Local Markov Equivalence and Local Causal Discovery for Identifying Controlled Direct Effects

Timothée Loranchet[1] **Charles K. Assaad**[1]

[1]Sorbonne Université, INSERM,, Institut Pierre Louis d'Epidémiologie, et de Santé Publique,, F75012, Paris, France

Abstract

Understanding and identifying controlled direct effects (CDEs) is crucial across numerous scientific domains, including public health. While existing methods can identify these effects from causal directed acyclic graphs (DAGs), the true underlying structure is often unknown in practice. Essential graphs, which represent a Markov equivalence class of DAGs characterized by the same set of d-separations, provide a more practical and realistic alternative. However, learning the full essential graph is computationally intensive and typically depends on strong, untestable assumptions. In this work, we characterize a local class of graphs, defined relative to a target variable, that share a specific subset of d-separations, and introduce a graphical representation of this class, called the local essential graph (LEG). We then present **LocPC**, a novel algorithm designed to recover the LEG from an observed distribution using only local conditional independence tests. Building on **LocPC**, we propose **LocPC-CDE**, an algorithm that discovers the portion of the LEG that is both sufficient and necessary to identify a CDE, bypassing the need of retrieving the full essential graph. Compared to global methods, our algorithms require less conditional independence tests and operate under weaker assumptions while maintaining theoretical guarantees. We illustrate the effectiveness of our approach through simulation studies.

1 INTRODUCTION

Understanding controlled direct effects [Pearl, 2000, 2001] (CDEs) is fundamental to causal inference across a wide array of scientific fields, including public health [Vansteelandt, 2009, Vanderweele, 2010] and industries [Assaad et al., 2023]. CDEs quantify how changes in one variable influence another independently of any mediating pathways, offering valuable insight into mechanisms of action and informing targeted interventions. For instance, suppose an epidemiological study on the effect of physical exercise on cardiovascular health, where estimating the CDE reveals that exercise improves heart outcomes even without weight loss. This insight is critical as it shows that interventions promoting exercise should be encouraged even in individuals who do not lose weight, shifting public health messaging and clinical advice to focus on exercise benefits beyond weight control.

Numerous tools exist for identifying CDEs when the underlying causal structure is known and can be represented as a directed acyclic graph (DAG). However, in many practical scenarios, the true DAG is unknown. Instead, one can often recover, under the *causal sufficiency* and the *faithfulness* assumptions [Spirtes et al., 2000], an essential graph [Andersson et al., 1997], which encodes a Markov equivalence class of DAGs consistent with the observed conditional independencies. While global causal discovery methods such as the PC algorithm [Spirtes et al., 2000] aim to recover the entire essential graph, they are often computationally intensive and require large amounts of data—particularly in high-dimensional settings where only a few causal relationships are of practical interest. In many real-world applications, these methods often fail, likely due to the violation of the strong and untestable assumptions they rely on [Uhler et al., 2012, Andersen, 2013, Aït-Bachir et al., 2023].

This motivates the development of local causal discovery methods that concentrate solely on the relevant subgraph surrounding the variables of interest. Restricting the discovery process to a local neighborhood, would not only reduce computational complexity but also enhance robustness, while still yielding valid and actionable causal insights. For instance, Gao and Ji [2015] proposed an algorithm to discover the immediate neighborhood of a target variable. Other works proposed algorithms aimed at discovering parts of the essential graph sufficient for identifying total effects

Accepted for the 1st Workshop on Causal Abstractions and Representations (CAR) at UAI 2025 (CAR 2025).

using only local information [Maathuis et al., 2008, Malinsky and Spirtes, 2016, Hyttinen et al., 2015, Gupta et al., 2023, Geffner et al., 2024]. Nevertheless, the problem of local causal discovery targeted for identifying a CDE remains largely unaddressed. Moreover, the characterization of the class of graphs recoverable through local information has received little attention in the literature.

In this paper, we characterize the class of graphs that share a specific notion of locality around a target variable Y that dependents on a targeted neighborhood hop distance specified by the user. We show that this class can be uniquely represented by a local essential graph (LEG) and introduce the **LocPC** algorithm for recovering the LEG from data, requiring weaker assumptions than those needed for recovering the full essential graph. Furthermore, we demonstrate that a naive application of **LocPC** can serve for local causal discovery aimed at identifying a CDE. Finally, we develop an improved version of the algorithm, called the **LocPC-CDE** algorithm that optimally discovers only the part of the LEG that is both necessary and sufficient for CDE identification in a non-parametric setting.

The remainder of the paper is organized as follows: Section 2 introduces preliminaries. Section 3 presents a characterization of all graphs that have the same local information. Section 4 presents the **LocPC** algorithm. Section 5 starts by showing how a naive application of **LocPC** can be used for identifying a CDE and then presents the **LocPC-CDE** algorithm. In Section 6 we evaluate the proposed algorithms on synthetic linear and non-linear datasets. Finally, Section 7 concludes the paper. All proofs are deferred to the supplementary material.

2 PRELIMINARIES

We use capital letters (Z) for variables, bold letters $(\mathbb{Z})$ for sets of variables, and $|\mathbb{Z}|$ for their size.

In this work, we rely on the framework of Structural Causal Models (SCMs) as introduced by Pearl [2000]. Formally, an SCM $\mathcal{M}$ is defined as a 4-tuple $(\mathbb{U}, \mathbb{V}, \mathbb{F}, P(\mathbb{U}))$, where $\mathbb{U}$ represents a set of exogenous variables and $\mathbb{V}$ denotes a set of endogenous variables. The set $\mathbb{F}$ contains causal mechanisms, each determining an endogenous variable from a corresponding exogenous variable and a subset of other endogenous variables, usually referred to as direct causes or parents. We assume that the SCM induces a directed acyclic graph (DAG) $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ consisting of a set of vertices $\mathbb{V}$ and directed edges $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$. Additionally, we assume no hidden confounding, also known as causal sufficiency.

Assumption 1 (Causal sufficiency)**.** *All exogenous variables are mutually independent and each influences only a single endogenous variable.*

In $\mathcal{G}$, a parent of $W_l \in \mathbb{V}$ is any $W_m \in \mathbb{V}$ such that $W_m \to W_l$ is in $\mathbb{E}$. The set of parents of W_l is denoted $Pa(W_l, \mathcal{G})$. The children of W_l, denoted $Ch(W_l, \mathcal{G})$, are the variables W_m such that $W_l \to W_m$ is in $\mathbb{E}$. The ancestors of W_l, $An(W_l, \mathcal{G})$, are all variables with a directed path to W_l, while its descendants, $De(W_l, \mathcal{G})$, are those reachable by a directed path from W_l. The neighbors of W_l, $Ne(W_l, \mathcal{G})$, are all variables connected to W_l in $\mathcal{G}$. The h-hop neighborhood of a target variable Y, denoted $Neighborhood(Y, h, \mathcal{G})$ is the set of nodes $\mathbb{Z} \subseteq \mathbb{V}$ such that the shortest path between Y and any node in $\mathbb{Z}$ is less than or equal to h. A node W_l in $\mathcal{G}$ is considered a collider on a path p if there exists a subpath $W_k \to W_l \leftarrow W_m$ within p. In this context, we will interchangeably refer to the triple $W_k \to W_l \leftarrow W_m$ and the node W_l as the collider. Furthermore, $W_k \to W_l \leftarrow W_m$ is termed an unshielded collider (UC) if W_k and W_m are not adjacent. A path p is said to be blocked by a set $\mathbb{Z}$ if and only if 1) p contains a non-collider triple (*i.e.*, $W_k \to W_l \to W_m$ or $W_k \leftarrow W_l \leftarrow W_m$ or $W_k \leftarrow W_l \to W_m$) such that the middle node (W_l) is in $\mathbb{Z}$, or 2) p contains a collider (*i.e.*, $W_k \to W_l \leftarrow W_m$) such that $(De(W_l, \mathcal{G}) \cup \{W_l\}) \cap \mathbb{Z} = \emptyset$. Two nodes W_l and W_m are d-separated by $\mathbb{Z}$, denoted $W_l \perp\!\!\!\perp_{\mathcal{G}} W_m \mid \mathbb{Z}$, if and only if all paths between W_l and W_m are blocked by $\mathbb{Z}$ [Pearl, 2000]. The d-separation between W_l and W_m by $\mathbb{Z}$ implies that W_l and W_m are independent conditional on $\mathbb{Z}$, denoted $W_l \perp\!\!\!\perp_{\mathcal{P}} W_m \mid \mathbb{Z}$, in every distribution $\mathcal{P}$ that is compatible with $\mathcal{G}$. We denote by $ds(W_l, W_m, \mathcal{G})$ all subsets that d-separates W_l and W_m in $\mathcal{G}$. Multiple DAGs can encode the same set of d-separations, forming what is known as a Markov equivalence class (MEC). Under Assumption 1, any two DAGs within the same MEC share both the same adjacencies and the same UCs [Verma and Pearl, 1990]. This structural similarity allows every MEC to be uniquely represented by an essential graph, also known as a CPDAG [Chickering, 2002, Andersson et al., 1997, Meek, 1995], denoted $\mathcal{C}$. An essential graph captures all common adjacencies and encodes edge orientations that are invariant across all DAGs in the MEC. Specifically, a directed edge $W_l \to W_m$ in the essential graph implies that this orientation is present in every DAG in the MEC. In contrast, an undirected edge $W_l - W_m$ signals ambiguity—some DAGs contain $W_l \to W_m$ while others contain $W_l \leftarrow W_m$. All structural relations—parents, children, ancestors, descendants, and neighbors—defined in DAGs naturally extend to essential graphs. Thus, we write $Pa(W_l, \mathcal{C})$, $Ch(W_l, \mathcal{C})$, $An(W_l, \mathcal{C})$, $De(W_l, \mathcal{C})$, and $Ne(W_l, \mathcal{C})$ to refer to their counterparts in the essential graph.

Essential graphs are particularly valuable because they can be learned from observational data under Assumption 1 and an additional key assumption, called the faithfulness assumption [Spirtes et al., 2000] which posits that all and only the conditional independencies observed in the data distribution correspond to d-separation relations in the true underlying causal DAG. Under these assumptions, structure learning algorithms can recover the essential graph corresponding

to the true DAG's MEC. One of the most well-known algorithms for this purpose is the PC algorithm [Spirtes et al., 2000]. In a nutshell, the PC algorithm uses conditional independence tests to infer the skeleton of the graph, meaning it removes edges between two nodes W_l and W_m if there exists a set $\mathbb{Z}$ such that $W_l \perp\!\!\!\perp_{\mathcal{P}} W_m \mid \mathbb{Z}$. Then, for each unshielded triple $W_k - W_l - W_m$ in the skeleton, it identifies it as an UC $W_k \rightarrow W_l \leftarrow W_m$ if the middle node W_l was not included in the conditioning set that yielded the independence between W_k and W_m. Finally, the algorithm orients as many other edges as possible using Meek's rules [Meek, 1995].

