Contents of Appendix

A Meek Rules

Meek rules refer to a collection of four edge orientation rules that are proven to be sound and complete when applied to a set of arcs that possesses a consistent extension to a directed acyclic graph (DAG) [Mee95]. With the presence of edge orientation information, it is possible to iteratively apply Meek rules until reaching a fixed point, thereby maximizing the number of oriented arcs.

Definition 15 (Consistent extension)**.** For a given graph G, a set of arcs is considered to have a *consistent DAG extension* π if there exists a permutation of the vertices satisfying the following conditions: (i) for every edge $\{u, v\}$ in G, it is oriented as $u \to v$ whenever $\pi(u) < \pi(v)$, (ii) there are no directed cycles, and (iii) all the given arcs are included in the extension.

Definition 16 (The four Meek rules [Mee95], see Figure 6 for an illustration)**.**

R1 Edge $\{a, b\} \in E \setminus A$ is oriented as $a \to b$ if $\exists\, c \in V$ such that $c \to a$ and $c \not\sim b$.

R2 Edge $\{a, b\} \in E \setminus A$ is oriented as $a \to b$ if $\exists\, c \in V$ such that $a \to c \to b$.

R3 Edge $\{a, b\} \in E \setminus A$ is oriented as $a \to b$ if $\exists\, c, d \in V$ such that $d \sim a \sim c$, $d \to b \leftarrow c$, and $c \not\sim d$.

R4 Edge $\{a, b\} \in E \setminus A$ is oriented as $a \to b$ if $\exists\, c, d \in V$ such that $d \sim a \sim c$, $d \to c \to b$, and $b \not\sim d$.

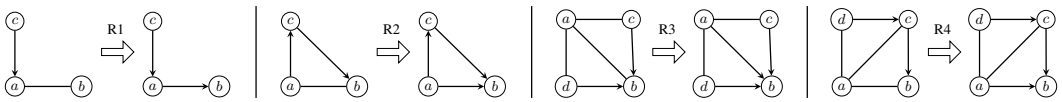


Figure 6: An illustration of the four Meek rules

An algorithm [WBL21, Algorithm 2] has been developed to compute the closure under Meek rules efficiently. The algorithm runs in $\mathcal{O}(d \cdot |E|)$ time, where d represents the degeneracy of the graph skeleton[14].

B Preliminaries and Other Useful Results

Here we state some useful notation and results. For an arc $u \to v$ in $\mathcal{G}$, we define $R_1^{-1}(\mathcal{G}, u \to v) \subseteq V$ and $R_k^{-1}(\mathcal{G}, u \to v) \subseteq 2^V$ to refer to interventions orienting an arc $u \to v$. Equivalently, $R_1^{-1}(\mathcal{G}, u \to v) = \{w \in V : u \to v \in R(\mathcal{G}, w)\}$ and $R_k^{-1}(\mathcal{G}, u \to v) = \{I \subseteq V : |I| \leq k, u \to v \in R(\mathcal{G}, I)\}$.

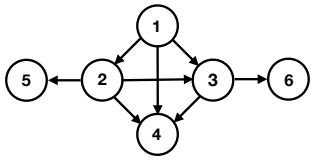
Proposition 17 (Theorem 7 in [CS23])**.** *Consider a DAG $\mathcal{G} = (V, E)$ and intervention sets $\mathcal{I}, \mathcal{J} \subseteq 2^V$. The following statements are true: 1) $skel(\mathcal{G}^{\mathcal{I}})$ is exactly the chain components of $\mathcal{E}_{\mathcal{I}}(\mathcal{G})$. 2) $\mathcal{G}^{\mathcal{I}}$ does not have new v-structures. 3) For any two vertices u and v in the same chain component of $\mathcal{E}_{\mathcal{I}}(\mathcal{G})$, we have $\texttt{Pa}_{\mathcal{G},\mathcal{I}}(u) = \texttt{Pa}_{\mathcal{G},\mathcal{I}}(v)$. 4) If $u \to v \in R(\mathcal{G}, \mathcal{I})$, then u and v belong to different chain components of $\mathcal{E}_{\mathcal{I}}(\mathcal{G})$. 5) Any acyclic completion of $\mathcal{E}(\mathcal{G}^{\mathcal{I}})$ can be combined with $R(\mathcal{G}, \mathcal{I})$ to obtain a valid DAG that has the same essential graph and $\mathcal{I}$-essential graph as $\mathcal{E}(\mathcal{G})$ and $\mathcal{E}_{\mathcal{I}}(\mathcal{G})$, respectively. 6) $R(\mathcal{G}^{\mathcal{I}}, \mathcal{J}) = R(\mathcal{G}, \mathcal{J}) \setminus R(\mathcal{G}, \mathcal{I})$. 7) $R(\mathcal{G}, \mathcal{I} \cup \mathcal{J}) = R(\mathcal{G}^{\mathcal{I}}, \mathcal{J}) \dot\cup R(\mathcal{G}, \mathcal{I})$. 8) $R(\mathcal{G}, \mathcal{I} \cup \mathcal{J}) = R(\mathcal{G}^{\mathcal{I}}, \mathcal{J}) \dot\cup R(\mathcal{G}^{\mathcal{J}}, \mathcal{I}) \dot\cup \big(R(\mathcal{G}, \mathcal{I}) \cap R(\mathcal{G}, \mathcal{J})\big)$.*

Lemma 18 (Theorem 10 in [CS23])**.** *Let $\mathcal{G} = (V, E)$ be a DAG without v-structures and $u \to v$ in $\mathcal{G}$ be unoriented in $\mathcal{E}(\mathcal{G})$. Then, $R_1^{-1}(\mathcal{G}, u \to v) = \texttt{Des}[w] \cap \texttt{Anc}[v]$ for some $w \in \texttt{Anc}[u]$.*

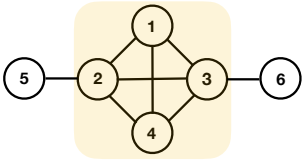
C Another Example of Algorithm 1

We provide another example of Algorithm 1 in an incomplete graph, highlighting that Meek separator by solely focusing on the $1/2$-clique separator.

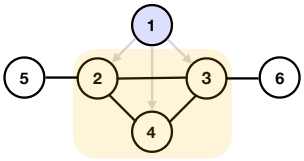
[14] A d-degenerate graph is an undirected graph in which every subgraph has a vertex of degree at most d. Note that the degeneracy of a graph is typically smaller than the maximum degree of the graph.



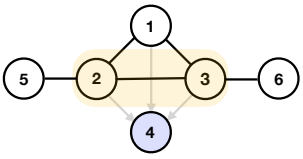
(a) Ground-truth DAG.



