb{S}_{p-1}$ and $(\mathbf{w}_1, \mathbf{w}_2) \in \mathbb{E}\}$
  $p = p + 1$
until $\mathbb{S}_p = \mathbb{S}_{p-1}$
return $\mathbb{S} = \mathbb{S}_p$

Algorithm 4 Triples($\mathcal{G}$)

Input: A MAG $\mathcal{G}$
Output: set of triples T such that $\mathrm{OCol}(\mathcal{G}) \subseteq T \subseteq \mathrm{ICol}(\mathcal{G})$

$T_0 = \{(a, b, c) \mid (a, b, c) \in \mathrm{Col}(\mathcal{G}), (a, c) \notin \mathrm{Adj}(\mathcal{G})\}$
$k = 0$
repeat $k = k + 1$
  $T_k = T_{k-1}$
  for each $(a, b, c) \in \mathrm{Col}(\mathcal{G}) \setminus T_{k-1}$ with $a \in \mathrm{sp}_{\mathcal{G}}(b) \cap \mathrm{pa}_{\mathcal{G}}(c)$ **do**
    $V = \{(t, u) \mid t, u \in \mathrm{pa}(c), t \leftrightarrow u$ in $G\} \cup \{(b, a)\}$
    $E = \{((t, u), (u, v) \mid (t, u, v)) \in T_{k-1}, (t, u), (u, v) \in V\}$
    $S = \mathrm{Reachable}((V, E), (b, a))$
    $X = \{x \mid \exists\, y, z, (z, y, x) \in T_{k-1}, (z, y) \in S\}$
    if $X \setminus \{v \mid (v, c) \in \mathrm{Adj}(G)\} \neq \emptyset$ **then** $T_k = T_k \cup \{(a, b, c), (c, b, a)\}$
    end if
  end for
until $T_k = T_{k-1}$
return $T = T_k$

Algorithm 5 Equivalent

Input: Two maximal ancestral graphs $\mathcal{G}_1$ and $\mathcal{G}_2$
Output: A Boolean variable indicating whether $\mathrm{I}_m(\mathcal{G}_1) = \mathrm{I}_m(\mathcal{G}_2)$ (That is, being the same I-map, which is entailing the same m-separation properties, which is being Markov equivalent [1]).

if $\mathrm{Adj}(\mathcal{G}_1) \neq \mathrm{Adj}(\mathcal{G}_2)$ **then return** False
end if
if $\mathrm{Triples}(\mathcal{G}_1) \setminus \mathrm{Col}(\mathcal{G}_2) \neq \emptyset$ **then return** False
end if
if $\mathrm{Triples}(\mathcal{G}_2) \setminus \mathrm{Col}(\mathcal{G}_1) \neq \emptyset$ **then return** False
end if
return True

A.2 Hu-Evans Algorithm

Algorithm 6 Algorithm to obtain $\tilde{\mathcal{S}}_3(\mathcal{G})$ for a MAG $\mathcal{G}$.

Input: A MAG $\mathcal{G}(\mathcal{V}, \mathcal{E})$
Output: Set $\tilde{S}_3(G)$

$S = \emptyset$
for each $v \in \mathcal{V}$ **do**
    obtain $an_{\mathcal{G}}(v) = \{v\} \cup an_{\mathcal{G}}(pa_G(v))$
    for each $w \in pa_{\mathcal{G}}(v)$ **do**
        $S = S \cup \{v, w\}$
    end for
    for each $z, w \in pa_{\mathcal{G}}(v)$ with $z \neq w$ and z not adjacent to w **do**
        $S = S \cup \{v, w, z\}$
    end for
end for
for each $v \leftrightarrow w$ **do**
    $S = S \cup \{v, w\}$
    $tail(\{v, w\}) = dis_an(\{v, w\})(v) \cup pa_G(dis_an(\{v, w\})(v)) \setminus \{v, w\}$
    for each $z \in tail(\{v, w\})$ with z not adjacent to both v and w **do**
        $S = S \cup \{v, w, z\}$
    end for
    for each $z \in sib_{\mathcal{G}}(an_{\mathcal{G}}(\{v, w\})) \cap dis_{\mathcal{G}}(v) \setminus (an_{\mathcal{G}}(\{v, w\}) \cup de_{\mathcal{G}}(\{v, w\}))$ **do**
        if z not adjacent to both v and w **then**
            obtain $dis_{an(\{v,w,z\})}(v)$
            if $z \in dis_{an(\{v,w,z\})}(v)$ **then**
                $S = S \cup \{v, w, z\}$
            end if
        end if
    end for
end for
return $\tilde{S}_3(\mathcal{G}) \equiv S$