In this paper, we concentrate on the controlled direct effect (CDE) of treatment variable X on a target variable Y in a non-parametric setting [Pearl, 2000, 2001], denoted as $CDE(x, x', Y)$ and formally expressed as $E(Y \mid do(x), do(pa_{Y \setminus X})) - E(Y \mid do(x'), do(pa_{Y \setminus X}))$, where $pa_{Y \setminus X}$ stands for any realization of the parents of Y, excluding X and $do()$ operator represents an intervention. A $CDE(x, x', Y)$ is said to be identifiable if it can be uniquely computed from the positive observed distribution [Pearl, 2000]. Causal graphs are invaluable for identifying causal effects in general. Specifically, it has been shown that under Assumption 1, the $CDE(x, x', Y)$ is always identifiable from a DAG. It is also identifiable from an essential graph if and only if there is no undirected edge connected to Y [Flanagan, 2020, Theorem 5.4].

3 CHARACTERIZATION OF LOCAL MARKOV EQUIVALENCE

In this section, we introduce notations and results that will be used to define our search space. We start by defining the local Markov equivalence class which contains all DAGs that satisfy a subset of d-separations that will be referred to as *local d-separations*, whose implied conditional independencies will be referred to as *local conditional independencies*.

Definition 1 (Local Markov equivalence class (LMEC))**.** *Consider a DAG $\mathcal{G} = (\mathbb{V}, \mathbb{E})$, a target vertex $Y \in \mathbb{V}$ and an integer h. We define the* local Markov equivalence class *of $\mathcal{G}$ relative to a vertex Y and its h-hop neighborhood, denoted by $LMEC(Y, h, \mathcal{G})$, as the set of graphs such that $\forall \mathcal{G}_i \in LMEC(Y, h, \mathcal{G})$, $\forall D \in Neighborhood(Y, h, \mathcal{G}_i)$, $\forall W \in \mathbb{V} \setminus \{D\}$:*

$$ds(D, W, \mathcal{G}) \quad = \quad ds(D, W, \mathcal{G}_i).$$

The following theorem derives graphical characterization of all DAGs within the same LMEC.

Theorem 1. *Consider a DAG $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ and a vertex of interest $Y \in \mathbb{V}$. We have the following $\forall \mathcal{G}_i, \mathcal{G}_j \in LMEC(Y, h, \mathcal{G})$:*

1. ***Same (h+1)-Neighborhood:***

   $$Neighborhood(Y, h{+}1, \mathcal{G}_i) = Neighborhood(Y, h{+}1, \mathcal{G}_j),$$

2. ***Same local adjacencies:***

   $$\forall D \in Neighborhood(Y, h, \mathcal{G}_i) : Ne(D, \mathcal{G}_i) = Ne(D, \mathcal{G}_j),$$

3. ***Same local UCs:*** *A UC involving the unordered triplet $\{D_1, D_2, A\}$ appears in $\mathcal{G}_i$ with $D_1, D_2 \in Neighborhood(Y, h, \mathcal{G}_i)$ and $A \in Neighborhood(Y, h+1, \mathcal{G}_i)$ if and only if the same UC appears in $\mathcal{G}_j$,*

4. ***Same inactive triples:*** *For any unordered triplet $\{D, A, W\}$ with $D \in Neighborhood(Y, h, \mathcal{G}_i)$, $A \in Ne(D, \mathcal{G}_i)$ and $W \in \mathbb{V} \setminus \{Neighborhood(Y, h, \mathcal{G}_i) \cup Ne(Y, \mathcal{G}_i)\}$:*

   $$D \rightarrow A \leftarrow W \in \mathcal{G}_i \text{ or } W \notin Ne(A, \mathcal{G}_i)$$
   $$\Longleftrightarrow$$
   $$D \rightarrow A \leftarrow W \in \mathcal{G} \text{ or } W \notin Ne(A, \mathcal{G}).$$

In the following, we present a new graphical representation, which we call *local essential graph* (LEG), that represents all graphs in a given LMEC.

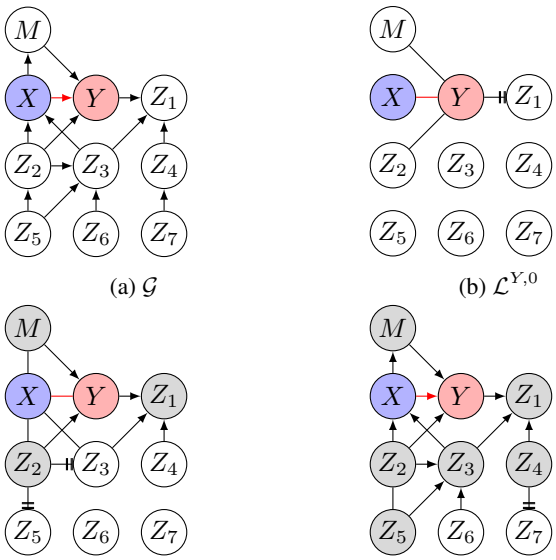
Definition 2 (Local essential graph (LEG))**.** *Let $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ be a directed acyclic graph (DAG), and let $Y \in \mathbb{V}$ be a vertex of interest. The* local essential graph *(LEG) associated with $LMEC(Y, h, \mathcal{G})$, denoted by $\mathcal{L}^{Y,h} = (\mathbb{V}, \mathbb{E}^{Y,h})$, is defined as the partially directed acyclic graph over $\mathbb{V}$ satisfying the following conditions for all nodes $D \in Neighborhood(Y, h, \mathcal{G})$, $A \in Neighborhood(Y, h{+}1, \mathcal{G})$, and $W_i \in \mathbb{V}$:*

1. ***Undirected edge:*** $(A - W)$
   $(A - W) \in \mathbb{E}^{Y,h}$ *if and only if, $\forall \mathcal{G}_i \in LMEC(Y, h, \mathcal{G})$, either $A \rightarrow W$ or $A \leftarrow W$ is present, and $\exists \mathcal{G}_i, \mathcal{G}_j \in LMEC(Y, h, \mathcal{G})$ such that $A \rightarrow W$ appears in $\mathcal{G}_i$ and $A \leftarrow W$ appears in G_j.*

2. ***Arrow edge:*** $(A \rightarrow W)$
   $(A \rightarrow W) \in \mathbb{E}^{Y,h}$ *if and only if, $\forall \mathcal{G}_i \in LMEC(Y, h, \mathcal{G})$, $A \rightarrow W$ appears in $\mathcal{G}_i$.*

3. ***Double-bar edge:*** $(D \dashv A)$
   $(D \dashv A) \in \mathbb{E}^{Y,h}$ *if and only if, $\forall \mathcal{G}_i \in LMEC(Y, h, \mathcal{G})$ and $\forall W \notin \{Neighborhood(Y, h, \mathcal{G}_i) \cup Ne(Y, \mathcal{G}_i)\}$, either $W \notin Ne(A, \mathcal{G}_i)$ or $D \rightarrow A \leftarrow W$.*

Note that from Definition 2, the absence of an edge between two nodes N_1 and N_2 in the LEG has different interpretations depending on the nodes. If at least one of the nodes belongs to $Neighborhood(Y, h, \mathcal{G})$, then the absence of an edge indicates that the nodes are non-adjacent in all DAGs of $LMEC(Y, h, \mathcal{G})$. If neither node is in



(a) $\mathcal{G}$ (b) $\mathcal{L}^{Y,0}$ (c) $\mathcal{L}^{Y,1}$ (d) $\mathcal{L}^{Y,2}$

Figure 1: A DAG $\mathcal{G}$ and the LEGs $\mathcal{L}^{Y,0}$, $\mathcal{L}^{Y,1}$, and $\mathcal{L}^{Y,2}$ around node Y. Red: outcome/target Y; blue: treatment X; grey: h-neighborhood nodes; red arrow: direct effect (M: mediator).

$Neighborhood(Y, h, \mathcal{G})$, then the absence of an edge is non-informative about the edge existence in the LMEC DAGs.

The first two items in Definition 2 closely resemble the conditions used to characterize a full essential graph, with the key difference being that they now apply at the local level (i.e., within a neighborhood). The major distinction compared to the global essential graph characterization lies in the introduction of a new type of orientation in item 3 of Definition 2. Interestingly, this additional orientation enables us—using only local d-separation—to infer that there cannot exist a node outside the neighborhood W that completes a structure of the form $D \to A \to W$, $D \leftarrow A \leftarrow W$, or $D \leftarrow A \to W$. This observation, which follows from item 3 of Theorem 1, will prove useful in Section 5 when addressing a stopping criterion during causal discovery for CDE identification.

The above definition may be relatively dense, and in general graphical concepts are often better understood through visual representation. To facilitate this, we provide an illustration in Figure 1, where Figure 1a depicts a DAG, Figure 1b displays its corresponding LEG relative to Y and its 0-hop neighborhood, while Figure 1c shows its corresponding LEG relative to Y and its 1-hop neighborhood and Figure 1d shows its corresponding LEG relative to Y and its 2-hop neighborhood. Notice that the LEG in Figure 1d contains more edge orientations than the one in Figure 1b, even among nodes that already belong to the 1-hop neighborhood of Y.

A LEG is particularly valuable in the context of local causal discovery, as it compactly encodes rich information about the local d-separations within the graph. Moreover, given any DAG from a specific LMEC, it is possible to reconstruct the corresponding LEG. To demonstrate this, we introduce four orientation rules: three are adaptations of The Meek rules to our local setting, referred to as the LocMeek-Rules, and the fourth is a newly proposed orientation rule, referred to as Loc-Rule.

Definition 3 (Local Rules)**.** *Consider any node denoted by D or D with a subscript (e.g., D_l) to be in $Neighborhood(Y, h, \mathcal{G})$; any node denoted by A to be in $Neighborhood(Y, h+1, \mathcal{G})$. The LocMeek-Rules are defines as follows:*

LocMeek-Rule-1 If $V_1 \to V_2 - V_3$ is unshielded then $V_1 \to V_2 \to V_3$;

LocMeek-Rule-2 If $V_1 \to V_2 \to V_3$ and $V_1 - V_3$ then $V_1 \to V_3$;

LocMeek-Rule-3 If $V_1 - V_2 \to V_3$, $V_1 - V_4 \to V_3$ and $V_1 - V_3$ then $V_1 \to V_3$,

where (V_1, V_2, V_3, V_4) is any ordering of $\{D_1, D_2, D_3, A\}$. Moreover, the LocRule is defined as follows:

Loc-Rule If $D - A$ such that:

(a) $A \notin Neighborhood(Y, h, \mathcal{G})$, and

(b) $\forall W \in \mathbb{V} \setminus \{Neighborhood(Y, h, \mathcal{G}) \cup Ne(D, \mathcal{G})\}, \exists \mathbb{S} \in ds(D, W, \mathcal{G}) : A \notin \mathbb{S}$,

then $D \dashv A$.

Now that all needed orientation rules are introduced, in the following theorem, we formally establish how these rules, along with necessary conditions taken from Theorem 1, can be applied to derive the LEG from any DAG in the LMEC.

