Universiteit Leiden

Master Computer Science

Causal Effect Identification via Multi-Step Prompting

Name:	Syed Ahmed Numan
Student ID:	s3893200
Date:	28/08/2025
Specialisation:	Data Science
1st supervisor:	Dr. Saber Salehkaleybar
2nd supervisor:	Dr. Matthijs van Leeuwen

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Einsteinweg 55
2333 CC Leiden
The Netherlands

Abstract

This thesis studies the use of large language models (LLMs) for causal effect identification, through finding canonical adjustment sets in directed acyclic graphs (DAGs). We evaluate a range of prompting strategies, from simple prompts to instruction-based, chain-of-thought, and a multi-step decomposition approach. Two models are compared: GPT-4o mini which is a general-purpose LLM, and DeepSeek Reasoner, a reasoning model. Results show that prompt design plays a central role in model performance, where structured prompts consistently outperform simple prompts, with the multi-step decomposition approach achieving the best overall F1 score and accuracy. In particular, for 8-node DAGs, the multi-step decomposition approach achieves an F1 score of 0.90 and an exact match of 83%, whereas the best-performing baseline prompt attains an F1 score of 0.67 and an exact match of 53%.

Acknowledgment

First and foremost I would like to thank my supervisor Dr. Saber Salehkaleybar for being a pillar of support throughout my thesis journey. Your feedback, insights and clarity of thought paved my journey and our weekly meetings is something I will always cherish. To my second supervisor Dr. Matthijs van Leeuwen , thank you for taking time off your busy schedule to supervise my thesis and for providing valuable feedback to help improve this thesis. I would not be here without the unconditional support of my family and friends. Thank you all!

Contents

1 Introduction **5**

2 Preliminaries **8**
2.1 Frameworks for Causality 8
2.2 Graphical Definitions in Causal Models 8
2.3 Do-Calculus 10
2.3.1 Identification (ID) Algorithm 10
2.4 The `pcalg` Package 11
2.5 Adjustment Sets 11
2.6 Prompt-Based Reasoning with Large Language Models 12

3 Methodology **14**
3.1 Adjustment Set Computation 14
3.2 DAG Dataset Generation 14
3.3 Prompting Strategies 15
3.4 LLMs Used in the Study 16
3.5 Baseline Models 16
3.6 Summary of Methodological Choices 17

4 Experiments & Results **18**
4.1 Experimental Pipeline: 18
4.2 Evaluation Metrics 19
4.3 Final Results: 20
4.4 LLM Metrics Across Prompts 21
4.4.1 Latency across prompting techniques: 22
4.4.2 Latency vs Performance 23
4.4.3 Completion Tokens across Prompting techniques: 25

5 Conclusion **27**

6 Limitations and Future Work **28**

A Prompt Templates **32**
A.1 Simple Prompt 32
A.2 Prompt with Instructions 32
A.3 Chain-of-Thought (CoT) 33
A.4 Multi-Step Decomposition 34
A.4.1 Prompt 1: 34
A.4.2 Prompt 2: 35
A.4.3 Prompt 3: 35

B Example of LLM Errors: **36**

C Additional Result Tables: **38**
C.1 Evaluation Metrics Across DAG Sizes 38
C.2 Multi-Step Decomposition Prompt - Breakdown Stats: 40

1 Introduction

Causality as a concept expresses the relation or a process linking two or more distinct events, that consists of one event bringing about the other. Ordinary language contains many expressions denoting causation such as: x makes y happen, x induces y, etc.
Causality lies at the heart of scientific understanding. Ranging from Physics to Economics to now Machine Learning, the ability to answer not just 'what is' but 'why' and 'what would happen if' is a defining trait of causality and reasoning. Causality allows us to differentiate between correlation and influence, to be able to reason about interventions, and to make sense of the underlying structure of complex systems.

Real-World Relevance. In public health we can ask: "Does smoking cause lung cancer?" In economics we can ask: "What is the effect of a new tax policy on employment?" In climate science: "If CO_2 emissions are reduced, will global temperatures decrease?" These are all examples of fundamentally causal questions. Answering them requires more than just statistical associations; it requires a formal model that captures dependencies and counterfactual reasoning.