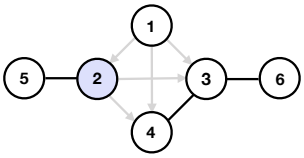
(b) Iteration 0: $K_0 = \{1, 2, 3, 4\}$.



(c) Iteration 1: $u_0 = 1,\ K_1 = \{2, 3, 4\}$.



(d) Iteration 2: $u_1 = 4,\ K_2 = \{2, 3\}$.



(e) Iteration 3: $u_2 = 2$.

Figure 7: An example of Algorithm 1 finding the Meek separator in an incomplete graph. The sets K_i are highlighted; oriented edges in $\mathcal{E}_{u_i}(\mathcal{G})$ by intervening on u_i are in grey; connected components in $CC(\mathcal{E}_{u_i}(\mathcal{G}))$ are in black. **(a)** ground-truth DAG $\mathcal{G}$. **(b)** suppose we take $K_0 = K = V(\mathcal{G})$ as the $1/2$-clique separator. **(c)** suppose we pick $u_0 = 1$ from K_0, then $K_1 = K_0 \cap Q_{u_0} = \{2, 3, 4\}$. **(d)** suppose we pick $u_1 = 4$ from K_1, then $K_2 = K_1 \cap P_{u_1} = \{2, 3\}$. **(e)** suppose we pick $u_2 = 2$ from K_2, then Algorithm 1 terminates (line 7) returning meek separator $\mathcal{I} = \{2\}$ and $\mathcal{J} = \{1, 4\}$ that helps find it.

D Remaining Proofs for Meek Separator

Here we provide all the remaining proofs for the Meek separator algorithm.

D.1 Proof for Lemma 11

Lemma 11 (Connected components)**.** *Let $\mathcal{G}$ be a moral DAG and $v \in V(\mathcal{G})$ be an arbitrary vertex. Any connected component $\mathcal{H} \in CC(\mathcal{E}_v(\mathcal{G}))$ satisfies one of the following conditions: $V(\mathcal{H}) = v$ or $V(\mathcal{H}) \subseteq B_v$ or $V(\mathcal{H}) \subseteq A_v$.*

Proof. Performing an intervention on node v results in orienting all its adjacent edges. Consequently, one of the connected components in $CC(\mathcal{E}_v(\mathcal{G}))$ is $\{v\}$, and it is adequate to focus on the remaining connected components.

Suppose, for the sake of contradiction, that there exists a connected component $\mathcal{H} \in CC(\mathcal{E}_v(\mathcal{G}))$ such that $\mathcal{H}$ contains two vertices a and b, such that, $a \in A_v$ and $b \in B_v$. Since a and b belong to the same connected component, we consider the path within H that connects these vertices. Notably, this path includes two adjacent vertices c and d, where $c \in A_v$ and $d \in B_v$, and the edge (c, d) remains unoriented.

However, according to Lemma 18, intervening on any vertex within the set $\texttt{Des}[w] \cap \texttt{Anc}[d]$ for some fixed $w \in \texttt{Anc}[c]$ will orient edge (c, d). As $w \in \texttt{Anc}[c]$, we have $\texttt{Des}[c] \subseteq \texttt{Des}[w]$ and therefore $\texttt{Des}[c] \cap \texttt{Anc}[d] \subseteq \texttt{Des}[w] \cap \texttt{Anc}[d]$. Consequently as $v \in \texttt{Des}[c] \cap \texttt{Anc}[d]$, we get that intervening on v orients the edge (c, d).

Hence, it is impossible for vertices $a \in A_v$ and $b \in B_v$ to belong to the same connected component. Consequently, we can conclude the proof. □

D.2 Proof for Lemma 12

Lemma 12 (Size of connected components)**.** *Let $\mathcal{G}$ be a graph and K be an α-separator of $\mathcal{G}$. Suppose $\mathcal{H}$ is a connected subgraph of $\mathcal{G}$ and $V(\mathcal{H}) \cap K = \varnothing$, then $|V(\mathcal{H})| \leq \alpha \cdot |V(\mathcal{G})|$.*

Proof. Since $\mathcal{H}$ is a connected subgraph of $\mathcal{G}$ and $V(\mathcal{H}) \cap V(K) = \varnothing$, removing K from $\mathcal{G}$ does not result in the deletion of any edges or vertices from the subgraph $\mathcal{H}$. Consequently, the vertices in $\mathcal{H}$ will remain connected even after the removal of K, and they will all belong to the same connected component in the graph $\mathcal{G}\backslash K$. Considering that K is an α-separator, this further implies that $|V(\mathcal{H})| \leq \alpha \cdot |V(\mathcal{G})|$. Thus, we can conclude the proof. □

D.3 Proof for Lemma 13

Lemma 13 (Properties of connected components)**.** *Consider a moral DAG $\mathcal{G}$ and let K be an α-clique separator. Let $v_1, v_2, \ldots, v_k$ be the vertices of this clique in a valid permutation. We observe that $|B_{v_{i+1}}| \leq |B_{v_i}|$, which in turn implies $|A_{v_{i+1}}| \geq |A_{v_i}|$. Additionally, we have $|A_{v_1}| \leq \alpha \cdot |V(\mathcal{G})|$.*

Proof. Since v_i precedes v_{i+1} in the true ordering defined by the underlying ground truth DAG $\mathcal{G}$, it follows that $v_{i+1} \in \texttt{Des}(v_i)$. Consequently, we can deduce that $\texttt{Des}(v_{i+1}) \subseteq \texttt{Des}(v_i)$, which implies $|B_{v_{i+1}}| \leq |B_{v_i}|$. Additionally, note that $|A_v| + |B_v| + 1 = |V(\mathcal{G})|$. Therefore, it holds that $|A_{v_{i+1}}| \geq |A_{v_i}|$.