Theorem 2. *Let $\mathcal{G}$ be a directed acyclic graph (DAG). The Local essential graph (LEG) $\mathcal{L}^{Y,h}$ associated with $\mathcal{G}$ and defined with respect to a target node Y and hop h can be constructed as follows:*

1. ***Same Neighborhood:***

$$Neighborhood(Y, h{+}1, \mathcal{L}^{Y,h}) = Neighborhood(Y, h{+}1, \mathcal{G}),$$

2. ***Same local adjacencies:***

$$\forall D \in Neighborhood(Y, h, \mathcal{G}), \quad Ne(D, \mathcal{L}^{Y,h}) = Ne(D, \mathcal{G}),$$

3. ***Same local UCs:*** *For all pairs of nodes $D_i, D_j \in Neighborhood(Y, h, \mathcal{G})$ and all $A \in Neighborhood(Y, h+1, \mathcal{G})$, the following UCs are preserved:*

   (a) If $D_i \to D_j \leftarrow A$ in $\mathcal{G}$, then $D_i \to D_j \leftarrow A$ in $\mathcal{L}^{Y,h}$;

(b) *If $D_i \to A \leftarrow D_j$ in $\mathcal{G}$, then $D_i \to A \leftarrow D_j$ in $\mathcal{L}^{Y,h}$.*

4. ***Additional local rules:*** *Apply LocMeek-Rule-1 to LocMeek-Rule-3 and Loc-Rule iteratively until no further rule can be applied.*

We refer back to Figure 1 to illustrate how a LEG can constructed from a DAG. Items 1 and 2 of Theorem 2 are straightforwardly demonstrated in all LEGs: the shaded nodes are always connected to each other in each LEG and to their neighbors exactly as in the DAG in Figure 1a. With Item 3 of Theorem 2, all unshielded colliders (UCs) involving at least two shaded nodes are exactly those present in the DAG shown in Figure 1a. The Loc-Rule of item 4 of Theorem 2 is exemplified in the LEG for the 1-hop neighborhood of Y in Figure 1c: the paths from Z_2 to Z_4, Z_6 and Z_7 containing Z_3 are naturally blocked in the DAG of Figure 1a, hence Z_3 do not appear in $ds(Z_2, Z_4, \mathcal{G})$, $ds(Z_2, Z_6, \mathcal{G})$ and $ds(Z_2, Z_7, \mathcal{G})$. Since Z_4, Z_6 and Z_7 are the only nodes outside $Neighborhood(Y, 1, \mathcal{G}) \cup Ne(Z_1, \mathcal{G})$, Loc-Rule applies, and we orient $Z_2 \dashv Z_3$ in the LEG of Figure 1c. This indicates that if an edge between Z_3 and Z_4, Z_6 or Z_7 exists in the DAG, then $Z_2 \to Z_3 \leftarrow Z_i$ (for $i = 4, 6, 7$) must hold in the DAG. Finally, local Meek rules are applied for example in the LEG of the 2-hop neighborhood of Y in Figure 1d. For example, after orienting $Z_6 \to Z_3$ via the UC involving Z_5, LocMeek-Rule-1 is applied iteratively to infer $Z_3 \to X \to Y$. At this stage ($h = 2$), the full DAG is recovered, which is expected since only one node lies outside the 3-hop neighborhood and $\mathcal{G} = \mathcal{C}$ in this example.

For any given LMEC, the corresponding LEG is unique, as emphasized by this corollary.

Corollary 1. *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two DAGs, and let $\mathcal{L}_1^{Y,h}$ and $\mathcal{L}_2^{Y,h}$ denote their associated LEGs. If $LMEC(Y, h, \mathcal{G}_1) = LMEC(Y, h, \mathcal{G}_2)$, then $\mathcal{L}_1^{Y,h} = \mathcal{L}_2^{Y,h}$.*

4 LOCAL CAUSAL DISCOVERY

In this section, our objective is to recover the LEG from an observed probability distribution, that is, to infer the LEG using conditional independence (CI) tests. To achieve this, we introduce a new algorithm, **LocPC**, which is an adaptation of the classical PC algorithm designed to focus exclusively on local conditional independencies. Given a target variable Y and an integer h representing the size of the local neighborhood of interest, LocPC begins with an undirected graph $\widehat{\mathcal{L}}$ where all nodes are connected to Y and initializes the set of focus variables $\mathbb{D}$ as $\{Y\}$. Next, **LocPC** performs conditional independence tests, inspired by the strategy of the original PC algorithm. For each $D \in \mathbb{D}$ and $W \in Ne(D, \widehat{\mathcal{L}})$, it checks whether $D \perp\!\!\!\perp_{\mathcal{P}} W \mid \mathbb{S}$ for subsets $\mathbb{S} \subseteq \text{Adj}(D, \widehat{\mathcal{L}})$, starting with $|\mathbb{S}| = 0$ and incrementally increasing the size of $\mathbb{S}$. If there exists at least a subset $\mathbb{S}$ such that $D \perp\!\!\!\perp_{\mathcal{P}} W \mid \mathbb{S}$ then the edge $D - W$ is removed and $\mathbb{S}$ is saved in $sepset(D, W, \hat{\mathcal{L}})$. This process continues until either the edge $D - W$ is removed or all possible conditioning sets have been exhausted. This first phase identifies the variables that cannot be rendered conditionally independent of Y, helping to delineate its 1-hop neighborhood. In the second phase, all neighbors of Y in $\widehat{\mathcal{L}}$ are added to $\mathbb{D}$, and undirected edges are drawn between all newly added nodes in $\mathbb{D}$ as well as between these new nodes and nodes outside of $\mathbb{D}$. The procedure from the first phase is then repeated, while avoiding redundant tests (for instance, excluding Y from further conditional independence tests in this phase). The algorithm iteratively expands $\mathbb{D}$ by including all neighbors of nodes in the current $\mathbb{D}$, repeating the process h-times, *i.e.*, until $\mathbb{D}$ contains at least one node whose shortest path to Y is equal to h. After that **LocPC** proceeds to orientation. It first detects all UCs using the same procedure as the PC algorithm. Then it iteratively applies LocMeek-Rule-1-LocMeek-Rule-3 and Loc-Rule until no more rules can be applied. LocMeek-Rule-1–LocMeek-Rule-3 are applied directly by replacing $\mathcal{G}$ with $\hat{\mathcal{L}}$, while Loc-Rule is applied by substituting ds with the separating set $sepset$ identified by the algorithm and also replacing $\mathcal{G}$ with $\hat{\mathcal{L}}$. A pseudo-code of the **LocPC** algorithm is given in Appendix C.

As with full essential graphs, it is generally not possible to recover the LEG purely from observational data without additional assumptions. Discovering essential graphs typically requires both Assumption 1 and the faithfulness assumptions. In the context of LEG recovery using **LocPC**, Assumption 1 remains necessary. However, a key insight is that the full faithfulness assumption is not required. Instead, we introduce a weaker assumption, which we refer to as *local faithfulness*. Due to its relaxed nature, local faithfulness can be more realistic and practical in many applications.

Assumption 2 (Local faithfulness)**.** *Consider a DAG $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ and a vertex of interest $Y \in \mathbb{V}$ with its h-hop neighborhood. We assume that for all $D \in Neighborhood(Y, h, \mathcal{G})$, for all $W \in \mathbb{V} \setminus \{D\}$:*

$$ci(D, W, \mathcal{G}) = ds(D, W, \mathcal{G}),$$

where $ci(D, W, \mathcal{G})$ represents all conditioning sets under which D is conditionally independent of W in any distribution $\mathcal{P}$ compatible with $\mathcal{G}$.

With all necessary assumptions now established, we proceed to demonstrate in the following theorem the correctness of the **LocPC** algorithm.

Theorem 3. *Let $\widehat{\mathcal{L}}$ be the output of the **LocPC** algorithm given a target node Y and an integer h representing the size of the local neighborhood of interest. If Assumptions 1 and 2 are satisfied and given perfect conditional independencies, then $\widehat{\mathcal{L}}$ corresponds to the true LEG $\mathcal{L}^{Y,h}$.*

5 LOCAL CAUSAL DISCOVERY FOR IDENTIFYING CDE

In this section, we aim to recover a portion of the LEG sufficient to determine the identifiability of the causal direct effect $CDE(x, x', Y)$. According to [Flanagan, 2020, Theorem 5.4], identifying this CDE requires verifying whether all edges adjacent to Y are oriented. A straightforward strategy, which we refer to as **naiveLocPC-CDE**, consists of initially applying the **LocPC** algorithm with $h = 0$, and subsequently checking whether all edges incident to Y have been oriented. If so, the $CDE(x, x', Y)$ is identifiable, and the procedure terminates. Otherwise, the process is repeated with $h = 2$, reusing previously obtained information and avoiding redundant conditional independence tests, and continues incrementally in this manner.

For instance, consider the CDE of $X \to Y$ (depicted as a red edge) in Figure 1. This effect is identifiable in $\mathcal{L}^{Y,2}$ because all edges adjacent to Y are oriented within the corresponding LEG.

When $CDE(x, x', Y)$ is not identifiable from the full essential graph, the **naiveLocPC-CDE** algorithm would, in principle, need to repeatedly apply **LocPC** until the entire essential graph is recovered. However, it is possible to anticipate cases where $CDE(x, x', Y)$ is non-identifiable and to identify variables whose exploration would not contribute to identification (i.e., adding them to $\mathbb{D}$ in **LocPC** would be uninformative). To address this, we introduce a stopping criterion that allows us to detect, in advance, when an edge into Y is non-orientable in the essential graph. In such cases, the algorithm can terminate early, as $CDE(x, x', Y)$ would remain non-identifiable even if the full essential graph were recovered.

Definition 4 (Non-orientability criterion in LEGs)**.** *Let $\mathbb{D} \subseteq Neighborhood(Y, h, \mathcal{G})$ be a subset of nodes, and consider the LEG $\mathcal{L}^{Y,h} = (\mathbb{V}, \mathbb{E}^{Y,h})$ with $h \geq 1$. $\mathbb{D}$ satisfies the non-orientability criterion if $\forall D \in \mathbb{D}$:*

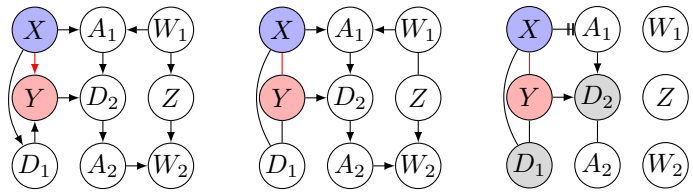
1. *$\nexists A \notin \mathbb{D}$ such that $(D - A) \in \mathbb{E}^{Y,h}$, and*
2. *$|\{A \notin \mathbb{D} : (D \dashv A) \in \mathbb{E}^{Y,h}\}| \leq 1$*

Theorem 4. *Let $\mathbb{D} \subseteq Neighborhood(Y, h, \mathcal{G})$, and let $\mathcal{L}^{Y,h} = (\mathbb{V}, \mathbb{E}^{Y,h})$ denote the LEG with $h \geq 1$. If $\mathbb{D}$ satisfies the non-orientability criterion (Def. 4), then $\forall D_i, D_j \in \mathbb{D}$:*

$$(D_i - D_j) \in \mathbb{E}^{Y,h} \implies \forall k > h : (D_i - D_j) \in \mathbb{E}^{Y,k}.$$

Corollary 2. *Let $\mathbb{D} \subseteq Neighborhood(Y, h, \mathcal{G})$ be a subset of nodes such that $Y \in \mathbb{D}$, and let $\mathcal{L}^{Y,h}$ denote the LEG with $h \geq 1$. If $\mathbb{D}$ satisfies the non-orientability criterion (Def. 4), then $CDE(x, x', Y)$ is not identifiable.*

Corollary 2 shows that full discovery of the essential graph is unnecessary when the CDE is not identifiable (as in **naiveLocPC-CDE**); the process can be halted earlier if the non-orientability criterion holds. Figure 2 illustrates such a case where identification of $CDE(x, x', y)$ is already precluded after obtaining the LEG $\mathcal{L}^{Y,1}$. By choosing $\mathbb{D} = \{Y, X, D_1\}$, we observe that there is no unoriented edge between a node in $\mathbb{D}$ and a node outside $\mathbb{D}$ (condition 1 of Definition 4). Moreover, there exists a unique $\dashv$ edge between the node $X \in \mathbb{D}$ and $A_1 \notin \mathbb{D}$ (condition 2 of Definition 4). Therefore, $\mathbb{D}$ satisfies the non-orientability criterion, and we conclude by Corollary 2 that the $CDE(x, x', Y)$ is not identifiable in $\mathcal{C}$ from the LEG $\mathcal{L}^{Y,1}$—hence, a full discovery of the essential graph is unnecessary. However, note that the set $\mathbb{D}' = \{Y, X, D_1, D_2\}$ does not satisfy the non-orientability criterion since the edge $D_2 - A_2$ is present in $\mathcal{L}^{Y,1}$.