Causality as a whole has two interlinked tasks: *causal discovery* and *causal inference*. Causal discovery involves learning the structure of a system — which is usually represented in the form of a graph from observational data. It aims to identify which variables influence the other. Causal inference on the other hand, focuses on estimating the effect of interventions (such as treatments or policies), assuming that a valid causal graph is known or partially known. Both are equally important to causality: without discovery, we would not know which paths to consider and without inference, we would not be able to predict the consequences of actions.

Structural Causal Models and Causal Reasoning. Seminal work by Judea Pearl [Pearl, 2009b] and others has established Structural Causal Models (SCMs) as a standard framework for causal inference, building on earlier foundations such as Sewall Wright's path diagrams [Wright, 1921] and structural equation models [Jöreskog, 1973]. SCMs provide a systematic way to represent and reason about cause-and-effect relationships: variables are represented as nodes, and causal dependencies as directed edges. This thesis adopts this framework to analyze causal relationships, enabling formal reasoning through tools such as d-separation and do-calculus.
A key concept in structural causal models is the *valid adjustment set*, a set of variables that, when conditioned on, blocks all non-causal (backdoor) paths between a treatment and an outcome without blocking the causal path itself. Identifying such a set enables an unbiased estimation of the causal effect from observational data, as formalized in Pearl's backdoor criterion [Pearl, 2009b, Peters et al., 2017]. In practice, determining a valid adjustment set requires reasoning about the causal structure and applying concepts such as d-separation (explained later in the thesis).

Large Language Models. Large language models (LLMs) such as GPT [Achiam et al., 2023] and LLaMA [Touvron et al., 2023] have shown impressive performance across a wide range of reasoning tasks—including mathematics, logical reasoning, and scientific explanation. These models are trained on massive data and can parse complex instructions, generate structured text, and exhibit chain-of-thought reasoning. Their zero-shot capabilities make them one of the best tools for simulating causal effect identification. Popular examples of LLM reasoning include:

- **Chain-of-Thought Prompting** [Wei et al., 2022], which significantly improved performance on arithmetic and symbolic reasoning benchmarks like GSM8K via step-by-step reasoning prompts.
- **Zero-Shot Chain-of-Thought** [Kojima et al., 2022], showing how simple prompts like "*Let's think step by step*" boost zero-shot reasoning on math and logic datasets (e.g., GSM8K, MultiArith, AQUA-RAT).
- The **Pathways Language Model (PaLM)** [Chowdhery et al., 2022], a 540B-parameter model by Google, demonstrated strong performance on multi-step reasoning tasks when paired with chain-of-thought prompting.
- **AlphaGeometry** [Trivedi et al., 2024] is a neuro-symbolic system from DeepMind combining LLMs with symbolic engines, solved 25 out of 30 IMO geometry problems, nearly matching gold-medalist performance.
- OpenAI's **o3 series** (2024-2025), with "private chain-of-thought," achieved much stronger reasoning performance on benchmarks like GPQA Diamond, ARC-AGI, and Codeforces compared to prior models. These are a series of models and a technical report for them is available [OpenAI, 2023].
- **DeepSeekMath-7B** [DeepSeek-AI, 2024], an open model trained with extensive math data, reached GPT-4-level performance on the MATH benchmark using self-consistency techniques.

Can LLMs perform causal reasoning? LLMs demonstrated strong performance in general reasoning and linguistic tasks, but their capabilities in structured causal tasks, such as identifying valid adjustment sets or reasoning over DAGs, have been underexplored. The CLadder benchmark introduced by [Jin et al., 2023] provided one of the first systematic evaluations of LLMs on formal causal inference tasks, by converting causal graphs and queries (including associational, interventional, and counterfactual questions) into natural language and testing performance using a chain-of-thought prompting variant called CausalCoT. This benchmark was found to be highly challenging for state-of-the-art models. Similarly, [Zhou et al., 2024] developed CausalBench, which assessed LLMs on tasks of increasing complexity such as skeleton identification and cause-effect reasoning. This study observed that LLMs struggled especially on larger-scale networks with collider structures.
Other recent evaluations reinforced these findings. [Chen et al., 2024] proposed the CaLM framework which is a large-scale benchmark including causal targets, adaptation strategies, and error analysis. This study showed that even high-capacity LLMs consistently failed in intervention and counterfactual reasoning tasks over hundreds of thousands of examples. These studies collectively showed that while LLMs can provide plausible responses through pattern matching, they lacked robust causal reasoning.