A.3 Wienobst-Bannach-Liskiwicz Algorithm

Algorithm 7 Algorithm for testing the constructive-SRC condition [8].

Input: Two MAGs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1), \mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$
Output: True if $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent, False otherwise

if conditions (i) or (ii) of the constructive-SRC are violated **then return** False
end if
for each $\mathcal{G}_k = (\mathcal{V}_k, \mathcal{E}_k),\ k = 1, 2$ **do**
    for each $y \in \mathcal{V}_k$ **do**
        for each $D \in \mathcal{D}(y)$ **do**
            Compute $\mathrm{Pa}_{\mathcal{G}_k}(D)$ and $\mathrm{Si}_{\mathcal{G}_k}(D)$
            if $\mathrm{Pa}_{\mathcal{G}_k}(D) \cup \mathrm{Si}_{\mathcal{G}_k}(D) \setminus \mathrm{Ne}_{\mathcal{G}_k}(y) \neq \emptyset$ **then**
                for each $b \in \mathrm{Si}_{\mathcal{G}_k}(D) \cap \mathrm{Si}_{\mathcal{G}_k}(y)$ **do**
                    Let $\mathcal{G}_{k'}$ denote the other MAG
                    $k' = 3 - k$
                    if $b \to y \in \mathcal{G}_{k'}$ **then return** False
                    end if
                end for
            end if
        end for
    end for
end for
    return True

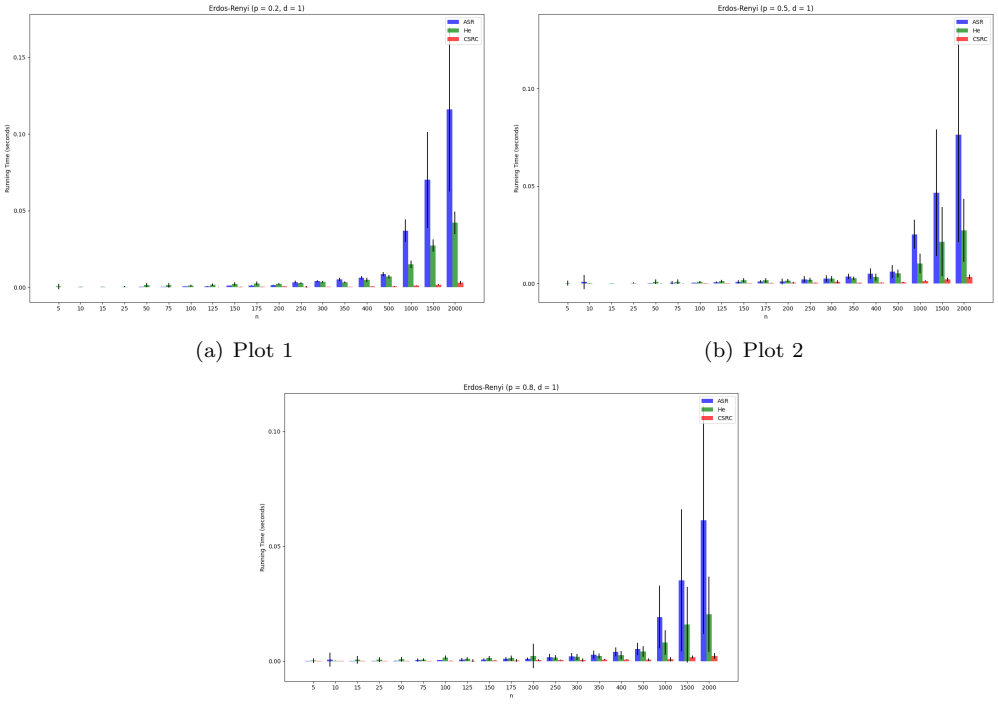
A.4 ADMG to MAG algorithm

Algorithm 8 ADMG $\mathcal{G}$ to MAG $\mathcal{G}_m$

Input: An ADMG $\mathcal{G}(\mathcal{V}, \mathcal{E})$
Output: A Markov equivalent MAG $\mathcal{G}_m(\mathcal{V}, \mathcal{E}_m)$