(a) $\mathcal{G}$. (b) $\mathcal{C}$. (c) $\mathcal{L}^{Y,1}$.

Figure 2: A DAG $\mathcal{G}$, its essential graph $\mathcal{C}$ and its associated LEG $\mathcal{L}^{Y,1}$ are given. $\mathbb{D} = \{Y, X, D_1\}$ satisfies the non-orientability criterion from Def 4. Theorem 4 then implies that no undirected edge in $\mathcal{L}^{Y,1}$ between nodes in $\mathbb{D}$ will be oriented in $\mathcal{L}^{Y,k}$ for any $k > 1$.

Building on this idea, we propose the **LocPC-CDE** algorithm. The core idea is to start from the target variable Y and progressively construct the LEG by incrementally increasing the neighborhood hop h. At each step, a local causal discovery procedure is performed, leveraging information from previous iterations to avoid redundant tests and to incorporate already discovered edges. After each expansion, we check whether a set satisfies the criterion defined in Definition 4. If no such set exists, the procedure continues; otherwise, causal discovery can be halted early since the non-identifiability of $CDE(x, x', Y)$ is already established. Other stopping criteria, such as detecting that X is a child of Y or that X is not adjacent to Y implying that $CDE(x, x', y) = 0$, are retained in the algorithm. A pseudo-code of the **LocPC-CDE** algorithm is provided in Appendix B. The **LocPC-CDE** algorithm is obviously more efficient than its naive counterpart, as the discovery process should terminate earlier in cases where the CDE is not identifiable. Theoretically, both algorithms (**LocPC-CDE** and the naive approach) are equivalent in terms of output. However, **LocPC-CDE** offers a computational advantage due to its early stopping criterion in cases where $CDE(x, x', Y)$ is not identifiable. The following theorem establishes the correctness of the **LocPC-CDE** algorithm.

Theorem 5. *If Assumptions 1 and 2 are satisfied and with*

*access to perfect conditional independencies, the **LocPC-CDE** algorithm will correctly detect if $CDE(x, x', Y)$ is identifiable and in case of identifiability it will return the LEG from which $CDE(x, x', Y)$ is identifiable.*

The following result emphasizes that **LocPC-CDE** will identify $CDE(x, x', Y)$ as fast as possible given the information retrieved by the **LocPC** algorithm.

Proposition 1. *Consider Assumptions 1 and 2 are satisfied and we have access to perfect conditional independencies, if **LocPC-CDE** returns that $CDE(x, x', Y)$ is not identifiable, then it was impossible to determine this at any earlier iteration of **LocPC-CDE**.*

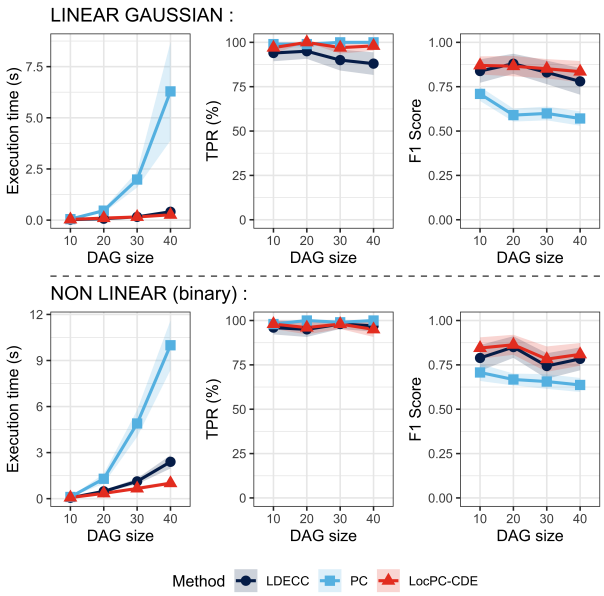
6 EXPERIMENTS

In this section, we present an empirical evaluation based on simulated data to support and validate our theoretical results. We evaluate the **LocPC-CDE** algorithm[1] by comparing it to the PC algorithm [Spirtes et al., 2000] (global discovery), and to LDECC [Gupta et al., 2023], initially designed for local causal discovery of the total effect when targeting the treatment node (X). When targeting the outcome node (Y), LDECC can also be used to identify the direct effect.

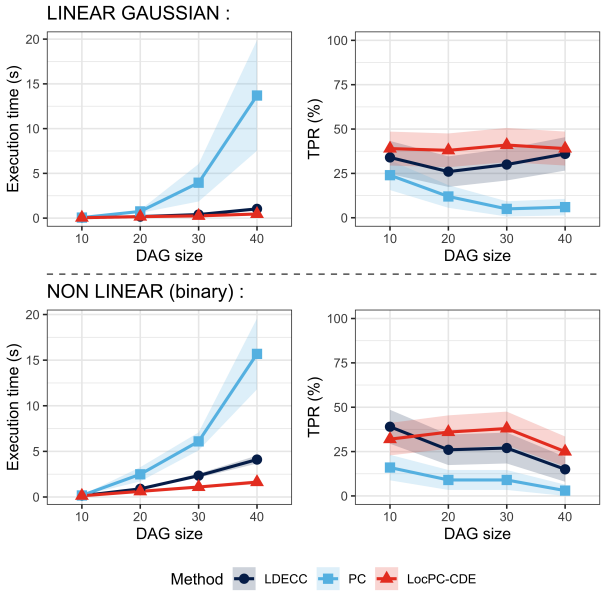
For each setting, experiments are conducted on 100 random Erdős–Rényi DAGs (with constant sparsity as the graph size increases). The DAGs are generated such that the CDE of a variable X on a variable Y is either identifiable or non-identifiable in the associated essential graph. For each DAG of size $|\mathbb{V}| = \{10, 20, 30, 40\}$, a sample of $n = 5000$ observations is generated according to a linear or nonlinear (binary) SCM consistent with the DAG structure. In the linear (Gaussian) case, the Fisher-Z conditional independence test [Fisher, 1921] is used, whereas for the binary case, the G^2 conditional independence test is employed [Tsamardinos et al., 2006] (both tests are performed at significance level $\alpha = 0.05$). For each experiment, we measure: (1) the execution time, (2) the true positive rate (TPR, *i.e.*, the proportion of graphs correctly labeled as having an identifiable CDE when it is identifiable, and non-identifiable otherwise), and (3) for identifiable cases, the F_1 score between the estimated parents of the outcome Y and the true parents in the DAG. Since this corresponds to a valid adjustment set for CDE estimation, an F_1 score of 1 indicates accurate CDE estimation capability. Details about experiments are presented in Appendix C. Results (mean $\pm 1.96 \times$sd) are shown in Figure 3.

In all experiments, **LocPC-CDE** outperforms PC and LDECC in computation time, with the advantage increasing as the number of nodes grows and being more pronounced

[1]In our implementation, we take into account practical considerations, as suggested in Colombo and Maathuis [2014], to make the algorithm order independent.



(a) Identifiable CDE experiment results.



(b) Non-identifiable CDE experiment results.

Figure 3: Empirical performance of **LocPC-CDE** across different graph sizes and SCM settings, compared to global discovery (PC) and state-of-the-art local discovery method (LDECC).

in the nonlinear setting. For identifiable cases, TPR do not differ significantly between **LocPC-CDE** and PC (*i.e.*, local discovery performs as well as global discovery). LDECC performs similarly to **LocPC-CDE** for nonlinear SCMs but worse in linear SCMs. Regarding the F_1 score in identifiable cases, **LocPC-CDE** is comparable or superior to LDECC in both linear and nonlinear settings (though differences are

not significant), and both local methods significantly outperform PC. Finally, in non-identifiable cases, **LocPC-CDE** consistently achieves higher TPR than baselines, and both local methods significantly outperform PC.

7 CONCLUSION

In this work, we addressed the problem of local causal discovery, aiming to identify causal relationships within a graph region sufficient to determine a controlled direct effect (CDE) of interest, without reconstructing the entire causal structure. We characterized a class of graphs sharing local properties with the true graph and introduced *local essential graphs* (LEGs) as a graphical representation of this class. We demonstrated that LEGs can be recovered from observational data under causal sufficiency and a mild local faithfulness assumption, and introduced the constraint-based local discovery algorithm **LocPC** to achieve this. Building on this, we proposed **LocPC-CDE**, an algorithm that leverages local constraints and iterative rule-based inference to efficiently recover the smallest LEG sufficient for identifying the CDE of interest. We proved the algorithm's soundness, completeness, and optimality, and demonstrated its efficiency and effectiveness on simulated data, where it outperformed baselines (global discovery algorithm and existing local discovery algorithm) in runtime while achieving comparable or better CDE identifiability.

This work has several limitations: the LEG characterization is incomplete (a single LEG may correspond to different LMECs); both **LocPC** and **LocPC-CDE** assume causal sufficiency, which may not hold in practice; and CDE identifiability in **LocPC-CDE** depends on identifiability within essential graphs.

References

Holly Andersen. When to expect violations of causal faithfulness and why it matters. *Philosophy of Science*, (5): 672–683, 2013. doi: 10.1086/673937.

Steen A. Andersson, David Madigan, and Michael D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2):505 – 541, 1997. doi: 10.1214/aos/1031833662.

Charles K. Assaad, Imad Ez-Zejjari, and Lei Zan. Root cause identification for collective anomalies in time series given an acyclic summary causal graph with loops. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 8395–8404. PMLR, 25–27 Apr 2023.

Ali Aït-Bachir, Charles K. Assaad, Christophe de Bignicourt, Emilie Devijver, Simon Ferreira, Eric Gaussier, Hosein Mohanna, and Lei Zan. Case studies of causal discovery from it monitoring time series, 2023. The History and Development of Search Methods for Causal Structure Workshop at the 39th Conference on Uncertainty in Artificial Intelligence.

David Maxwell Chickering. Learning equivalence classes of bayesian-network structures. *J. Mach. Learn. Res.*, 2: 445–498, March 2002. ISSN 1532-4435. doi: 10.1162/153244302760200696.

Diego Colombo and Marloes H. Maathuis. Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782, January 2014. ISSN 1532-4435.

Ronald Aylmer Sir Fisher. On the probable error of a coefficient of correlation deduced from a small sample. *Metron*, 1:1–32, 1 1921.

Emily Flanagan. *Identification and Estimation of Direct Causal Effects*. PhD thesis, 2020.