Furthermore, as v_1 is the source node of the clique, removing K ensures that all vertices in A_{v_1} are still in the remaining graph. They also induce a connected subgraph. Otherwise suppose $a \in A_{v_1}$ and $b \in A_{v_1}$ are not connected. Then the parent c of v_1 on a path from a to v_1 and the parent d of v_1 on a path from b to v_1 are not connected. This creates a v-structure $c \to v_1 \leftarrow d$, contradicting $\mathcal{G}$ being moral. Thus A_{v_1} is a connected subgraph with no intersection with K. As K is a α-clique separator, we have by Lemma 12 that $|A_{v_1}| \leq \alpha \cdot |V(\mathcal{G})|$. We conclude the proof. □

D.4 Proof for Lemma 10

Lemma 10 (Meek separator)**.** *Let $\mathcal{G}$ be a moral DAG and K be a $1/2$-clique separator of $\mathcal{G}$. There exists a vertex $u \in K$ satisfying the* constraints *$|A_u| \leq |V(\mathcal{G})|/2$ and $|A_v| > |V(\mathcal{G})|/2$ for all $v \in \texttt{Des}(u) \cap K$. Furthermore, such a vertex u satisfies one of the two* conditions*: 1). either u is a sink vertex[15] of $\mathcal{G}[K]$, or 2). there exists a vertex x such that, $x \in \texttt{Des}(u) \cap K$ and $\big((V(\mathcal{G})\backslash \texttt{Des}[x]) \cap \texttt{Des}(u)\big) \cap K = \varnothing$ (i.e., x and u are consecutive vertices in the valid permutation of clique K). In both the cases respectively, either $\{\{u\}\}$ or $\{\{u\}, \{x\}\}$ is a $1/2$-Meek separator.*

Proof. Consider the vertices $v_1, \ldots, v_k$ of the clique K in the true ordering defined by the underlying ground truth $\mathcal{G}$. Since K is a $1/2$-clique separator, according to Lemma 13, we deduce that $|A_{v_1}| \leq |V(\mathcal{G})|/2$. Let u denote the last vertex, in terms of the true ordering, within the clique K that satisfies $A_u \leq |V(\mathcal{G})|/2$. It is important to note that u fulfills the *constraints* specified in the lemma.

Therefore there exists a vertex u that fulfills these *constraints*. Now we show that any vertex that fulfills these constraints will satisfy one of two *conditions*, and that either $\{\{u\}\}$ or $\{\{u\}, \{x\}\}$ is a $1/2$-Meek separator.

Suppose u is a sink vertex of K. Let $\mathcal{H} \in CC(\mathcal{E}_u(\mathcal{G}))$ and note that according to Lemma 11, $V(\mathcal{H}) = \{u\}$ or $V(\mathcal{H}) \subseteq \texttt{Des}(u)$ or $V(\mathcal{H}) \subseteq V(\mathcal{G})\backslash \texttt{Des}[u]$. Since $|A_u| \leq |V(\mathcal{G})|/2$, any $\mathcal{H}$ such that $V(\mathcal{H}) \subseteq V(\mathcal{G})\backslash \texttt{Des}[u]$ satisfies $|V(\mathcal{H})| \leq |V(\mathcal{G})|/2$. Now, suppose $V(\mathcal{H}) \subseteq \texttt{Des}(u)$. In this case, we observe that $V(\mathcal{H}) \cap K = \varnothing$ and $\mathcal{H}$ is a connected subgraph of G. By utilizing Lemma 12, we immediately have $|V(\mathcal{H})| \leq |V(\mathcal{G})|/2$. Therefore, $\{\{u\}\}$ will be a $1/2$-Meek separator.

Suppose u is not a sink vertex. Consider the vertex x that follows u within the clique K in the ordering defined by the DAG $\mathcal{G}$. Note that $x \in \texttt{Des}(u) \cap K$ and $(\texttt{Des}(u) \cap (V(\mathcal{G})\backslash \texttt{Des}[x])) \cap K = \varnothing$. Furthermore from the definition of u, we also have that $|A_x| > |V(\mathcal{G})|/2$, which further implies $|B_x| \leq |V(\mathcal{G})|/2$. As intervening on more vertices only creates finer-grained connected components, we have that for any $\mathcal{J} \subseteq V(\mathcal{G})$, $w \in V(\mathcal{G})$, and $\mathcal{H}' \in CC(\mathcal{E}_{\mathcal{J} \cup \{w\}}(\mathcal{G}))$, there exists an $\mathcal{H}'' \in CC(\mathcal{E}_{\mathcal{J}}(\mathcal{G}))$ such that $\mathcal{H}'$ is a subgraph of $\mathcal{H}''$.

[15] u is a sink vertex if and only if it has no children.

Let $\mathcal{I} = \{x, u\}$, consider any $\mathcal{H} \in CC(\mathcal{E}_{\mathcal{I}}(G))$ and note that there exist connected components $M \in CC(\mathcal{E}_u(\mathcal{G}))$ and $L \in CC(\mathcal{E}_x(\mathcal{G}))$ such that $\mathcal{H}$ is a subgraph of both M and L. We apply Lemma 11 to deduce that $V(M) = u$ or $V(M) \subseteq \texttt{Des}(u)$ or $V(M) \subseteq V(\mathcal{G})\backslash\texttt{Des}[u]$ and $V(L) = x$ or $V(L) \subseteq \texttt{Des}(x)$ or $V(L) \subseteq V(\mathcal{G})\backslash\texttt{Des}[x]$.

If $V(M) = u$ or $V(M) \subseteq V(\mathcal{G})\backslash\texttt{Des}[u]$ or $V(L) = x$ or $V(L) \subseteq \texttt{Des}(x)$, then $V(\mathcal{H}) = u$ or $V(\mathcal{H}) = x$ or $V(\mathcal{H}) \subseteq A_u$ or $V(\mathcal{H}) \subseteq B_x$. As $|A_u|, |B_x| \leq |V(\mathcal{G})|/2$, in all of these cases, we have that $|V(\mathcal{H})| \leq |V(\mathcal{G})|/2$. Now consider the remaining cases where $V(M) \subseteq \texttt{Des}(u)$ and $V(L) \subseteq V(\mathcal{G})\backslash\texttt{Des}[x]$. As $\mathcal{H}$ is a subgraph of both M and L, we have that $\mathcal{H}$ is a subgraph of C, where C satisfies $V(C) = \texttt{Des}(u) \cap (V(\mathcal{G})\backslash\texttt{Des}[x])$. As $(\texttt{Des}(u) \cap (V(\mathcal{G})\backslash\texttt{Des}[x])) \cap K = \varnothing$, we have that $V(C) \cap K = \varnothing$. Furthermore, as $\mathcal{H}$ is a subgraph of C we also have that $V(\mathcal{H}) \cap K = \varnothing$. As $\mathcal{H}$ is a connected subgraph of G and $V(\mathcal{H}) \cap K = \varnothing$, by utilizing Lemma 12, we have that $|V(\mathcal{H})| \leq |V(\mathcal{G})|/2$. Therefore in all cases, we have that $|V(\mathcal{H})| \leq |V(\mathcal{G})|/2$, and thus $\{\{u\}, \{x\}\}$ is a $1/2$-Meek separator, which concludes the proof. □

D.5 Proof for Lemma 14

Here we present a proof for Lemma 14, which describes the structure of the solution returned by our Meek separator algorithm. In order to prove this lemma, we first establish the following result.