Start with $\mathcal{G}_m$ that has the same vertices as $\mathcal{G}$ but no adjacencies;
for each $v \in \mathcal{V}$ **do**
    obtain $an_{\mathcal{G}}(v) = \{v\} \cup an_{\mathcal{G}}(pa_{\mathcal{G}}(v))$
    $tail(v) = dis_{an}(v) \cup pa_{\mathcal{G}}(dis_{an(v)}) \setminus \{v\}$
    for each $w \in tail(v)$ **do**
        add $w \to v \in E_m$
    end for
end for
for each $v, w \in \mathcal{V}$ with no ancestral relation and in the same district **do**
    obtain $dis_{an}(\{v, w\})$
    if $w \in dis_{an}(\{v, w\})$ **then**
        add $v \leftrightarrow w \in \mathcal{E}_m$
    end if
end for
return $\mathcal{G}_m$

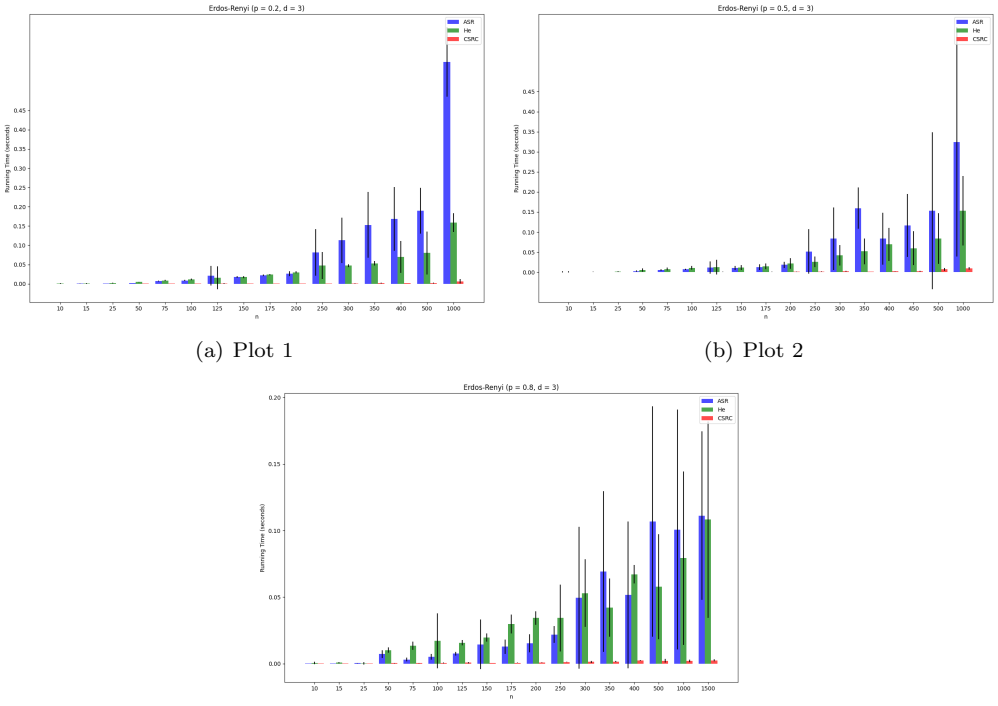
A.5 Plots for the modified Erdős-Rényi model