This thesis studies the potential and limitations of LLMs in *causal inference*. Specifically, we evaluate whether language models can understand and apply formal causal concepts such as backdoor paths, adjustment sets, and d-separation. In particular, whether they can identify valid adjustment sets for estimating causal effects, using chain-of-thought (CoT) reasoning strategies as suggested in prior work on causal reasoning with LLMs [Jin et al., 2023, Zhou

et al., 2024]. And finally, whether their performance is improved if the prompt is split into multiple steps.

Thesis Structure. The remainder of this thesis is organized as follows. The Preliminaries section reviews the necessary background on causal inference, graphical models, and prompting strategies for large-language models. The Methodology section details the setup, including the generation of synthetic DAGs, the computation of ground-truth adjustment sets, and the prompting strategies used. The Experiments & Results section presents the experimental setup and evaluation results, comparing LLM outputs to the ground truth and analyzing reasoning as well as prompt quality. The Limitations & Future work section lists the limitations of this study and outlines future research directions, and finally the Conclusion section summarizes key findings to conclude the thesis.

2 Preliminaries

This section provides the foundational concepts required to understand the experiments and analyses carried out in this thesis. Starting with an overview of causality, followed by a discussion of Directed Acyclic Graphs (DAGs), which are important for the representation and reasoning tasks in causal inference and discovery.

2.1 Frameworks for Causality

Traditional statistical methods focus mainly on analyzing associations within observational data by estimating parameters of a probability distribution. These analyses are limited to observation. Causal inference on the other hand aims to measure the effect of interventions, asking not just "What is?", but "What if?"
Two widely recognized frameworks for causal inference are:

- **Structural Causal Models (SCMs)** [Pearl, 2009b], which provide a formal mathematical framework for representing and reasoning about cause-effect relationships. SCMs combine a set of structural equations with a corresponding causal graph, where nodes represent variables and edges denote causal influences. This framework enables the formulation of causal queries using the do-calculus and the identification of causal effects via graphical criteria such as d-separation and the backdoor criterion [Pearl, 2009b, Peters et al., 2017].
- **Potential Outcomes Framework** [Rubin, 1974, Imbens and Rubin, 2015], which formalizes causal effects by comparing potential outcomes under different interventions. In this framework, each unit has a set of potential outcomes—one for each possible treatment condition—and the causal effect is defined as the comparison between these outcomes. The framework provides several statistical approaches to causal inference, such as randomized controlled trials and matching methods.

2.2 Graphical Definitions in Causal Models

A *Directed Acyclic Graph* (DAG) is a graph $G = (V, E)$, where V is the set of vertices (variables) and E is the set of directed edges, such that there are no directed cycles. In causal modeling, each node represents a variable, and an edge $X \to Y$ denotes that X is a direct cause of Y. DAGs encode assumptions about conditional independence via the *Markov property*: each variable is independent of its non-descendants, given its parents. This property allows conditional independence relationships to be read from the graph using the notion of *d-separation* [Pearl, 1988].

Blocking and d-separation. A path between two variables X and Y is said to be *blocked* by a conditioning set Z if at least one of the following holds:

- The path contains a chain $A \to B \to C$ or a fork $A \leftarrow B \to C$ where $B \in Z$.
- The path contains a collider $A \to B \leftarrow C$ where neither B nor any descendant of B is in Z.

If all paths between X and Y are blocked by Z, we say X and Y are *d-separated* by Z. d-separation provides a formal link between graphical structure and probabilistic independence.

Treatment and outcome. In causal inference, the *treatment* variable X is the variable whose effect we aim to estimate, and the *outcome* variable Y is the variable potentially influenced by X. The choice of treatment and outcome depends fully on the research question and must be clearly defined before analysis.