Lemma 19. *Let $\mathcal{G}$ be a connected moral DAG and $v \in V(\mathcal{G})$ be an arbitrary vertex. For any connected component $\mathcal{H} \in CC(\mathcal{E}_v(\mathcal{G}))$, there exists a directed path from v to $\mathcal{H}$ in $\mathcal{E}_v(\mathcal{G})$ if and only if $\mathcal{H}$ satisfies $V(\mathcal{H}) \subseteq \texttt{Des}(v)$. Furthermore, we can certify if $\mathcal{H}$ is a subset of $\mathcal{G}\backslash\texttt{Des}[v]$ or $\texttt{Des}(v)$ in polynomial time.*

Proof. Suppose there exists a directed path from v to a connected component $\mathcal{H}$, then note that all the vertices in this directed path are descendants of v and there exists a vertex $u \in V(\mathcal{H})$ such that, $u \in \texttt{Des}(v)$. Furthermore, by Lemma 11, we know that all the connected components $\mathcal{H}$ should satisfy: $V(\mathcal{H}) = \{v\}$ or $V(\mathcal{H}) = \texttt{Des}(v)$ or $V(\mathcal{H}) = V(\mathcal{G})\backslash\texttt{Des}(v)$. Using the fact that there exists a vertex $u \in V(\mathcal{H})$ which belongs to $\texttt{Des}(v)$. Therefore, $\texttt{Des}(v) \cap V(\mathcal{H})$ is not empty and we conclude that $V(\mathcal{H}) \subseteq \texttt{Des}(v)$.

In the subsequent part of the proof, we examine the converse direction. Assume there exists a connected component $\mathcal{H}$ such that $V(\mathcal{H}) \subseteq \texttt{Des}(v)$, and we aim to demonstrate the existence of a directed path from v to $\mathcal{H}$ in the interventional essential graph $\mathcal{E}_v(\mathcal{G})$. Since $\mathcal{H} \subseteq \texttt{Des}(v)$, we consider the shortest directed path from v to $\mathcal{H}$ in $\mathcal{G}$. Let w be the endpoint of this path P within $\mathcal{H}$.

Suppose that all edges in P are oriented. In this case, we have already found a directed path from v to $\mathcal{H}$ in $\mathcal{E}_v(\mathcal{G})$, and our objective is achieved. Hence, we focus on the scenario where some edges in P are unoriented. Let $P : v = v_0 \to v_1 \to \cdots \to v_\ell \to w$ denote the vertices along the path P, and let $v_j \sim v_{j+1}$ represent the first unoriented edge encountered.

Since all edges incident to v are oriented, we have $v_j \neq v$. Now, consider the situation where v_{j-1} and v_{j+1} are not adjacent. In such a case, the Meek rule R1 would orient the edge $v_j \to v_{j+1}$. However, since $v_j \to v_{j+1}$ remains unoriented, it implies that the edge $v_{j-1} \to v_{j+1}$ must exist. However, this contradicts the fact that P is the shortest directed path. Therefore, it must be the case that P is either entirely oriented or entirely unoriented. As some edges in P are oriented, we conclude that path P is completely oriented, thereby establishing the presence of a directed path from v to $\mathcal{H}$ in the interventional essential graph $\mathcal{E}_v(\mathcal{G})$.

In the remaining part of the proof, we discuss the time complexity of finding these directed paths. To certify whether $V(\mathcal{H}) \subseteq \texttt{Des}(v)$ or $V(\mathcal{H}) \subseteq V(\mathcal{G})\backslash\texttt{Des}[v]$, all we need to do is check whether a directed path exists between v and any $u \in V(\mathcal{H})$. If the direction of the path is from v to $\mathcal{H}$, then $V(\mathcal{H}) \subseteq \texttt{Des}(v)$ or else $V(\mathcal{H}) \subseteq V(\mathcal{G})\backslash\texttt{Des}[v]$. To check whether a directed path exists between two vertices can be done in polynomial time and therefore we can certify if $V(\mathcal{H}) \subseteq \texttt{Des}(v)$ or $V(\mathcal{H}) \subseteq V(\mathcal{G})\backslash\texttt{Des}[v]$. We conclude the proof. □

Lemma 14 (Output of MeekSeparator)**.** *The algorithm* MeekSeparator *performs at most $\mathcal{O}(\log |K|)$ interventions in expectation and finds a vertex $u \in K$ that is either a $1/2$-Meek separator or satisfies the following conditions: $|A_u| \leq {}^{|V(\mathcal{G})|}/_2$ and $|A_v| > {}^{|V(\mathcal{G})|}/_2$ for all $v \in \texttt{Des}(u) \cap K$.*

Proof. Consider a clique K and a true ordering π defined on the vertices $V(\mathcal{G})$ of the underlying ground truth DAG $\mathcal{G}$. Let $v_1, v_2, \ldots, v_k$ represent the vertices in the clique K, labeled according to

the true ordering π. In other words, v_i precedes v_j in π whenever $i < j$. Since K is a $1/2$-Clique separator, according to Lemma 10, we have $|A_{v_1}| \leq |V(\mathcal{G})|/2$. Additionally, based on Lemma 13, we know that $|A_{v_j}| \leq |A_{v_{j+1}}|$ for all $j \in [1, k-1]$. Let v_{j^*} be the last vertex within the clique K in the true ordering π such that $|A_{v_{j^*}}| \leq |V(\mathcal{G})|/2$.

In our proof, we demonstrate that our algorithm can either discover a $1/2$-Meek separator or identify the vertex v_{j^*} within $\mathcal{O}(\log |K|)$ interventions, on average. If, at any stage of the algorithm, u_i is a $1/2$-Meek separator, our task is complete. Consequently, for the remainder of the proof, we concentrate on the alternative scenario and establish that our algorithm locates v_{j^*}. In this case, we observe that either v_{j^*} corresponds to the sink node of K or we uncover the subsequent vertex, v_{j^*+1}, in the ordering. Notably, according to Lemma 10, either case implies that $\mathcal{I} = \{u\}$ or $\mathcal{I} = \{u, x\}$ constitutes a $1/2$-Meek separator.

During each iteration i of our algorithm, we intervene on a uniformly random chosen vertex u_i from the remaining clique K_i. Let $\mathcal{H}_i$ represent the largest connected component in the interventional essential graph $\mathcal{E}_{u_i}(\mathcal{G})$. The algorithm terminates when the size of the clique becomes empty. Notably, when we intervene on a vertex u_i, by Lemma 19, the existence of a directed path from u_i to $\mathcal{H}_i$ implies that $\mathcal{H}_i$ corresponds to a connected component in the descendant subgraph of u_i. Since $|V(\mathcal{H}_i)| > |V(\mathcal{G})|/2$, it follows that $|\texttt{Des}(u_i)| > |V(\mathcal{G})|/2$, which further implies $|A_{u_i}| \leq |V(\mathcal{G})|/2$. We assign $u = u_i$ and observe that $|\texttt{Des}(u)| = |\texttt{Des}(u_i)| > |V(\mathcal{G})|/2$, implying $|A_u| \leq |V(\mathcal{G})|/2$. In this scenario, we recursively proceed with the set $Q_{u_i} \cap K_i$, which comprises the nodes that appear after the vertex u_i in the true ordering π. It is worth noting that these vertices $y \in Q_{u_i} \cap K_i$ satisfy $|A_y| \geq |A_{u_i}|$ since $y \in \texttt{Des}(u_i)$ (Lemma 10).