(a) Plot 1

(b) Plot 2

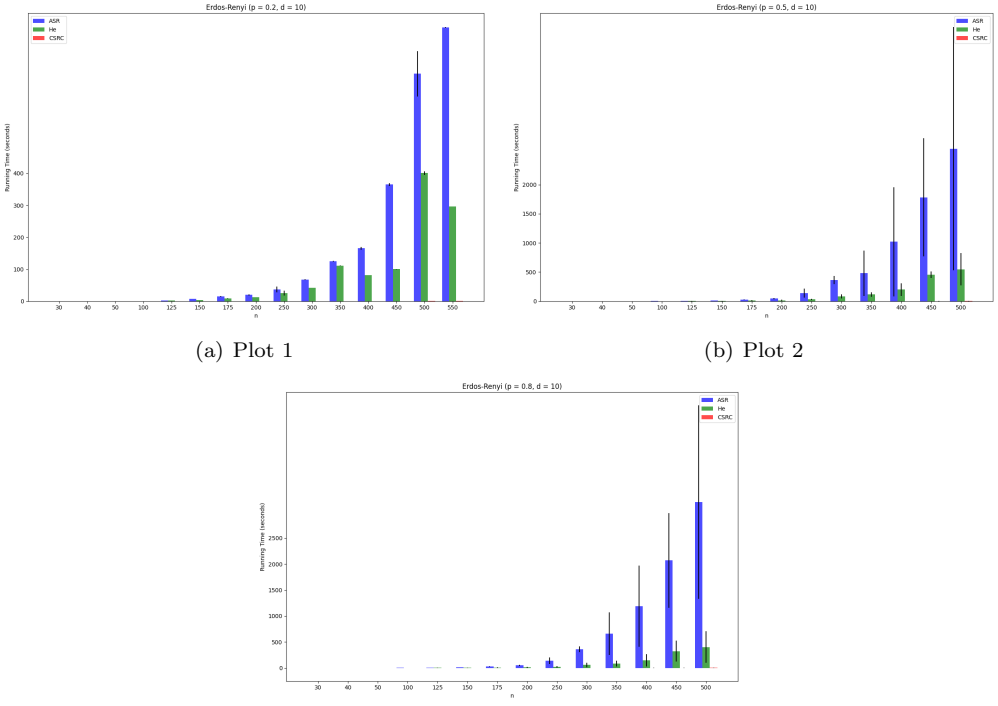
Figure 17: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for Erdős-Rényi MAGs with $d = 1$ and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).



(a) Plot 1

(b) Plot 2

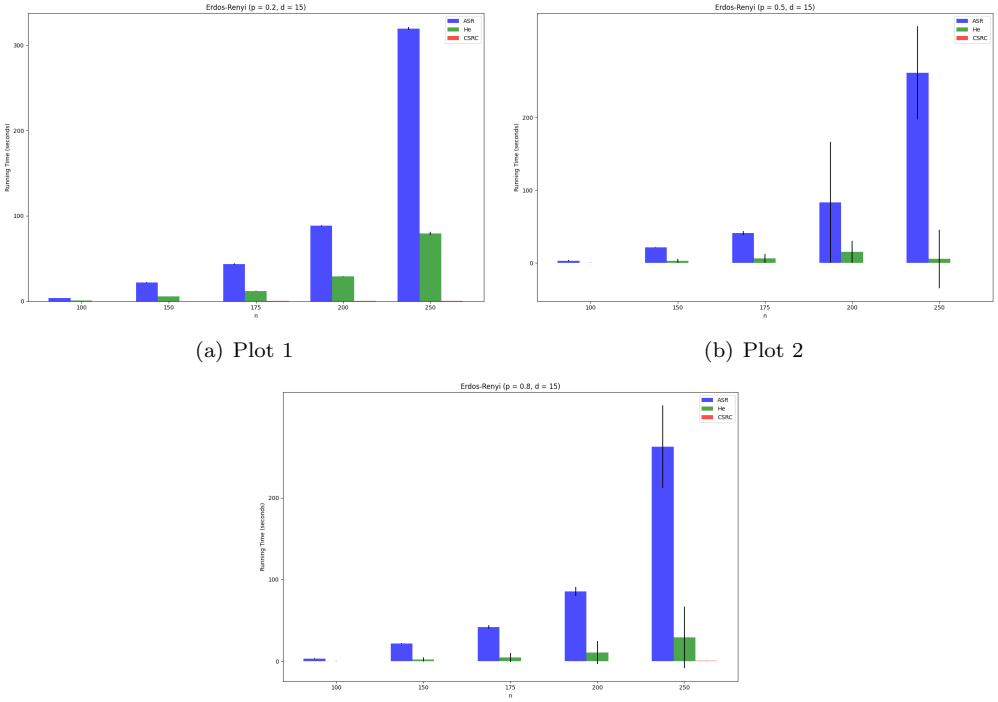
Figure 18: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for Erdős-Rényi MAGs with $d = 3$ and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).