Backdoor and causal paths. A *backdoor path* from a treatment variable X to an outcome variable Y is any path that begins with an arrow into X [Pearl, 1994]. Such paths represent potential confounding and must be blocked to avoid bias. For example, in $X \leftarrow Z \rightarrow Y$, the variable Z is a common cause that opens a backdoor path unless conditioned upon. A *causal path*, in contrast, is a path from X to Y where all edges are oriented away from X toward Y, representing the direct mechanism of causal influence.

Conditional independence. Two variables X and Y are said to be *conditionally independent* given a set of variables Z, denoted $X \perp\!\!\!\perp Y \mid Z$, if the joint distribution factorizes such that knowledge of Y does not provide any additional information about X once Z is known [Dawid, 1979]. In graphical models, conditional independence corresponds to d-separation.

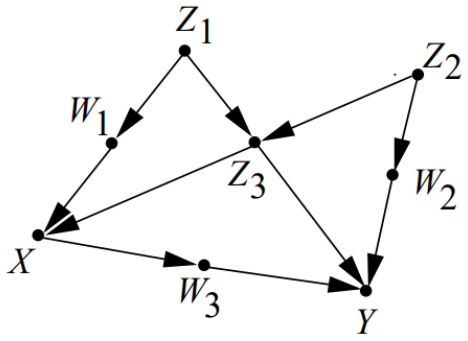


Figure 1: A graphical model illustrating the back-door criterion [Pearl, 2009a]

As per Pearl's overview on causal inference [Pearl, 2009a], in an observational study, our goal is to estimate the causal effect of a treatment X on a response Y. The factors that influence this relationship can include both measurable variables (such as age, gender) and unmeasurable ones (like genetic traits, lifestyle). The challenge is to identify a subset of these factors to adjust for, so that comparing treated and untreated individuals with the same values of the selected factors yields the correct causal effect. Such a subset is called a "sufficient set" for adjustment.
Based on the back-door criterion (Figure 1) [Pearl, 2009a], we see, for example, that the sets $\{Z_1, Z_2, Z_3\}$, $\{Z_1, Z_3\}$, and $\{W_2, Z_3\}$ are each sufficient for adjustment, as they block all back-door paths between X and Y. The set $\{Z_3\}$, however, is not sufficient, since it fails to block the path $X \leftarrow W_1 \leftarrow Z_1 \rightarrow Z_3 \leftarrow Z_2 \rightarrow W_2 \rightarrow Y$. The implication of identifying a sufficient set S is that stratifying on S removes all confounding bias, making the causal effect of X on Y identifiable.

Causal Effect and Interventions. The *causal effect* of X on Y measures the change in Y's distribution that would occur under an intervention setting X to a specific value. Pearl's

do-operator [Pearl, 2009b] formalizes this as follows: Under the structural causal model (SCM) framework, an intervention $do(X = x)$ replaces the structural equation for X with $X := x$ and removes all incoming edges to X in the DAG.

2.3 Do-Calculus

Do-calculus, introduced by Pearl [Pearl, 1994, 2009b], provides a formal system of inference rules for transforming an expression involving interventions, such as $P(y \mid \text{do}(x))$, equivalent expressions involving only observational conditional distributions. The three rules of do-calculus establish conditions under which intervention operators can be replaced or removed, making it possible to reduce causal queries into terms that can be estimated from data. The three rules are as follows:

- **Insertion/deletion of observations:**

$$P(y \mid \text{do}(x), z, w) = P(y \mid \text{do}(x), w) \quad \text{if } (Y \perp Z \mid X, W)_{G_{\overline{X}}}$$

  This rule allows conditional variables to be added or removed if they are d-separated in the graph where incoming edges into X are removed ($G_{\overline{X}}$).

- **Action/observation exchange:**

$$P(y \mid \text{do}(x), \text{do}(z), w) = P(y \mid \text{do}(x), z, w) \quad \text{if } (Y \perp Z \mid X, W)_{G_{\overline{X},\underline{Z}}}$$

  This rule allows replacing an intervention with an observation when the d-separation holds in the modified graph where edges into X are removed and edges out of Z are deleted.