In the other case, when no path exists from u_i to $\mathcal{H}_i$, by Lemma 19, we deduce that $V(\mathcal{H}_i) \subseteq A_{u_i}$, thereby leading to $|A_{u_i}| > V(\mathcal{G})/2$. Consequently, we assign $x = u_i$ and therefore $|A_x| > V(\mathcal{G})/2$. In this situation, we recursively proceed with the set $P_{u_i} \cap K_i$, which represents the vertices appearing before the vertex u_i in the true ordering π.

In both cases, it is important to note that K_i always consists of a contiguous set of vertices (defined by the true ordering) within the clique K. Let s_{i+1} and t_{i+1} denote the source and sink vertices, respectively, in the remaining clique K_{i+1}. In the first case, where $K_{i+1} = Q_{u_i} \cap K_i$, the vertex $u = u_i$ satisfies $\texttt{Des}(u) \cap (V(\mathcal{G}) \backslash \texttt{Des}(s_{i+1})) \cap K = \varnothing$. In the latter case, where $K_{i+1} = P_{u_i} \cap K_i$, the vertex $x = u_i$ satisfies $\texttt{Des}(t_{i+1}) \cap (V(\mathcal{G}) \backslash \texttt{Des}(x)) \cap K = \varnothing$. In simpler terms, this means that vertex u (immediate parent) precedes the remaining clique K_{i+1} in the true ordering π, while vertex x (immediate child) succeeds it within the clique K.

Our algorithm terminates when the remaining clique becomes empty, implying that either u is a sink vertex or u and x are consecutive vertices within the clique. In other words, $\texttt{Des}(u) \cap (V(G) \backslash \texttt{Des}(x)) \cap K = \varnothing$. Therefore, the solution returned by our algorithm satisfies the conditions of the lemma. The only remaining task is to bound the number of interventions required by our algorithm.

To bound the number of interventions or iterations, we need to analyze the decrease in the size of the clique K_i at each iteration. Recall that K_i consists of a contiguous set of vertices from the original clique K, and in each iteration, we randomly select a vertex u_i. As discussed earlier, our algorithm either outputs a $1/2$-Meek separator at some intermediate step or performs a randomized binary search to locate the vertices v_{j^*} and v_{j^*+1} (if it exists), satisfying $|A_{v_{j^*}}| \leq |V(\mathcal{G})|/2$ and $|A_{v_{j^*+1}}| > |V(\mathcal{G})|/2$.

Since the parents and children of a vertex v within the clique K are known after intervening on it, the algorithm essentially performs a binary search to find the vertices v_{j^*} and v_{j^*+1}. The standard analysis of randomized binary search provides an expected upper bound of $\mathcal{O}(\log |K|)$ iterations.

Combining these insights, we can conclude the proof by establishing that the expected number of iterations is bounded by $\mathcal{O}(\log |K|)$. □

D.6 Proof for Theorem 5

Theorem 5 (Meek separator). *Given an essential graph $\mathcal{E}(\mathcal{G})$ of an unknown DAG $\mathcal{G}$, there exists a randomized procedure* MeekSeparator *(given in Algorithm 1) that runs in polynomial time and*

adaptively intervenes on a set of atomic interventions $\mathcal{I}$ such that we can find a $1/2$-Meek separator of size at most 2 and $\mathbb{E}[|\mathcal{I}|] \leq \mathcal{O}(\log \omega(\mathcal{G}))$,[16] where $\omega(\mathcal{G})$ denotes the size of the largest clique in $\mathcal{G}$.

Proof. The proof of the theorem for a moral DAG follows immediately by combining Lemma 14 and Lemma 10.

In the remainder, we extend our result to encompass general directed acyclic graphs. While an informal argument for general DAGs has been provided at the beginning of Section 4, we now delve into further details to substantiate that argument. Let us recall the definition of $\mathcal{G}^{\mathcal{I}} = \mathcal{G}[E \setminus R(\mathcal{G}, \mathcal{I})]$ and focus on the graph $\mathcal{G}^{\varnothing}$. This subgraph is derived by removing both the v-structure edges and the oriented edges resulting from the application of Meek rules. According to Proposition 17, we establish that $\mathcal{G}^{\varnothing}$ does not introduce any new v-structures and, therefore, is a moral DAG.

To proceed, let $\mathcal{H} \in CC(\mathcal{E}_{\varnothing}(\mathcal{G}))$ be the largest connect component within $\mathcal{E}_{\varnothing}(\mathcal{G})$ and designate $\mathcal{I}$ as a $1/2$-Meek separator of $\mathcal{H}$. Note that $R(\mathcal{H}, \mathcal{I}) \subseteq R(\mathcal{G}^{\varnothing}, \mathcal{I})$ and as highlighted in Proposition 17, we also determine that $R(\mathcal{G}^{\varnothing}, \mathcal{I}) = R(\mathcal{G}, \mathcal{I}) \setminus R(\mathcal{G}, \varnothing)$. Consequently, the connected components within both the intervention essential graphs $\mathcal{E}_{\mathcal{I}}(\mathcal{G})$ and $\mathcal{E}_{\mathcal{I}}(\mathcal{G}^{\varnothing})$ are identical. Since $\mathcal{I}$ is a $1/2$-Meek separator for $\mathcal{G}^{\varnothing}$, it also functions as a half $1/2$-Meek separator for $\mathcal{G}$. □

E Remaining Proofs for Subset Search

Here we present all the proofs for the subset search problem. The proof for the lower bound and the upper bound results are presented in Appendix E.1 and Appendix E.2 respectively.

E.1 Lower Bound for Subset Search (Lemma 6)

Here we prove our lower bound result for the subset verification problem.