(a) Plot 1

(b) Plot 2

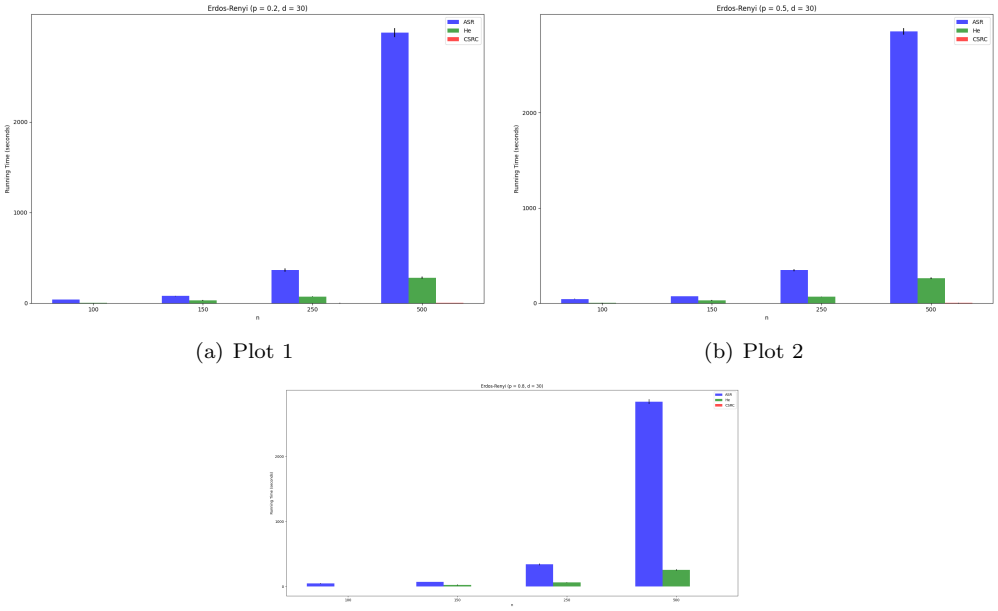
Figure 19: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for Erdős-Rényi MAGs with $d = 10$ and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).



(a) Plot 1

(b) Plot 2

Figure 20: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for Erdős-Rényi MAGs with $d = 15$ and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).

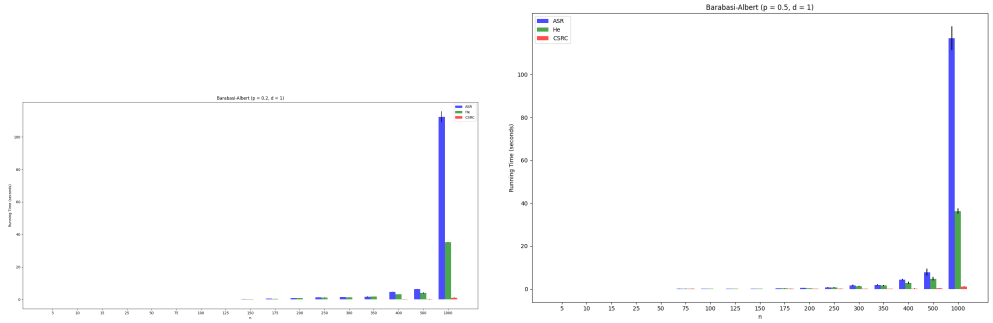


(a) Plot 1

(b) Plot 2

Figure 21: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for Erdős-Rényi MAGs with $d = 30$ and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).