- **Insertion/deletion of actions:**

$$P(y \mid \text{do}(x), \text{do}(z), w) = P(y \mid \text{do}(x), w) \quad \text{if } (Y \perp Z \mid X, W)_{G_{\overline{X},\overline{Z(W)}}}$$

  This rule allows adding or removing interventions if the d-separation holds in the graph where incoming edges into X and into Z (excluding those that are ancestors of W) are removed.

2.3.1 Identification (ID) Algorithm

Building on do-calculus, the Identification (ID) algorithm was developed to provide a systematic procedure to determine whether a causal effect is identifiable from observational data given a causal graph [Shpitser and Pearl, 2006]. The algorithm takes as input a causal query (e.g., $P(y \mid \text{do}(x))$) and a graph, and outputs either an expression for the effect in terms of observational quantities or a statement that the effect is not identifiable. However, the output of the ID algorithm is not necessarily unique, as multiple expressions can identify the same causal effect. For this reason, this study focuses on adjustment set–based identification, which yields a unique expression for the causal effect and is better suited for evaluating prompting strategies with LLMs.

2.4 The `pcalg` Package

The `pcalg` package in R [Kalisch et al., 2012] provides implementations of several well-known algorithms to learn causal structures and for causal effect estimation from observational data. Its core methods include the Peter–Clark (PC) algorithm [Spirtes et al., 2000], the Fast Causal Inference (FCI) algorithm and its computationally efficient variant, the Really Fast Causal Inference (RFCI) algorithm [Spirtes et al., 1999, 2000, Colombo et al., 2012], as well as identifying valid adjustment sets.
The PC algorithm (named after its inventors Peter Spirtes and Clark Glymour) is a constraint-based method that uses conditional independence tests to determine the structure of the given causal graph. It starts with a fully connected undirected graph and incrementally removes edges when conditional independencies are detected. The resulting structure represents a Markov equivalence class of directed acyclic graphs (DAGs) that are in line with the observed independencies.
One of the main functions in causal effect estimation with `pcalg` is the identification of **adjustment sets**, which are subsets of variables that, when conditioned upon, block all non-causal paths between a treatment and an outcome. Such sets are important because they enable an unbiased estimation of causal effects from the given observational data.

2.5 Adjustment Sets

An *adjustment set* is a set of variables Z that, when conditioned on, blocks all backdoor paths from a treatment variable X to an outcome variable Y without introducing bias through conditioning on the descendants of X. The *backdoor criterion* [Pearl, 1994] states that Z is a valid adjustment set if:

- No element of Z is a descendant of X.
- Z blocks all backdoor paths between X and Y.

Given a valid Z, the causal effect of X on Y can be computed via:

$$P(Y \mid do(X)) = \sum_z P(Y \mid X, z)\, P(z).$$

In practice, adjustment sets can be computed in `pcalg` using the `adjustment()` function. The main inputs to this function are a directed acyclic graph (DAG), usually represented by an `igraph` or adjacency matrix, together with user-specific treatment and outcome variables. Given these inputs, `adjustment()` returns one or more sets of variables that satisfy the backdoor criterion and allow for unbiased estimation of the causal effect of the treatment on the outcome. The output depends on the parameter `set.type`, which can be set to *"all"*, *"minimal"*, or *"canonical"*. These are essentially three variants of adjustment sets. If `set.type` is *"all"*, the output contains all valid adjustment sets that satisfy the backdoor criterion. If `set.type` is *"minimal"*, only minimal sufficient adjustment sets are returned; these are valid sets for which no proper subset is also valid. Finally, if `set.type` is *"canonical"*, a single adjustment set is produced consisting of all (possible) ancestors of X and Y, excluding (possible) descendants of nodes on proper causal (the exact definition is given in the following). This canonical set is guaranteed to be valid if any valid adjustment set exists.