Tian Gao and Qiang Ji. Local causal discovery of direct causes and effects. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Tomas Geffner, Javier Antoran, Adam Foster, Wenbo Gong, Chao Ma, Emre Kiciman, Amit Sharma, Angus Lamb, Martin Kukla, Nick Pawlowski, Agrin Hilmkil, Joel Jennings, Meyer Scetbon, Miltiadis Allamanis, and Cheng Zhang. Deep end-to-end causal inference. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Shantanu Gupta, David Childers, and Zachary Chase Lipton. Local causal discovery for estimating causal effects. In *2nd Conference on Causal Learning and Reasoning*, 2023.

Antti Hyttinen, Frederick Eberhardt, and Matti Järvisalo. Do-calculus when the true graph is unknown. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI'15, page 395–404, Arlington, Virginia, USA, 2015. AUAI Press. ISBN 9780996643108.

Marloes Maathuis, Markus Kalisch, and Peter Bühlmann. Estimating high-dimensional intervention effects from observation data. *The Ann. Stat.*, 37, 10 2008. doi: 10.1214/09-AOS685.

Daniel Malinsky and Peter Spirtes. Estimating causal effects with ancestral graph Markov models. In Alessandro Antonucci, Giorgio Corani, and Cassio Polpo Campos, editors, *Proceedings of the Eighth International Conference*

on Probabilistic Graphical Models, volume 52 of *Proceedings of Machine Learning Research*, pages 299–309, Lugano, Switzerland, 06–09 Sep 2016. PMLR.

Christopher Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, page 403–410, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603859.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-77362-8.

Judea Pearl. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 411–420, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608001.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT press, 2nd edition, 2000.

Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1): 31–78, 3 2006. doi: 10.1007/s10994-006-6889-7.

Caroline Uhler, Garvesh Raskutti, Peter Buhlmann, and Bin Yu. Geometry of the faithfulness assumption in causal inference. *Annals of Statistics*, 41:436–463, 2012.

Tyler Vanderweele. Controlled direct and mediated effects: Definition, identification and bounds. *Scandinavian Journal of Statistics*, 38:551 – 563, 12 2010. doi: 10.1111/j.1467-9469.2010.00722.x.

Stijn Vansteelandt. Estimating direct effects in cohort and case–control studies. *Epidemiology*, 20(6):851–860, 2009. ISSN 10443983.

Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '90, page 255–270, USA, 1990. Elsevier Science Inc. ISBN 0444892648.

Supplementary Material

Timothée Loranchet[1] **Charles K. Assaad**[1]

[1]Sorbonne Université, INSERM,, Institut Pierre Louis d'Epidémiologie, et de Santé Publique,, F75012, Paris, France

A PROOFS

A.1 PROOF OF THEOREM 1

We restate Theorem 1 and then we prove it.

Theorem 1. *Consider a DAG $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ and a vertex of interest $Y \in \mathbb{V}$. We have the following $\forall \mathcal{G}_i, \mathcal{G}_j \in LMEC(Y, h, \mathcal{G})$:*

1. ***Same (h+1)-Neighborhood:***

$$Neighborhood(Y, h+1, \mathcal{G}_i) = Neighborhood(Y, h+1, \mathcal{G}_j),$$

2. ***Same local adjacencies:***

$$\forall D \in Neighborhood(Y, h, \mathcal{G}_i) : Ne(D, \mathcal{G}_i) = Ne(D, \mathcal{G}_j),$$

3. ***Same local UCs:*** *A UC involving the unordered triplet $\{D_1, D_2, A\}$ appears in $\mathcal{G}_i$ with $D_1, D_2 \in Neighborhood(Y, h, \mathcal{G}_i)$ and $A \in Neighborhood(Y, h+1, \mathcal{G}_i)$ if and only if the same UC appears in $\mathcal{G}_j$,*
4. ***Same inactive triples:*** *For any unordered triplet $\{D, A, W\}$ with $D \in Neighborhood(Y, h, \mathcal{G}_i)$, $A \in Ne(D, \mathcal{G}_i)$ and $W \in \mathbb{V} \setminus \{Neighborhood(Y, h, \mathcal{G}_i) \cup Ne(Y, \mathcal{G}_i)\}$:*

$$D \to A \leftarrow W \in \mathcal{G}_i \text{ or } W \notin Ne(A, \mathcal{G}_i)$$
$$\Longleftrightarrow$$
$$D \to A \leftarrow W \in \mathcal{G} \text{ or } W \notin Ne(A, \mathcal{G}).$$

Proof. Let $\mathcal{G}_i, \mathcal{G}_j \in LMEC(Y, h, \mathcal{G})$. We prove every item of the theorem, starting by proving 1 and 2 simultaneously:

Items 1 and 2: Proven by induction on h.

(*Base case* Consider $h = 0$. We show that $Neighborhood(Y, 1, \mathcal{G}_i) = Neighborhood(Y, 1, \mathcal{G}_j)$, i.e., $Ne(Y, \mathcal{G}_i) = Ne(Y, \mathcal{G}_j)$. Assume for contradiction that $N \in Ne(Y, \mathcal{G}_i)$ but $N \notin Ne(Y, \mathcal{G}_j)$. Since $N \in Ne(Y, \mathcal{G}_i)$, no set[1] d-separates Y and N in $\mathcal{G}_i$. However, since $N \notin Ne(Y, \mathcal{G}_j)$, there exists a set $\mathbb{S} \in ds(Y, N, \mathcal{G}_j)$ that d-separates Y and N in $\mathcal{G}_j$. This contradicts the assumption that $\mathcal{G}_i$ and $\mathcal{G}_j$ belong to the same LMEC (i.e., share the same d-separation structure). Hence, $Neighborhood(Y, 1, \mathcal{G}_i) = Neighborhood(Y, 1, \mathcal{G}_j)$.

(*Induction step*) Assume the induction hypothesis $\mathcal{H} : Neighborhood(Y, h, \mathcal{G}_i) = Neighborhood(Y, h, \mathcal{G}_j)$ holds for some $h \geq 0$. We prove it holds for $h+1$. Let $D \in Neighborhood(Y, h, \mathcal{G}_i) = Neighborhood(Y, h, \mathcal{G}_j)$ (by $\mathcal{H}$). We show $Ne(D, \mathcal{G}_i) = Ne(D, \mathcal{G}_j)$. Suppose, for contradiction, that $A \in Ne(D, \mathcal{G}_i)$ but $A \notin Ne(D, \mathcal{G}_j)$. Then no

[1]Note that when we say there is no separating set for A and B in $\mathcal{G}$, we mean that $ds(A, B, \mathcal{G})$ is undefined, not that $ds(A, B, \mathcal{G}) = \emptyset$. If A and B are unconditionally d-separated, then $ds(A, B, \mathcal{G}) = \emptyset$, and we say they are separated by the empty set.

Accepted for the 1st Workshop on Causal Abstractions and Representations (CAR) at UAI 2025 (CAR 2025).

set S d-separates D and A in $\mathcal{G}_i$, but some set S d-separates them in $\mathcal{G}_j$. Again, this contradicts the assumption that $\mathcal{G}_i$ and $\mathcal{G}_j$ are in the same LMEC. Therefore, for all $D \in Neighborhood(Y, h, \mathcal{G}_i)$, $Ne(D, \mathcal{G}_i) = Ne(D, \mathcal{G}_j)$, implying $Neighborhood(Y, h+1, \mathcal{G}_i) = Neighborhood(Y, h+1, \mathcal{G}_j)$.

This completes the induction and proves both first items of the theorem. From this point onward, whenever we write $Neighborhood(Y, h+1, \mathcal{G}_i)$, it is understood to denote the same set as $Neighborhood(Y, h+1, \mathcal{G}_j)$.

Item 3: Let $D_k, D_l \in Neighborhood(Y, h, \mathcal{G}_i)$ and $A \in Neighborhood(Y, h+1, \mathcal{G}_i)$. Suppose, for contradiction, that the unshielded collider (UC) $D_k \to A \leftarrow D_l$ exists in $\mathcal{G}_i$ but not in $\mathcal{G}_j$. By item 2, the adjacencies $A \in Ne(D_k, \mathcal{G}_j)$, $A \in Ne(D_l, \mathcal{G}_j)$, and $D_k \notin Ne(D_l, \mathcal{G}_j)$ also hold in $\mathcal{G}_j$. Since the triplet (D_k, A, D_l) is not a collider in $\mathcal{G}_j$, it must be a chain ($D_k \to A \to D_l$ or $D_l \to A \to D_k$) or fork ($D_k \leftarrow A \to D_l$), implying $A \in \mathbb{S}$ for all $\mathbb{S} \in ds(D_k, D_l, \mathcal{G}_j)$. In contrast, since $D_k \to A \leftarrow D_l$ in $\mathcal{G}_i$, A is a collider and thus $A \notin \mathbb{S}$ for all $\mathbb{S} \in ds(D_k, D_l, \mathcal{G}_i)$. This contradicts the fact that $\mathcal{G}_i, \mathcal{G}_j \in LMEC(Y, h, \mathcal{G})$.

Hence, $D_k \to A \leftarrow D_l$ must also exist in $\mathcal{G}_j$. The argument similarly applies to UCs of the form $D_k \to D_l \leftarrow A$.

Item 4: We show that if $D \to A \leftarrow W \in \mathcal{G}_i$ or $W \notin Ne(A, \mathcal{G}_i)$, then it must also hold that $D \to A \leftarrow W \in \mathcal{G}_j$ or $W \notin Ne(A, \mathcal{G}_j)$. Equivalently, if the triple (D, A, W) is neither a chain nor a fork in $\mathcal{G}_i$, it must also be neither a chain nor a fork in $\mathcal{G}_j$.

We proceed by contradiction. Suppose $D \to A \leftarrow W \in \mathcal{G}_i$ or $W \notin Ne(A, \mathcal{G}_i)$, but that (D, A, W) forms a chain or a fork in $\mathcal{G}_j$. Then the path $\langle D, A, W \rangle$ must be blocked to d-separate D and W in $\mathcal{G}_j$, which requires conditioning on A. Hence, for all $\mathbb{S} \in ds(D, W, \mathcal{G}_j)$, it must be that $A \in \mathbb{S}$. We show that this contradicts the assumption that $\mathcal{G}_i$ and $\mathcal{G}_j$ belong to the same LMEC by demonstrating that there always exists $\mathbb{S} \in ds(D, W, \mathcal{G}_i)$ such that $A \notin \mathbb{S}$.