A.6 Plots for the Barabási-Albert model



(a) Plot 1

(b) Plot 2

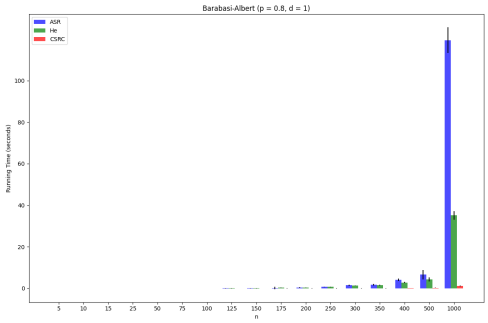
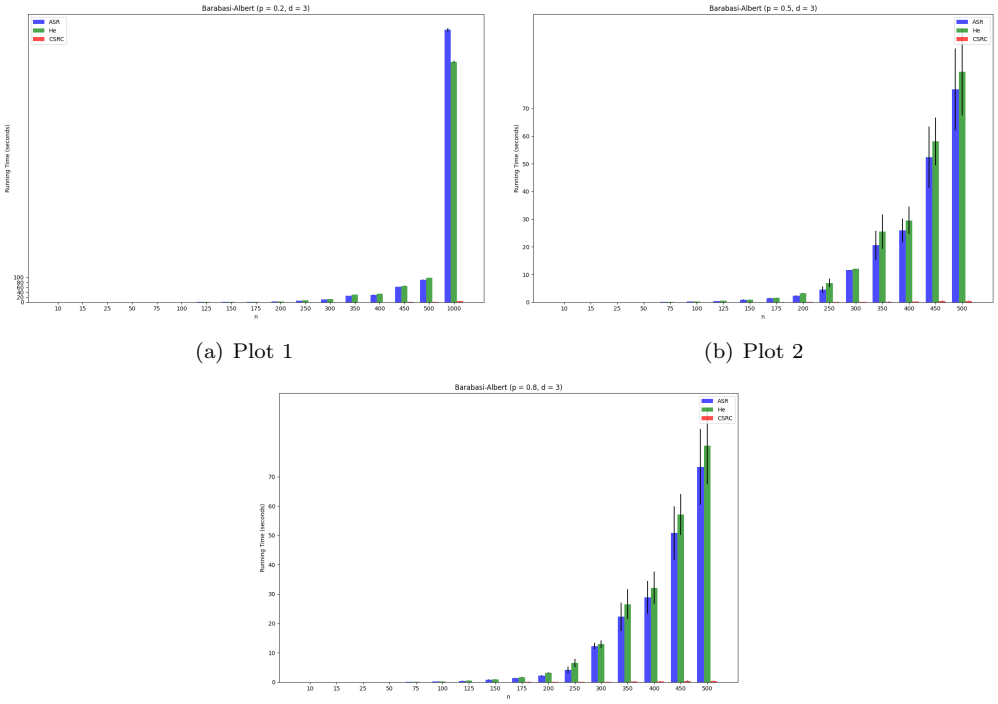


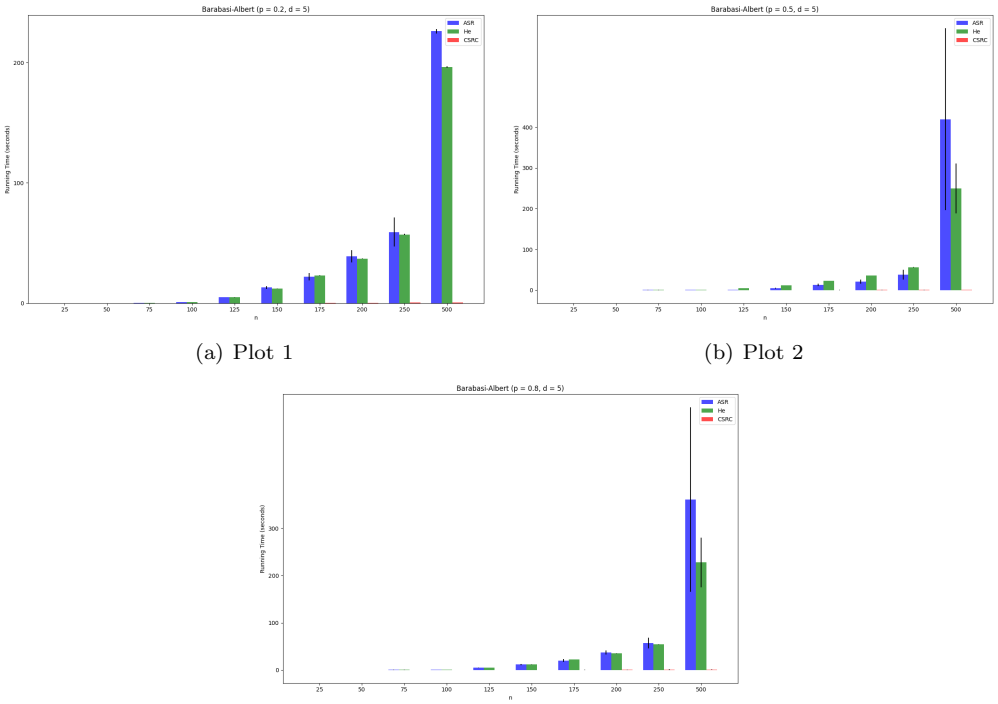
Figure 22: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for Barabasi-Albert MAGs with $d = 1$ and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).