Canonical Adjustment Set: Definition

As per `pcalg` package [Kalisch et al., 2012] if set.type is "canonical", a single adjustment set is returned that consists of all (possible) ancestors of x and y, minus (possible) descendants of nodes on proper causal paths. This canonical adjustment set is always valid if any valid set exists at all.
The definition used for the canonical adjustment set is consistent with the implementation in the `pcalg` package [Kalisch et al., 2012]. Given a DAG G, treatment variable X, and outcome variable Y, the canonical adjustment set Z_{can} is defined as:

$$Z_{\text{can}} = \text{An}(Y) \setminus (\text{De}(X) \cup \text{Forb}(X, Y)),$$

where:

- $\text{An}(Y)$ denotes the set of ancestors of Y (excluding Y itself),
- $\text{De}(X)$ denotes the set of descendants of X (including latent variables),
- $\text{Forb}(X, Y)$ is the set of all descendants of nodes that lie in both $\text{An}(Y)$ and $\text{De}(X)$.

Computation Procedure

The canonical adjustment set is computed following the procedure described in the `pcalg` R package [Kalisch et al., 2012]:

1. Identify all descendants of the treatment node X, denoted $\text{De}(X)$.
2. Identify all ancestors of the outcome node Y, excluding X, denoted $\text{An}(Y)$.
3. Determine the set $\text{Forb}(X, Y)$ by finding the intersection $\text{De}(X) \cap \text{An}(Y)$ and then computing all descendants of nodes in this intersection.
4. Subtract both $\text{De}(X)$ and $\text{Forb}(X, Y)$ from $\text{An}(Y)$ to obtain the canonical adjustment set Z_{can}.[1]

In this thesis, the canonical adjustment set is used as the ground truth for evaluation. This decision was purely because of its formal definition and ease of expression in plain language for large language models (LLMs). Unlike the set of all or minimal valid sets, the canonical set yields a single, deterministic target - avoiding ambiguity in evaluation.

2.6 Prompt-Based Reasoning with Large Language Models

A *prompt* can be defined as the input text provided to a Large Language Model (LLM) serving as its conditioning signal for the text generation process. The prompt specifies the task, provides relevant context, and defines the form of the output. Prompts can be constructed from templates that define placeholders for task-specific variables, allowing experimentation with different phrasings. The effectiveness of a prompt depends both on its quality/clarity, completeness, and structure—and on the underlying capabilities of the LLM [Wei et al., 2022, Liu et al., 2023].

[1] In this work, the above steps were implemented in Python using the `networkx` library to replicate the functionality of the `pcalg` package.

In-Context Learning. In-context learning (ICL) is a fundamental concept in prompting where the model "learns" from examples embedded directly within the prompt, without updating its parameters [Brown et al., 2020]. This enables fast adaptation to new tasks by conditioning on a small set of examples. ICL can be categorized into *zero-shot*, *one-shot*, and *few-shot* settings, depending on the number of examples provided. In zero-shot prompting, only task instructions are given. One-shot and few-shot prompts include one or several input–output examples, to guide the model's response. The mechanism behind ICL has been interpreted through several lenses: Bayesian inference over implicit hypotheses [Xie et al., 2022], gradient descent on latent representations [Akyürek et al., 2022], and meta-learning over tasks [Dong et al., 2022].

Prompt Engineering Strategies. The design of effective prompts often requires iterative refinement, guided by extensive testing and task-specific constraints [Liu et al., 2023]. Several strategies have proven to be effective:

- **Clear Descriptions:** Prompts should specify the task in a precise manner and must avoid ambiguous phrasing. This is essential in structured reasoning tasks such as causal inference.
- **Guiding the Reasoning Process:** Including phrases such as "Let's think step by step" can activate more deliberate reasoning and lead to improved performance in multi-hop tasks [Kojima et al., 2022].
- **Providing Reference Information:** Adding relevant domain knowledge such as defintions and formalizations within the prompt can constrain outputs to be factual and context-appropriate. Retrieval-Augmented Generation (RAG) [Lewis et al., 2020] represents this, combining retrieved documents with the prompt.
- **Attention to Format:** The structure of the prompt—including bullet points, numbered steps and delimiters can help the model parse information more effectively and reduce misinterpretation [Liu et al., 2023].