- If $D \to A \leftarrow W \in \mathcal{G}_i$, then conditioning on A activates the collider path between D and W, and thus A cannot belong to any separating set: $\forall \mathbb{S} \in ds(D, W, \mathcal{G}_i),\ A \notin \mathbb{S}$. This directly contradicts the requirement in $\mathcal{G}_j$ that all separating sets contain A.
- If $W \notin Ne(A, \mathcal{G}_i)$, we consider two subcases:
  (a) If no active path between D and W contains A in $\mathcal{G}_i$, then A is irrelevant for d-separation, and so there exists a minimal separating set $\mathbb{S} \in ds(D, W, \mathcal{G}_i)$ such that $A \notin \mathbb{S}$, contradicting the assumption.
  (b) If there exists at least one active path from D to W passing through A, we distinguish two further cases:
    i. If $D \notin De(W, \mathcal{G}_i)$, then by standard properties of d-separation, $Pa(W, \mathcal{G}_i)$ separates D from W. Since $W \notin Ne(A, \mathcal{G}_i)$, it follows that $A \notin Pa(W, \mathcal{G}_i)$, and thus $A \notin \mathbb{S}$ for some $\mathbb{S} \in ds(D, W, \mathcal{G}_i)$ — again a contradiction.
    ii. If $D \in De(W, \mathcal{G}_i)$, then any path from W to D through A must be directed: $W \to \cdots \to K \to A \to D$. These paths are blocked by K, and conditioning on K does not activate any other path since K is neither a collider nor a descendant of one — otherwise it would form a cycle. Thus, the path is blocked without conditioning on A, and again there exists $\mathbb{S} \in ds(D, W, \mathcal{G}_i)$ such that $A \notin \mathbb{S}$.

In all cases, we reach a contradiction. Therefore, it must be that $D \to A \leftarrow W \in \mathcal{G}_j$ or $W \notin Ne(A, \mathcal{G}_j)$. The reverse implication follows by symmetry, exchanging the roles of $\mathcal{G}_i$ and $\mathcal{G}_j$.

□

A.2 PROOF OF THEOREM 2

We proceed to prove the theorem after having restated it.

Theorem 2. *Let $\mathcal{G}$ be a directed acyclic graph (DAG). The Local essential graph (LEG) $\mathcal{L}^{Y,h}$ associated with $\mathcal{G}$ and defined with respect to a target node Y and hop h can be constructed as follows:*

1. ***Same Neighborhood:***

$$Neighborhood(Y, h+1, \mathcal{L}^{Y,h}) = Neighborhood(Y, h+1, \mathcal{G}),$$

2. ***Same local adjacencies:***

$$\forall D \in Neighborhood(Y, h, \mathcal{G}), \quad Ne(D, \mathcal{L}^{Y,h}) = Ne(D, \mathcal{G}),$$

3. ***Same local UCs:*** *For all pairs of nodes $D_i, D_j \in Neighborhood(Y, h, \mathcal{G})$ and all $A \in Neighborhood(Y, h+1, \mathcal{G})$, the following UCs are preserved:*
   (a) *If $D_i \to D_j \leftarrow A$ in $\mathcal{G}$, then $D_i \to D_j \leftarrow A$ in $\mathcal{L}^{Y,h}$;*
   (b) *If $D_i \to A \leftarrow D_j$ in $\mathcal{G}$, then $D_i \to A \leftarrow D_j$ in $\mathcal{L}^{Y,h}$.*
4. ***Additional local rules:*** *Apply LocMeek-Rule-1 to LocMeek-Rule-3 and Loc-Rule iteratively until no further rule can be applied.*

Proof. Items 1, 2, and 3 follow directly from Items 1, 2, and 3 of Theorem 1, which ensures that the edges of type $(-)$ and $(\to)$ introduced through these items are consistent with Definition 2.

LocMeek-Rule-1, LocMeek-Rule-2, and LocMeek-Rule-3 guarantee that no new unshielded collider common to all DAGs in the LMEC (according to Theorem 1) is introduced, and that no cycles are created. Therefore, the directed edges added by these rules also satisfy Definition 2.

We now prove the soundness of the local rule Loc-Rule. Let $\mathcal{G}_i \in LMEC(Y, h, \mathcal{G})$. Suppose that there exists a node $W_0 \in \mathbb{V} \setminus \{Neighborhood(Y, h, \mathcal{G}_i) \cup Ne(D, \mathcal{G}_i)\}$ such that the triple (D, A, W_0) forms a chain or a fork in $\mathcal{G}_i$. Then necessarily, for all separating sets $\mathbb{S} \in ds(D, W_0, \mathcal{G}_i)$, it must hold that $A \in \mathbb{S}$. Thus, if the condition of rule Loc-Rule is satisfied for all $W \notin Neighborhood(Y, h, \mathcal{G}_i)$, it implies that, for all such W, either $W \notin Ne(A, \mathcal{G}_i)$ or $D \to A \leftarrow W$ is a collider in $\mathcal{G}_i$. By Item 4 of Theorem 1, this property holds in all DAGs of the LMEC. Therefore, introducing the edge $D \dashv A$ is consistent with Definition 2.

□

Corollary 1. *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two DAGs, and let $\mathcal{L}_1^{Y,h}$ and $\mathcal{L}_2^{Y,h}$ denote their associated LEGs. If $LMEC(Y, h, \mathcal{G}_1) = LMEC(Y, h, \mathcal{G}_2)$, then $\mathcal{L}_1^{Y,h} = \mathcal{L}_2^{Y,h}$.*

Proof. If $\mathcal{G}_1$ and $\mathcal{G}_2$ belong to the same LMEC, then Theorem 1 ensures that all the structural features required for constructing the LEG, as defined in Theorem 2, are shared by both DAGs. Consequently, they have the same LEG. □

A.3 PROOF OF THEOREM 3

Theorem 3. *Let $\widehat{\mathcal{L}}$ be the output of the **LocPC** algorithm given a target node Y and an integer h representing the size of the local neighborhood of interest. If Assumptions 1 and 2 are satisfied and given perfect conditional independencies, then $\widehat{\mathcal{L}}$ corresponds to the true LEG $\mathcal{L}^{Y,h}$.*

Proof. Let $\mathcal{G}$ denote the true underlying graph, $\mathcal{L}^{Y,h}$ the associated LEG, and $\widehat{\mathcal{L}}$ the output of **LocPC**. We show that under Assumptions 1 and 2, and assuming access to a perfect conditional independence (CI) oracle, the output $\widehat{\mathcal{L}}$ satisfies all items of Theorem 2.

Items 1, 2: We first show that **LocPC** correctly recovers the neighborhood of Y: $Ne(Y, \widehat{\mathcal{L}}) = Ne(Y, \mathcal{L}^{Y,h})$. Initially, all nodes are adjacent to Y. For each node $N \neq Y$, **LocPC** tests whether there exists a conditioning set $\mathbb{S}$ such that Y and N are independent conditionally on $\mathbb{S}$. If $N \notin Ne(Y, \mathcal{G})$, then there must exists $\mathbb{S} \in ds(Y, N, \mathcal{G})$, implying $\mathbb{S} \in ci(Y, N, \mathcal{G})$. The CI oracle allows **LocPC** to identify such $\mathbb{S}$ and remove the edge between Y and N. Conversely, if $N \in Ne(Y, \mathcal{G})$, then no such $\mathbb{S}$ exists under Assumption 2, and **LocPC** retains the edge. Therefore, $Ne(Y, \widehat{\mathcal{L}}) = Ne(Y, \mathcal{G})$, and by Theorem 2, this equals $Ne(Y, \mathcal{L}^{Y,h})$. Extending this argument recursively to the h-hop neighborhood yields:

$$Neighborhood(Y, h+1, \widehat{\mathcal{L}}) = Neighborhood(Y, h+1, \mathcal{L}^{Y,h}),$$

and for all $D \in Neighborhood(Y, h, \widehat{\mathcal{L}})$:

$$Ne(D, \widehat{\mathcal{L}}) = Ne(D, \mathcal{L}^{Y,h}).$$

Item 3: **LocPC** examines all unshielded triples $\{D_i, D_j, A\}$ where $D_i, D_j \in Neighborhood(Y, h, \widehat{\mathcal{L}})$, $A \in Neighborhood(Y, h+1, \widehat{\mathcal{L}})$, and either $D_i - D_j - A$ or $D_i - A - D_j$ exists in $\widehat{\mathcal{L}}$. Without loss of generality, assume the latter. From Items 1 and 2, we know $D_i, D_j \in Ne(A, \mathcal{L}^{Y,h})$ and $D_i \notin Ne(D_j, \mathcal{L}^{Y,h})$. Suppose this triple

does not form a collider in $\mathcal{G}$. Then for all $\mathbb{S} \in ds(D_i, D_j, \mathcal{G})$, we must have $A \in \mathbb{S}$. Under Assumption 2, this implies $A \in \mathbb{C}$ for all $\mathbb{C} \in ci(D_i, D_j, \mathcal{G})$. Thus, if the CI oracle returns a separating set that excludes A, the triple must be a collider. Conversely, if A is always included, **LocPC** does not falsely orient the triple as a collider. Hence, **LocPC** identifies exactly the unshielded colliders (UCs) present in $\mathcal{G}$, which, by Theorem 2, are also present in $\mathcal{L}^{Y,h}$. Therefore, Item 3 is satisfied.

Item 4: Rules LocMeek-Rule-1, LocMeek-Rule-2, and LocMeek-Rule-3 are applied in **LocPC** exactly as defined in Definition 3. Therefore, Item 4 of Theorem 2, concerning Meek's rules, is necessarily satisfied by the output $\widehat{\mathcal{L}}$.

To show that Loc-Rule is also correctly applied, we focus on a subtle point: the definition of Rule Loc-Rule requires that for all $W \in \mathbb{V} \setminus \{Neighborhood(Y, h, \mathcal{G}) \cup Ne(D, \mathcal{G})\}$, there *exists* a separating set $\mathbb{S} \in ds(D, W, \mathcal{G})$ such that $A \notin \mathbb{S}$. However, the **LocPC** algorithm only finds *one* separating set per d-separated pair and not all of them. We must therefore verify that this is sufficient to identify all node pairs (D, A) satisfying the local rule. This follows by rewriting the condition using contraposition:

$$\forall W \in \mathbb{V} \setminus \{Neighborhood(Y, h, \mathcal{G}) \cup Ne(D, \mathcal{G})\},\ \exists \mathbb{S} \in ds(D, W, \mathcal{G}) : A \notin \mathbb{S}$$

$$\iff$$

$$\nexists W \in \mathbb{V} \setminus \{Neighborhood(Y, h, \mathcal{G}) \cup Ne(D, \mathcal{G})\} : \forall \mathbb{S} \in ds(D, W, \mathcal{G}), A \in \mathbb{S}$$

The right-hand side shows that it suffices for **LocPC** to find a single separating set between each D and candidate $W \in \mathbb{V} \setminus \{Neighborhood(Y, h, \widehat{\mathcal{L}}) \cup Ne(D, \widehat{\mathcal{L}})$. If such a set contains A for any W, then Rule Loc-Rule is not satisfied. Conversely, if no such W is found, then the edge $D \dashv\!\!\!\mid A$ can be correctly added. Therefore, Rule Loc-Rule is soundly applied, and Item 4 of Theorem 2 is fully satisfied by $\widehat{\mathcal{L}}$.

The **LocPC** algorithm thus recovers all the elements of Theorem 2 characterizing the LEG. Moreover, **LocPC** performs no additional operations beyond identifying these elements. Consequently, the graph returned by **LocPC** necessarily corresponds to the LEG of the LMEC of the underlying graph. □

A.4 PROOF OF THEOREM 4

Theorem 4. *Let $\mathbb{D} \subseteq Neighborhood(Y, h, \mathcal{G})$, and let $\mathcal{L}^{Y,h} = (\mathbb{V}, \mathbb{E}^{Y,h})$ denote the LEG with $h \geq 1$. If $\mathbb{D}$ satisfies the non-orientability criterion (Def. 4), then $\forall D_i, D_j \in \mathbb{D}$:*

$$(D_i - D_j) \in \mathbb{E}^{Y,h} \implies \forall k > h : (D_i - D_j) \in \mathbb{E}^{Y,k}.$$

Proof. After constructing the LEG $\mathcal{L}^{Y,h}$ for a given hop h, the only way the undirected edge $D_i - D_j$ could become oriented in a LEG $\mathcal{L}^{Y,k}$ with $k > h$ is through the propagation of orientations via Meek's rules. These are the only mechanisms capable of orienting edges already present at hop h.