Lemma 6 (Lower bound). *Let $\mathcal{G} = (V, E)$ be a DAG and $T \subseteq E$ be a subset of target edges, then, $\nu_1(\mathcal{G}, T) \geq \max_{I \subseteq V} \sum_{\mathcal{H} \in CC(\mathcal{E}_I(\mathcal{G}))} \mathbb{1}(E(\mathcal{H}) \cap T \neq \varnothing)$.*

Proof. Consider an intervention set $\mathcal{I}$ and let $CC(\mathcal{E}_{\mathcal{I}}(\mathcal{G}))$ be the set of all connected components in the intervention essential graph $\mathcal{E}_{\mathcal{I}}(\mathcal{G})$. As interventions within each connected components are independent[HB14], we have that,

$$\nu_1(\mathcal{G}, T) \geq \sum_{\mathcal{H} \in CC(\mathcal{E}_{\mathcal{I}}(\mathcal{G}))} \nu_1(\mathcal{H}, T \cap E(\mathcal{H})) \ .$$

For each $\mathcal{H} \in CC(\mathcal{E}_{\mathcal{I}}(\mathcal{G}))$, where $T \cap E(\mathcal{H})$ is non-empty, it is trivial that $\nu_1(\mathcal{H}, T \cap E(\mathcal{H})) \geq 1$ as we need at least one intervention to orient all the edges in $T \cap E(\mathcal{H})$. Therefore the lower bound follows, which concludes the proof. □

E.2 Upper Bound for Subset Search (Theorem 7)

Here we present comprehensive proof for the upper bound result. Although a proof sketch of this result has already been presented in Section 5, we now provide additional details to solidify our argument.

Theorem 7 (Upper bound). *Let $\mathcal{G} = (V, E)$ be a DAG with $|V| = n$ and $T \subseteq E$ be a subset of target edges. Algorithm 2 that takes essential graph $\mathcal{E}(\mathcal{G})$ and T as input, runs in polynomial time and adaptively intervenes on a set of atomic interventions $\mathcal{I} \subseteq V$ that satisfies, $\mathbb{E}[|\mathcal{I}|] \leq \mathcal{O}\big(\log n \cdot \log \omega(\mathcal{G})\big) \cdot \nu_1(\mathcal{G}, T)$ and $T \subseteq R(\mathcal{G}, \mathcal{I})$. Furthermore, in the special case of $T = E$, the solution returned by it satisfies, $\mathbb{E}[|\mathcal{I}|] \leq \mathcal{O}(\log n) \cdot \nu_1(\mathcal{G})$ and $E \subseteq R(\mathcal{G}, \mathcal{I})$.*

Proof. As discussed in Section 5, the correctness of the Algorithm 2 follows because of the termination condition of the algorithm. Note that, upon termination, all the connected components that encompass the target edges T have a size of 1. This observation leads to the immediate implication that all the edges belonging to T are oriented.

[16] The expectation in the result is over the randomness of the algorithm.

In the remaining part of the proof, we analyze the cost of the algorithm and divide the analysis into two parts: the number of outer loops and the cost per loop.

The bound on the number of outer loops is straightforward. Since the size of connected components containing the target edges decreases at least by a factor of $1/2$ in each iteration, and after $\mathcal{O}(\log n)$ iterations all connected components either consist of a single vertex or do not have any incident target edges T, therefore our algorithm terminates in $\mathcal{O}(\log n)$ iterations.

To bound the cost per loop, note that, we invoke the Meek separator only on the connected components $\mathcal{H}$ that have at least one target edge. For each such component $\mathcal{H}$, Lemma 6 establishes that any algorithm must perform at least one intervention on this component. Our Meek separator algorithm performs at most $\mathcal{O}(\log \omega(\mathcal{H})) \in \mathcal{O}(\log \omega(\mathcal{G}))$ interventions for each component $\mathcal{H}$. Hence, in any iteration, we perform at most $O(\log \omega(\mathcal{G}))\nu_1(\mathcal{G}, T)$ interventions. It is important to mention that when $T = E$, we use the lower bound from [SMG+20] which states that any algorithm would require at least $\Omega(\omega(\mathcal{H}))$ interventions to orient all edges within the connected component $\mathcal{H}$, whereas we only perform $O(\log \omega(\mathcal{H}))$ interventions in that iteration. Consequently, in the special case of $T = E$, the cost per iteration is at most $O(\nu_1(\mathcal{G}, E))$.

Combining these analyses, we conclude that our algorithm orients all the edges in T by performing at most $\mathcal{O}(\log n \log \omega(\mathcal{G}))\nu_1(\mathcal{G}, T)$ interventions. In the special case of $T = E$, our total number of interventions is at most $\mathcal{O}(\log n) \cdot \nu_1(\mathcal{G})$. This completes the proof. □

F Remaining Proofs for Causal Mean Matching

Similar to the subset search, we use the Meek separator as a subroutine to provide an approximation algorithm. A crucial step of our algorithm is a source finding algorithm, which given an essential graph $\mathcal{E}(\mathcal{G})$ and a subset of nodes $U \subseteq V(\mathcal{G})$, returns a source node of U by performing at most $\mathcal{O}(\log n \cdot \log \omega(\mathcal{G}))$ number of interventions. Such a source-finding algorithm immediately helps us solve the causal state-matching problem. In the remainder of the section we provide the source finding algorithm and use it solve the causal state matching problem. To understand our source finding algorithm, we need the following lemma.

Lemma 20. *Let $\mathcal{G} = (V, E)$ be a DAG, $\mathcal{I} \subseteq V$ be an intervention set and let $S, \mathcal{H} \in \mathcal{E}_{\mathcal{I}}(\mathcal{G}^*)$. Suppose there exists a directed path from S to $\mathcal{H}$, then no directed path exists from $\mathcal{H}$ to S. Furthermore, if $s, t \in V$ be such that $t \in \texttt{Des}(s)$ and they are not in the same connected component, then there exists a directed path that is oriented from the connected component containing s to the connected component containing t.*

Proof. We are given that there exists a directed path from S to $\mathcal{H}$. As there exists a directed path from S to $\mathcal{H}$, there exist a vertex $s \in V(S)$ and $t \in V(\mathcal{H})$ such that $t \in \texttt{Des}(s)$.

Suppose for a contradiction assume that there exists a directed path from $\mathcal{H}$ to S. Let u and v be the endpoint of this directed path in $\mathcal{H}$ and S respectively. By Proposition 17, we know that all the recovered parents for the vertices within the same connected component are the same; therefore we have that, $\texttt{Pa}_{u,\mathcal{I}}(\mathcal{G}) = \texttt{Pa}_{t,\mathcal{I}}(\mathcal{G})$ and $\texttt{Pa}_{s,\mathcal{I}}(\mathcal{G}) = \texttt{Pa}_{v,\mathcal{I}}(\mathcal{G})$. Let t' be the parent of t on the directed path that connects the vertices s and t. Note that $t' \in \texttt{Des}[s]$ and t' could be s or some other vertex. As s and t belong to different connected components $t' \to t$ is oriented. As $\texttt{Pa}_{u,\mathcal{I}}(\mathcal{G}) = \texttt{Pa}_{t,\mathcal{I}}(\mathcal{G})$, we have that $t' \in \texttt{Pa}_{u,\mathcal{I}}(\mathcal{G})$, therefore $u \in \texttt{Des}(t')$ which further implies $u \in \texttt{Des}(s)$. As $v \in \texttt{Des}(u)$ and as they are in different connected components, a similar argument as above implies that $s \in \texttt{Des}(u)$, which is a contradiction. Therefore $\mathcal{H}$ to S directed path does not exist. The analysis conducted above verifies the first claim of the lemma. In the subsequent portion, we focus on establishing the second part.