Advanced Prompting Methods. Besides basic prompting, there are more advanced prompting methods that aim to draw out intermediate reasoning steps. An extensively studied example is *Chain of Thought* (CoT) prompting [Wei et al., 2022], where the model is instructed to generate a sequence of intermediate inferences before arriving at the final answer. CoT can be combined with few-shot demonstrations (*few-shot CoT*) to further improve performance of complex reasoning tasks [Kojima et al., 2022]. This approach is relevant for causal inference tasks, where arriving at the final answer, i.e., valid adjustment sets, requires splitting the problem into subproblems such as identifying ancestors, descendants, and blocking sets.
In this thesis, these prompting strategies from direct instructions to reasoning-oriented methods are applied to the task of identifying valid adjustment sets from text-based DAG descriptions. The variation in prompt design allows an in-depth analysis of how different levels of reasoning and prompting strategies influence LLM performance in causal reasoning.

3 Methodology

While the ID algorithm offered a complete solution for causal effect identification, implementing it in our LLM-based setting was challenging, as there might be no unique expression for causal effect identification in the general case. Therefore, the focus of this thesis shifted toward evaluating LLMs through their ability to identify *valid adjustment sets*. This direction allowed for a more verifiable solution: adjustment sets can be directly computed using established causal inference libraries such as `pcalg` in R [Kalisch et al., 2012], giving us a clear ground truth. This change also facilitated the comparison of multiple prompting strategies, models, and baselines using objective correctness measures.
Therefore the main goal of the study is: *Can LLMs reliably identify valid adjustment sets in causal graphs, given knowledge of graphical rules such as the back-door criterion and d-separation?* This question serves as a proxy for the broader goal of understanding whether LLMs can exhibit causal reasoning capabilities.

3.1 Adjustment Set Computation

When randomized experimentation is not feasible, we have to rely on observational data. One common criterion is to adjust for a set of covariates Z that block all back-door paths from X to Y. This approach is based on the *back-door criterion* [Pearl, 1994], which ensures that the association between X and Y reflects the causal effect rather than confounding.
While several sets might satisfy the backdoor criterion, the **canonical adjustment set** provides a deterministic choice that can be used as a reference point. The canonical set is especially useful for evaluation purposes because it is strictly defined under a given DAG and can be computed using existing algorithms. In this study, the canonical adjustment set was used as ground truth to measure precision, recall, F1 score, and exact match accuracy, while "all adjustment sets" were used as ground truth to measure the overall accuracy - to see if the LLM predicted set was valid for the given DAG, treatment and outcome or not.

3.2 DAG Dataset Generation

A key part of the experimental setup involved generating synthetic directed acyclic graphs (DAGs) to serve as the dataset and input for both the LLMs and the ground-truth adjustment set computation. Synthetic graphs were preferred over real-world causal models to maintain control over their structure, size, and complexity.

Remark on the Initial Pilot Experiment. In early trials, DAGs were encoded as textual lists of directed edges (e.g., "X1 $\rightarrow$ X2") and directly presented to a GPT-based LLM. This approach, however, had several limitations:

- Text-based edge representations proved difficult for LLMs to parse consistently, a limitation also observed in recent work showing that reducing graph structures to plain-text edge lists can hinder structured reasoning performance [Agrawal et al., 2025].
- Prompting results varied significantly over repeated runs with identical inputs.
- The use of a web-based interface introduced unintended cross-session memory effects, which contaminated the independence of experimental trials.

Due to these challenges, the encoding strategy and experimental setup were revised to an API based approach with structured DAGs and a fixed ground truth.
In this work, DAGs were generated with a fixed number of nodes, where edges were added probabilistically under the constraint of acyclicity.[2] For each DAG, one node was randomly assigned as the treatment variable X and another distinct node as the outcome variable Y. The DAG structure, along with X and Y, was then used both for computing canonical adjustment sets with the `pcalg` package and for constructing prompts for the LLMs.

Testing with different sized DAGs

To test LLM performance under varying graph complexity, additional datasets were generated:

- **4-node DAGs**: Simple graphs with no latent variables.
- **6-node DAGs**: Medium-complexity DAGs, also without latent variables.
- **8-node DAGs with latent variables**: More complex DAGs where one randomly selected node was marked as unobserved or latent.