To prevent such propagation, we first require that there are no undirected edges between nodes in $\mathbb{D}$ and nodes outside $\mathbb{D}$; such connections could serve as pathways for orientation propagation under Meek's rules.

Furthermore, for each node $D \in \mathbb{D}$, if there exists at most one node $A \notin \mathbb{D}$ such that $D \dashv\!\!\!\mid A$, then we ensure the following:[2]

- Either no new neighbor of A will be discovered (i.e., one that was not already included at hop h), in which case the edge $D - A$ cannot be oriented and no propagation can occur;
- Or, if new neighbors of A are discovered (which are non-neighbors D), then $D \to A$ will be oriented. This orientation, however, is incompatible with all of Meek's rules for propagating directions back into $\mathbb{D}$.

Therefore, if the non-orientability conditions (Def. 4) are satisfied, no orientation can reach the edge $D_i - D_j$ in $\mathcal{L}^{Y,k}$ for any $k > h$. As a result, $D_i - D_j$ remains undirected in all subsequent LEGs. □

We now turn to the proof of the corollary associated with Theorem 4.

[2] If multiple such nodes A exist, they could form an unshielded collider with D in some LEG of hop $k > h$, possibly leading to orientations like $A \to D$, and thus allowing orientations to propagate within $\mathbb{D}$ via Meek's rules.

Corollary 2. *Let $\mathbb{D} \subseteq Neighborhood(Y, h, \mathcal{G})$ be a subset of nodes such that $Y \in \mathbb{D}$, and let $\mathcal{L}^{Y,h}$ denote the LEG with $h \geq 1$. If $\mathbb{D}$ satisfies the non-orientability criterion (Def. 4), then $CDE(x, x', Y)$ is not identifiable.*

Proof. First, note that the essential graph $\mathcal{C} = \mathcal{L}^{Y,k_{\max}}$ where $k_{\max}$ is the length of the longest path from Y to any other node in the graph. By applying Theorem 4 to the undirected edges adjacent to Y that are included in the subset $\mathbb{D}$, we deduce that these edges remain undirected in the essential graph $\mathcal{C}$. Then, by Theorem 5.4 of [Flanagan, 2020], it follows that the CDEs on Y are not identifiable. □

A.5 PROOF OF THEOREM 5

Theorem 5. *If Assumptions 1 and 2 are satisfied and with access to perfect conditional independencies, the **LocPC-CDE** algorithm will correctly detect if $CDE(x, x', Y)$ is identifiable and in case of identifiability it will return the LEG from which $CDE(x, x', Y)$ is identifiable.*

Proof. Assume that the CDE is identifiable. This means that in the essential graph $\mathcal{C}$, all nodes adjacent to Y are oriented. Under Assumptions 1 and 2, and assuming access to a perfect conditional independence oracle, it follows from Theorem 3 that at each iteration over h, the locally estimated essential graph (LEG) $\widehat{\mathcal{L}}$ discovered by **LocPC** is correct. Moreover, if the CDE is identifiable, then by Corollary 2, the non-orientability criterion (Def. 4) will never be satisfied. Consequently, the local discovery will proceed until all edges adjacent to Y are oriented. This will eventually occur since, in the worst case, the discovered graph $\widehat{\mathcal{L}}$ becomes equal to the essential graph $\mathcal{C}$ if all relevant nodes are included. Therefore, there exists an iteration in which all edges adjacent to Y are oriented, and the algorithm will conclude that the CDEs with respect to Y are identifiable. The returned LEG necessarily has all of Y's adjacents oriented, which is sufficient for estimating the CDE.

Assume that the CDE is not identifiable. This implies that in the essential graph $\mathcal{C}$, the adjacency of Y is not fully oriented. Under Assumptions 1 and 2, and given a conditional independence oracle, the LEG $\widehat{\mathcal{L}}$ discovered at each iteration h is correct by Theorem 3. According to Corollary 2, if the non-orientability criterion (Def. 4) is satisfied—which acts as a stopping condition—then the CDE is not identifiable. If the stopping condition is never triggered, the algorithm continues the local discovery process until it recovers the full essential graph $\mathcal{C}$. Thus, in all cases, when the algorithm terminates and finds that Y's adjacency is not fully oriented, we are guaranteed that the CDE is indeed not identifiable. □

Proposition 1. *Consider Assumptions 1 and 2 are satisfied and we have access to perfect conditional independencies, if **LocPC-CDE** returns that $CDE(x, x', Y)$ is not identifiable, then it was impossible to determine this at any earlier iteration of **LocPC-CDE**.*

Proof. Assume that the CDE is not identifiable and that the algorithm declares non-identifiability at iteration $\widehat{h}$. We now show that it was not possible to conclude non-identifiability at iteration $\widehat{h} - 1$. If the algorithm did not terminate at iteration $\widehat{h} - 1$, this implies that the non-orientability condition (Definition 4) was not satisfied at that stage. However, as demonstrated in the proof of Theorem 4, when the non-orientability condition is not satisfied, it remains possible that an edge in the discovery set $\mathbb{D}$ becomes oriented as the size of the discovery increases. Therefore, it follows directly that without continuing the discovery at iteration $\widehat{h}$, it was not possible to be certain that the CDE was not identifiable. □

B PSEUDO-CODES

B.1 LOCPC

We present here the main steps of Algorithm 1, which describes the **LocPC** procedure for learning a local essential graph (LEG) in the neighborhood of a target node Y, up to a given hop h.

The algorithm begins by initializing essential data structures. Specifically, line 1 initializes the estimated LEG $\widehat{\mathcal{L}}$ as a fully disconnected graph over the set of observed variables $\mathbb{V}$. Line 2 initializes the exploration frontier $\mathbb{D}$ with the target node Y, and line 3 sets the list of visited nodes to initially contain Y. The hop counter k, used to track the depth of exploration in the first degree, is initialized to 0 in line 4.

The local skeleton discovery phase is implemented in the loop spanning lines 5–26. As long as $k \leq h$, the algorithm expands the neighborhood of the current frontier. At each iteration, line 6 resets the container $\mathbb{D}_{\text{new}}$ for nodes discovered in the current

hop. Then, lines 7–10 ensure all nodes $B \in \mathbb{V} \setminus$ visited are connected to each current node $D \in \mathbb{D}$, temporarily introducing edges.

Lines 11–24 implement the constraint-based local structure learning phase, analogous to the PC algorithm [Spirtes et al., 2000], but restricted to the current local frontier. The variable s denotes the size of conditioning sets and is initialized in line 11. The algorithm proceeds iteratively, increasing s until no separating sets of size s are found. At each iteration, the adjacency set of each $D \in \mathbb{D}$ is cached (line 14) to ensure order independence [Colombo and Maathuis, 2014]. Then, for each neighbor B of D (line 18), if there are at least s other adjacent nodes, the algorithm considers all subsets $\mathbb{S} \subseteq \text{adj}(D) \setminus \{B\}$ of size s. If a subset $\mathbb{S}$ is found such that $D \perp\!\!\!\perp B \mid \mathbb{S}$ (line 22), it is recorded as a separating set (lines 23-24), the corresponding edge is removed from $\widehat{\mathcal{L}}$ (line 25), and further testing for this pair is halted.

After all candidate edges have been tested for a given s, the newly discovered neighbors of each D are added to $\mathbb{D}_{\text{new}}$ (line 27). This process repeats with increasing s until no further conditional independencies can be established, at which point $\mathbb{D}$ is updated for the next hop (line 29), and the hop counter k is incremented (line 30).

Upon completion of the local skeleton discovery, the algorithm proceeds to edge orientation. Lines 31-33 orient unshielded colliders using standard criteria: for each unshielded triple $A - B - C$, if $B \notin \text{Sepset}(A, C)$ and only one of the nodes belongs to the $(h+1)$-hop neighborhood of Y, then $A \to B \leftarrow C$ is oriented.

Subsequently, line 34 applies the three local Meek rules inspired by [Meek, 1995]—namely, Rules LocMeek-Rule-1, LocMeek-Rule-2, and LocMeek-Rule-3—to further orient edges where possible.

Finally, lines 35-43 implement a local orientation rule (Rule Loc-Rule) that aims to distinguish between within-neighborhood and out-of-neighborhood connections. For each candidate edge $D - A$, where D belongs to the h-hop neighborhood and A does not, the algorithm verifies that A is not present in any separating set $\text{Sepset}(D, W)$ for nodes W lying outside the neighborhood. If this condition is satisfied for all such W, an edge $D \dashv A$ is added (line 43), indicating the possibility of an unmeasured confounder or non-identifiable causal relationship.

The algorithm concludes by returning the estimated LEG $\widehat{\mathcal{L}}$, the map of separating sets $\mathcal{S}$, and the list of visited nodes, which can be reused in subsequent procedures such as **LocPC-CDE** (see 2) to avoid redundant computations.

B.2 LOCPC-CDE

First, we define the set of non-arrow neighbors of a node Y in a graph $\mathcal{G}_0 = (\mathbb{V}_0, \mathbb{E}_0)$ as

$$Ne_{\text{nar}}(Y, \mathcal{G}_0) = Ne(Y, \mathcal{G}_0) \setminus \{V \in \mathbb{V}_0 \mid (Y \to V) \in \mathbb{E}_0 \text{ or } (Y \leftarrow V) \in \mathbb{E}_0\}.$$

We now detail the main steps of Algorithm 2. The algorithm first calls **LocPC** to discover the 0-hop LEG around Y (line 1). At this stage, discovered separating sets and visited nodes are stored, with the visited set initialized to $\{Y\}$. The set $\mathbb{D}$, containing nodes to test for non-orientability, is initialized to $\{Y\}$ (line 3), and the hop counter h is set to 1 (line 3). While there exists a node X adjacent to Y, which is not a child of Y, and there remain unoriented edges connected to Y, and not all nodes have been discovered, the exploration continues (line 4). At each iteration, the h-hop LEG is discovered using and updating previous knowledge through the known separating sets $\mathcal{S}$ and visited nodes (line 5). Next, all nodes connected by unoriented edges to any node in $\mathbb{D}$ and belonging to the h-hop neighborhood are added to $\mathbb{D}$ to form a candidate subset for the non-orientability criterion (lines 6–7). This ensures $\mathbb{D}$ contains nodes with unoriented edges and always includes Y. The algorithm checks whether the updated set $\mathbb{D}$ satisfies the non-orientability criterion (line 8); if so, discovery stops (line 9). Otherwise, the algorithm increments h and continues exploration (line 10). Upon termination (by any stopping criterion), two cases arise: (1) if X is adjacent to Y, not a child of Y, and unoriented edges remain connected to Y, then $CDE(x, x', y)$ is not identifiable (lines 11–12); (2) otherwise, the CDE is identifiable (lines 13–14).

The algorithm returns only identifiability and the LEG; however, estimation steps can be added when the CDE is identifiable. For example, in a linear setting, one can estimate it by regression adjusting on the parents of Y in the estimated LEG.