Consider vertices s and t such that $t \in \texttt{Des}(s)$. If both s and t belong to the same connected component in the interventional essential graph $\mathcal{E}_{\mathcal{I}}(\mathcal{G})$, the lemma statement holds trivially. Henceforth, we assume that s and t belong to distinct connected components, denoted as S and T respectively.

Since $t \in \texttt{Des}(s)$, there exists a directed path from vertex s to vertex t in the ground truth DAG $\mathcal{G}$. Let us denote Q as the shortest directed path from s to t in $\mathcal{G}$. To form path P, we remove the edges within the connected components S and T from Q and retain only the edges that connect S and T. Consequently, P represents this modified portion of path Q. Furthermore, let v and w be the

respective endpoints of path P in S and T. As S and T are two different connected components, we have that some of the edges in P are oriented.

Suppose that all edges in P are oriented. In this case, we have already found a directed path from S to T in $\mathcal{E}_{\mathcal{I}}(\mathcal{G})$, and our objective is achieved. Hence, we focus on the scenario where some edges in P are unoriented. Let $P : v = v_0 \to v_1 \to \cdots \to v_\ell \to w$ denote the vertices along the path P, and let $v_j \sim v_{j+1}$ represent the first unoriented edge encountered.

Since v_1 does not belong to the connected component S, we have that $v \to v_1$ is oriented and we have $v_j \neq v$. Now, consider the situation where v_{j-1} and v_{j+1} are not adjacent. In such a case, the Meek rule R1 would orient the edge $v_j \to v_{j+1}$. However, since $v_j \to v_{j+1}$ remains unoriented, it implies that the edge $v_{j-1} \to v_{j+1}$ must exist. However, this contradicts the fact that P is the shortest directed path. Therefore, it must be the case that P is either entirely oriented or entirely unoriented. As some edges in P are oriented, we conclude that path P is completely oriented, thereby establishing the presence of a directed path from S to T in the interventional essential graph $\mathcal{E}_{\mathcal{I}}(\mathcal{G})$, which concludes the proof. □

F.1 Proof for Lemma 8

We now provide the analysis of the source-finding algorithm. The guarantees of this algorithm are summarized in the lemma that follows.

Lemma 8 (Source finding)**.** *Let $\mathcal{G} = (V, E)$ be a DAG and $U \subseteq V$ be a subset of vertices. Algorithm 3 that takes essential graph $\mathcal{E}(\mathcal{G})$ and U as input, runs in polynomial time and adaptively intervenes on a set of atomic interventions $\mathcal{I} \subset V$, identifies a source vertex of the induced subgraph $\mathcal{G}[U]$ with $\mathbb{E}[|\mathcal{I}|] \leq \mathcal{O}\big(\log n \cdot \log \omega(\mathcal{G})\big)$.*

Proof. As $\mathcal{J}_i$ is a $1/2$-Meek separtors, we immediately have that $|V(\mathcal{H}_{i+1})| \leq |V(\mathcal{H}_i)|/2$. Therefore our algorithm terminates in at most $\mathcal{O}(\log n)$ iterations. By Theorem 5, note that, in each iteration, we make at most $O(\log \omega(\mathcal{H}_i)) \in \mathcal{O}(\log \omega(\mathcal{G}^*))$ number of iterations. Therefore, the total number of interventions are at most $\mathcal{O}(\log n \cdot \log \omega(\mathcal{G}^*))$. It remains to show that we find the source node of U.

As in each iteration we recurse on connected component $\mathcal{H}_i \in C_i$ that has no incoming edge from any other component $\mathcal{H} \in C_i$. From Lemma 20, we know that $\mathcal{H}_i$ contains one of the source nodes of U. Therefore our algorithm always recurses on a connected component containing a source node until the algorithm finds it. □

Algorithm 4 CausalMeanMatch$(\mathcal{E}(\mathcal{G}), P, \mu^*)$

Input: Essential graph $\mathcal{E}(\mathcal{G})$ of a DAG $\mathcal{G}$, observational distribution of V, and desired mean μ^*.
Output: Atomic intervention set $\mathcal{I}$.
Initialize $\mathcal{I}^* = \varnothing$ and $\mathcal{I} = \{\varnothing\}$.
while $\mathbb{E}_{P^{\mathcal{I}^*}}(V) \neq \mu^*$
  Let $T = \{i | i \in [p], \mathbb{E}_{P^{\mathcal{I}^*}}(V_i) \neq \mu_i^*\}$.
  Let $\mathcal{G}$ be the subgraph of $\mathcal{E}_{\mathcal{I}}(\mathcal{G})$ induced by T.
  Let U_T be the identified source nodes in T.
  while $U_T = \varnothing$
    Let $\mathcal{C}$ be a chain component of $\mathcal{G}$ with no incoming edges.
    Perform interventions by FindSource$(\mathcal{E}(\mathcal{G}), S)$ and append interventions to $\mathcal{I}$.
    Update $\mathcal{G}$ and U_T as the outer loop.
  Set $a_i = \mu_i^* - \mathbb{E}_{P^{\mathcal{I}^*}}(V_i)$ for i in U_T.
  Include the atomic interventions with perturbation targets U_T and shift values $\{a_i\}_{i \in U_T}$ respectively in $\mathcal{I}^*$ and perform $\mathcal{I}^*$.
return $\mathcal{I}^*$

F.2 Proof for Theorem 9

Given such a source-finding algorithm, we can use it to solve the mean matching problem. We summarize this result below.