(a) Plot 1

(b) Plot 2

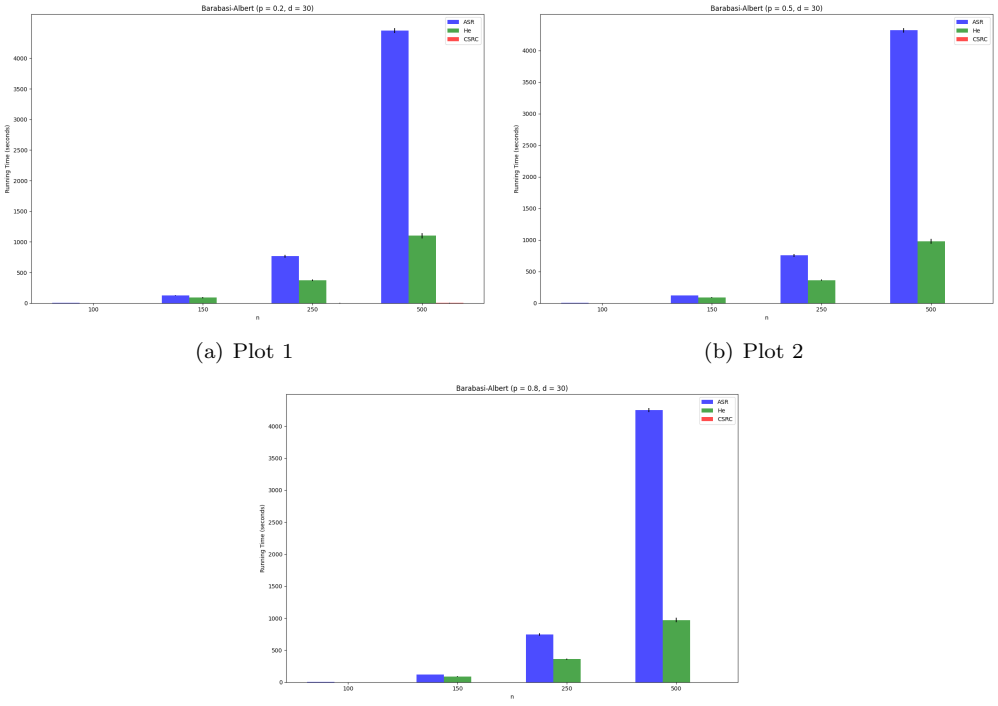
Figure 23: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for Barabasi-Albert MAGs with $d = 3$ and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).



(a) Plot 1

(b) Plot 2

Figure 24: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for Barabasi-Albert MAGs with $d = 5$ and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).



(a) Plot 1

(b) Plot 2

Figure 25: Running time plots for each algorithm (Ali-Spirtes-Richardson (ASR) in blue, Hu-Evans (He) in green and Constructive-SRC (CSRC) in red) for Barabasi-Albert MAGs with $d = 30$ and $p = 0.2$ (Figure (a)), $p = 0.5$ (Figure (b)), $p = 0.8$ (Figure (c)).