C EXPERIMENTS

We detail here the procedure used to generate the graphs and simulate the data, as well as how the evaluation metrics are computed.

Algorithm 1 LocPC

Require: Variables $\mathbb{V}$, target node Y, hop h, known sepsets $\mathcal{S}$

$\widehat{\mathcal{L}} =$ Fully unconnected graph over $\mathbb{V}$
$\mathbb{D} = [Y]$
visited $= [Y]$
$k = 0$
while $k \leq h$ **do**
  $\mathbb{D}_{\text{new}} = \emptyset$
  for $D \in \mathbb{D}$ **do**
    for $B \in \mathbb{V} \setminus$ visited **do**
      $\widehat{\mathcal{L}}$.addEdge$(D - B)$
  $s = 0$
  stop $=$ False
  while not stop **do**
    for $D \in \mathbb{D}$ **do**
      $adj(D) = Ne(D, \widehat{\mathcal{L}})$
    stop $=$ True
    for $D \in \mathbb{D}$ **do**
      visited $=$ visited $\cup \{D\}$
      for $B \in Ne(D, \widehat{\mathcal{L}})$ **do**
        if $|adj(D) \setminus \{B\}| \geq s$ **then**
          stop $=$ False
          for all $\mathbb{S} \subseteq adj(D) \setminus \{B\}$ **with** $|\mathbb{S}| = s$ **do**
            if $D \perp\!\!\!\perp B \mid \mathbb{S}$ **then**
              $Sepset(D, B) = \mathbb{S}$
              Update $\mathcal{S}$ with $Sepset(D, B) = \mathbb{S}$
              $\widehat{\mathcal{L}}$.removeEdge$(D - B)$
              break
      $\mathbb{D}_{\text{new}} = \mathbb{D}_{\text{new}} \cup Ne(D, \widehat{\mathcal{L}})$
    $s = s + 1$
  $\mathbb{D} = \mathbb{D}_{\text{new}}$
  $k = k + 1$
for every unshielded triple $A - B - C \in \widehat{\mathcal{L}}$ with only one node in $Neighborhood(Y, h+1, \widehat{\mathcal{L}})$ **do**
  if $B \notin Sepset(A, C)$ **then**
    Orient $A \rightarrow B \leftarrow C$
Apply rules LocMeek-Rule-1, LocMeek-Rule-2, and LocMeek-Rule-3 repeatedly on $\widehat{\mathcal{L}}$ until no more edges can be oriented.
for $D \in Neighborhood(Y, h, \widehat{\mathcal{L}})$ **do**
  for $A \notin Neighborhood(Y, h, \widehat{\mathcal{L}})$ **such that** $D - A \in \widehat{\mathcal{L}}$ **do**
    $LocRule =$ True
    for $W \in \mathbb{V} \setminus \{Neighborhood(Y, h, \widehat{\mathcal{L}}) \cup Ne(D, \widehat{\mathcal{L}})\}$ **do**
      if $A \in Sepset(D, W)$ **then**
        $LocRule =$ False
        break
    if $LocRule$ **then**
      $\widehat{\mathcal{L}}$.addEdge$(D \dashv A)$
return $\widehat{\mathcal{L}}, \mathcal{S}$, visited

C.1 DAGS GENERATION

All random graph models considered are homogeneous Erdős–Rényi models with edge existence probability $p = \frac{2}{|\mathbb{V}|-1}$. This ensures constant sparsity as the number of variables varies (on average, each node in the graph is adjacent to 2 edges).

Algorithm 2 LocPC-CDE

Require: Variables $\mathbb{V}$, treatment X, outcome Y, known sepsets $\mathcal{S}_0$

$\widehat{\mathcal{L}}, \mathcal{S}$, visited $=$ LocPC$(\mathbb{V}, Y, h = 0, \mathcal{S}_0)$
$\mathbb{D} = [Y]$
$h = 1$
while $X \in Ne(Y, \widehat{\mathcal{L}})$ and $X \notin Ch(Y, \widehat{\mathcal{L}})$ and $Ne_{\text{nar}}(Y, \widehat{\mathcal{L}}) \neq \emptyset$ and visited $\neq \mathbb{V}$ **do**
  $\widehat{\mathcal{L}}, \mathcal{S}$, visited $=$ LocPC$(\mathbb{V}, Y, h, \mathcal{S})$
  for $D \in \mathbb{D}$ **do**
    $\mathbb{D} = \mathbb{D} \cup \{Ne_{\text{nar}}(D, \widehat{\mathcal{L}}) \cap Neighborhood(Y, h, \widehat{\mathcal{L}})\}$
  if $\mathbb{D}$ satisfies the non-orientability criterion (Def. 4) **then**
    break
  $h = h + 1$
if $X \in Ne(Y, \widehat{\mathcal{L}})$ and $X \notin Ch(Y, \widehat{\mathcal{L}})$ and $Ne_{\text{nar}}(Y, \widehat{\mathcal{L}}) \neq \emptyset$ **then**
  identifiable $= False$
else
  identifiable $= True$
return identifiable, $\widehat{\mathcal{L}}$

C.1.1 Identifiable CDE Case

For each number of variables $|\mathbb{V}|$, we generate a DAG as follows:

1. Generate an undirected Erdős–Rényi graph, where each edge exists independently with probability p. Then, sample a random permutation σ of $\{1, \ldots, |\mathbb{V}|\}$ to define a topological (causal) order. For each undirected edge $i - j$, if $\sigma(i) < \sigma(j)$, orient the edge as $i \to j$. This results in a DAG whose sparsity is controlled by the edge probability p.
2. Convert the resulting DAG into its essential graph. Search for a pair of variables (X, Y) such that (i) $X \to Y$ is in the DAG (to ensure the existence of a direct effect) and (ii) all adjacents of Y are oriented in the essential graph. This guarantees identifiability of the CDE $CDE(x, x', y)$, according to Theorem 5.4 of [Flanagan, 2020].
3. If no such pair (X, Y) exists, generate a new random graph and repeat until the condition is met. The final graph thus guarantees that the direct effect from X to Y is identifiable.

Data is then simulated according to the linear/non-linear SCM as described below.

C.1.2 Non-Identifiable CDE Case

The procedure is similar, with a modified condition to ensure non-identifiability:

1. Generate an undirected Erdős–Rényi graph and orient it according to a random topological order σ, as described above.
2. Convert the DAG to its essential graph and look for a pair of variables (X, Y) such that (i) $X \to Y$ is present in the DAG, and (ii) at least one adjacent edge to Y remains unoriented in the essential graph. This guarantees that $CDE(x, x', y)$ is *not* identifiable, according to Theorem 5.4 of [Flanagan, 2020].
3. If no such pair (X, Y) exists, repeat the process until one is found.

The data is then simulated using the linear/non-linear SCM procedure described below.

C.2 DATA SIMULATION

C.2.1 Linear Gaussian SCM

Let $\mathcal{G}$ denote the causal structure. A linear Gaussian SCM can be expressed, for each variable V_i, $i = 1, \ldots, |\mathbb{V}|$, as:

$$V_i = \sum_{V_j \in Pa(V_i, \mathcal{G})} a_{j,i} V_j + \xi_i,$$

with noise $\xi_i \sim \mathcal{N}(0, \sigma_i^2)$. Equivalently, the model can be written in matrix form as:

$$\mathbb{V} = B\mathbb{V} + \boldsymbol{\xi},$$

where B is a coefficient matrix that can be permuted to lower-triangular form (due to the causal ordering) and $\boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ is a Gaussian noise vector of dimension $|\mathbb{V}|$. The solution is then given by:

$$\mathbb{V} = (I - B)^{-1}\boldsymbol{\xi}.$$

The simulation procedure is as follows:

1. Generate a random lower-triangular coefficient matrix B, with non-zero entries sampled uniformly from $\{x \in [-1, 1] : |x| > 0.2\}$;
2. Sample 5000 independent noise vectors $\boldsymbol{\xi}$, with each component $\xi_j \sim \mathcal{N}(0, \sigma_j^2)$ and $\sigma_j^2 \sim \mathcal{U}[0.8, 1]$;
3. For each noise vector, compute $\mathbb{V} = (I - B)^{-1}\boldsymbol{\xi}$, resulting in 5000 independent observations from the linear SCM.

C.2.2 Non-linear SCM

For the nonlinear case, we simulate binary variables to model categorical data commonly encountered in practice. Let $\mathcal{G}$ be the causal DAG. For each variable V_i, $i = 1, \ldots, |\mathbb{V}|$, the binary variable is generated as:

$$V_i = \mathbb{I}_{\xi_i \leq p_i},$$

where $\mathbb{I}$ is the indicator function, $\xi_i \sim \mathcal{U}([0, 1])$ is a uniform random variable, and

$$p_i = \frac{1}{1 + \exp\left(-\sum_{V_j \in Pa(V_i, \mathcal{G})} a_{j,i} V_j\right)}.$$

The coefficients $a_{j,i}$ are sampled uniformly from $\{x \in [-5, 5] : |x| > 0.2\}$. Then, 5000 independent observations are generated by first sampling vectors $\xi_i \sim \mathcal{U}([0, 1])$ of size 5000 and simulating the variables V_i following the causal ordering.

Note that this is equivalent to stating that each variable V_i, conditional on its parents, follows a Bernoulli distribution with parameter p_i: $V_i \mid Pa(V_i, \mathcal{G}) \sim \mathcal{B}(p_i)$.

C.3 ESTIMATION AND EVALUATION METRICS

For each method and each graph, we apply the causal discovery algorithm, which outputs either a fully or partially oriented causal graph. We evaluate the output based on the following criteria:

1. **Identifiability detection:** Whether the method correctly determines if all adjacents of Y are oriented (in the identifiable case) or not (in the non-identifiable case).
2. **Parent recovery (identifiable case only):** If all adjacents of Y are oriented, we compare the set of estimated parents of Y to the true parents.
3. **Computation time:** The time taken by each method is recorded.

The proportion of correctly identified non-identifiable graphs, shown in Figure 3, is computed as the ratio of graphs for which the method returns a non-identifiable output to the total number of graphs (100) for each value of $|\mathbb{V}|$.

F1 Score Calculation

The F1 score is computed to evaluate the accuracy of parent recovery in identifiable cases. Let $\widehat{Pa(Y)}$ be the set estimated by the method and $Pa(Y, \mathcal{G})$ be the set of true parents. Then:

$$\text{Precision} = \frac{|\widehat{Pa(Y)} \cap Pa(Y, \mathcal{G})|}{|\widehat{Pa(Y)}|}, \quad \text{Recall} = \frac{|\widehat{Pa(Y)} \cap Pa(Y, \mathcal{G})|}{|Pa(Y, \mathcal{G})|}, \quad F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

C.4 BASELINES IMPLEMENTATION

We use the implementations of the PC and LDECC algorithms from [Gupta et al., 2023], available at `https://github.com/acmi-lab/local-causal-discovery?utm_source=catalyzex.com`. These algorithms were minimally modified to report the same evaluation metrics as **LocPC-CDE** (number of CI tests, computation time, identifiability, etc.) and to use exactly the same conditional independence tests (`https://causal-learn.readthedocs.io/en/latest/independence_tests_index/index.html`), in order to ensure fair comparisons with respect to runtime. Aside from these adjustments, the original code remains unchanged.