Theorem 9 (Causal mean matching)**.** *Let $\mathcal{G}$ be a DAG and $\mathcal{I}^*$ be the unique solution to the causal mean matching problem with desired mean μ^*. Algorithm 4 that takes $\mathcal{E}(\mathcal{G})$ and μ^* as input, runs in polynomial time and adaptively intervenes on set $\mathcal{I} \subseteq V$, identifies $\mathcal{I}^*$ with $\mathbb{E}[|\mathcal{I}|] \leq \mathcal{O}\big(\log n \cdot \log \omega(\mathcal{G})\big) \cdot |\mathcal{I}^*|$.*

Proof. Note the outer loop in Algorithm 4 takes at most $|\mathcal{I}^*|$ round. In each of this round, the inner loop is ended with at most $\mathcal{O}(\log n \log \omega(\mathcal{G}^*))$ interventions in expectation, as proven by Lemma 8. Therefore, in expectation, the number of interventions performed is upper bounded by $\mathcal{O}(\log n \log \omega(\mathcal{G}^*))|\mathcal{I}^*|$. □

G Details of Numerical Experiments

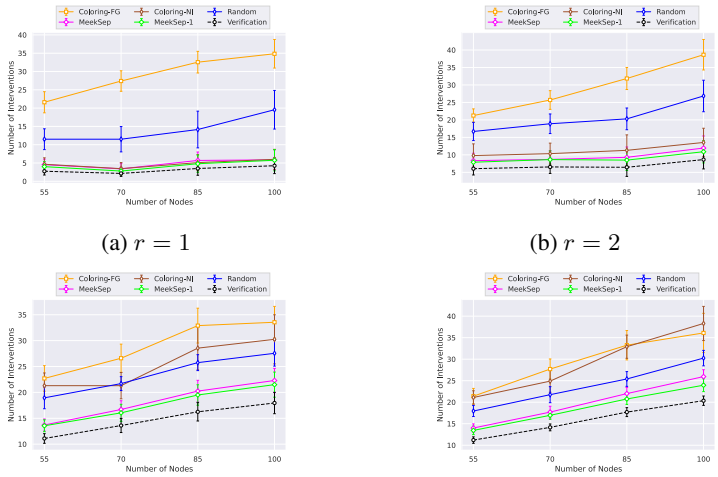
We implemented our methods using the NetworkX package [HSSC08] and the CausalDAG package `https://github.com/uhlerlab/causaldag`. All code is written in Python and run on CPU. The source code of our implementation can be found at `https://github.com/uhlerlab/meek_sep`.

G.1 Subset Search

Problem Generation: We consider the r-hop model in [CS23]. In this model, an Erdös-Rényi graph [ER60] with edge density 0.001 on n nodes is first generated. Then a random tree on these n nodes is generated. The final DAG is obtained by (1) combining the edge sets using a fixed topological order, where $u \to v$ if it is in the combined edge sets and u has a smaller vertex label than v, and (2) removing v-structures by connecting $u \to w$ where $u \to v \leftarrow w$ and u has a smaller vertex label than w. Then the subset of edges is selected to be the edges within r-neighborhood of a randomly picked vertex.

Multiple Runs: For each dot presented in the results, we run each method on 20 different instances using the generation method described above. The average and standard deviation across instances are reported in the results.

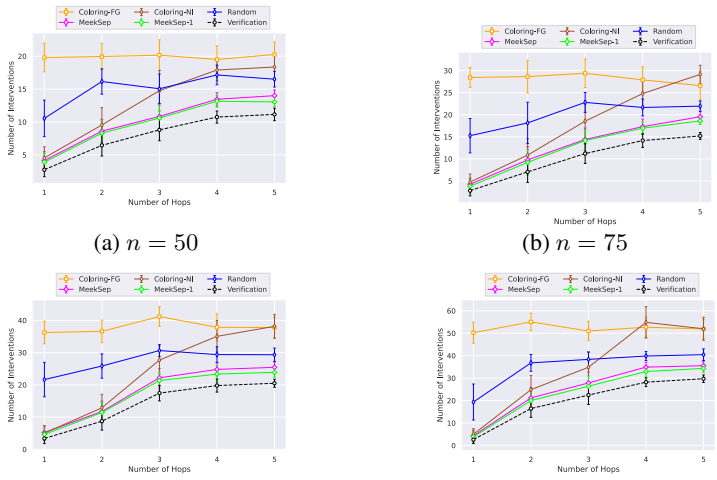
Figure 8 shows similar results as Figure 5a on the 3-hop model, where we vary the number of hops $r \in \{1, 2, 4, 5\}$ on DAGs with different sizes. Figure 9 shows the trend of varying number of hops on DAGs with different sizes. We observe our method `MeekSep` and `MeekSep-1` to consistently outperform existing baselines.



(a) $r = 1$ (b) $r = 2$

(c) $r = 4$ (d) $r = 5$

Figure 8: Meek separator for subset search on r-hop model. Each dot is averaged across 20 DAGs, where the error bar shows 0.5 standard deviation.



(a) $n = 50$ (b) $n = 75$

(c) $n = 100$ (d) $n = 150$

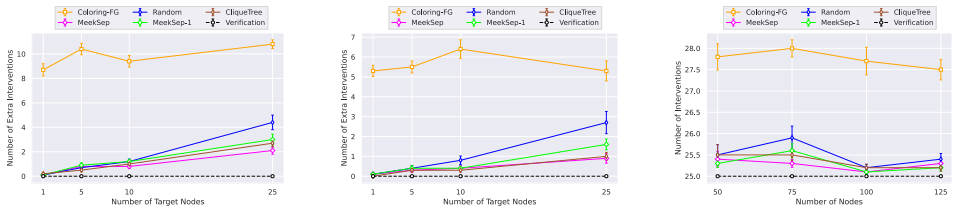
Figure 9: Meek separator for subset search on r-hop model. Each dot is averaged across 20 DAGs, where the error bar shows 0.5 standard deviation.

G.2 Causal Mean Matching

Problem Generation: We consider three random graph models: Erdös-Rényi graphs, Barabási–Albert graphs [AB02], and random tree graphs. The edge density in Erdös-Rényi graphs is 0.2 where the number of edges to attach from a new node to existing nodes in Barabási–Albert graphs is set to 2. The intervention targets of $\mathcal{I}^*$ is a random subset of n vertex in the DAG.

Multiple Runs: For each dot presented in the result, we run each method on 10 different instances using the generation method described above. The average and standard deviation across instances are reported in the results.

Figure 5b shows the result on Erdös-Rényi graphs, where we vary the number of targets in $\mathcal{I}^*$ on DAGs with 50 nodes. Figure 10a and Figure 10b show similar results on random tree graphs and Barabási–Albert graphs. In Figure 10c, we consider Erdös-Rényi graphs where $|\mathcal{I}^*|$ is set to 25. This result shows the trend of varying number of nodes. We observe that our method is empirically competitive with the state-of-the-art method `CliqueTree` across all cases.



(a) Random tree graphs (b) Barabási–Albert graphs (c) $|\mathcal{I}^*| = 25$

Figure 10: Meek separator for causal matching. Each dot is averaged across 10 DAGs, where the error bar shows 0.2 standard deviation