Each graph was generated with a random edge probability from the set $\{0.2, 0.4, 0.6, 0.8\}$, to control density. The treatment and outcome were selected uniformly at random, while making sure they were distinct. It was also ensured that both the treatment and the outcome were observable and not latent.
This variation allowed for studying how LLM performance varied with graph size and latent confounding, two important dimensions of complexity in causal inference.

3.3 Prompting Strategies

Prompt design played a crucial role in determining the quality and correctness of outputs generated by the LLMs. To systematically study how different prompt formats affect causal reasoning, four prompting strategies were evaluated:

- **Simple Prompt** — A minimal prompt that directly presents the DAG structure along with the treatment and outcome variables, asking the model to identify the canonical valid adjustment set without providing any definitions or instructions.
- **Prompt with Instructions** — Included textual definitions of the back-door criterion and related causal graph concepts, along with step-by-step guidance to find the canonical adjustment set.
- **Chain-of-Thought (CoT)** - Encouraged the model to reason step by step by enumerating back-door paths, identifying colliders, and eliminating invalid adjustment sets before producing an answer. Included a worked example with ground truth.
- **Multi-Step Decomposition** — Inspired by cognitive decomposition in AI research [Kramer and Baumann, 2024] and problem decomposition principles outlined in *Foundations of Large Language Models* [Xiao and Zhu, 2025], this prompt splits the task into three labeled subtasks: identifying ancestors and descendants, computing the forbidden set, and compiling the adjustment set, in order to distribute the model's reasoning effort.

The exact text of each prompt used in the experiments is provided in Appendix A.

[2] Implementation was carried out in Python using the `networkx` library.

3.4 LLMs Used in the Study

This study evaluated the causal reasoning capabilities of two state-of-the-art large language models:

- GPT-4o Mini (OpenAI, 2024)
- DeepSeek Reasoner (DeepSeek, 2024)

GPT-4o Mini

GPT-4o Mini ("o" for "omni") is a fast, efficient variant of OpenAI's flagship GPT-4 Omni model, designed for both text and image inputs. According to available documentation, it is a smaller and more cost-efficient version of GPT-4o, optimized for compact inference and lower latency.[OpenAI] The model supports a context window of 128,000 tokens, can produce up to 16,384 output tokens, and has a knowledge cutoff date of October 1, 2023.

DeepSeek Reasoner

DeepSeek Reasoner is designed specifically for reasoning-heavy tasks, featuring embedded Chain-of-Thought (CoT) generation capabilities. Its API returns both the final answer and the complete CoT as part of the response. The model supports up to 64,000 tokens of context and has a default maximum combined output length of 32,000 tokens, which can be extended to 64,000 tokens when including CoT. The DeepSeek Reasoner (DeepSeek-R1) architecture contains 671 billion total parameters, with 37 billion parameters activated during inference, and supports a context window of 128,000 tokens, as documented in the model's specifications. [DeepSeek]

General vs. Reasoning LLMs

This study aims to investigate whether general-purpose LLMs and reasoning-based LLMs show similar or differing levels of proficiency in tasks that require causal reasoning. General-purpose LLMs are generally optimized for a broad range of language tasks, while reasoning-specialized LLMs are explicitly tuned for logical manipulation and the decomposition of complex problems, making them more suited for tasks such as causal inference.
In this work, we do not fine-tune any models. Instead, we rely solely on in-context learning zero-shot and one-shot prompts to evaluate the models' causal reasoning capabilities. This approach allows us to see how reasoning-specialized LLMs compare to general-purpose LLMs when both are given the same minimal in-context examples, and to what extent causal reasoning ability emerges from general LLM training without additional specialization.

3.5 Baseline Models

To provide a baseline to measure the performance of the LLMs, two types of randomized baselines were coded. These baselines serve as example of a naive models that does not rely on any understanding of the graphical structure or the rules of causal inference. Their inclusion is important to see whether the LLMs are actually performing meaningful reasoning or simply guessing.

Fixed-Length Random Baseline

The first baseline generates adjustment sets of a fixed length, chosen based on the most frequently observed size of canonical adjustment sets in the ground truth data. For example, if the majority of valid adjustment sets across all DAGs contain two variables, then each prediction from this baseline is a randomly sampled set of two observed nodes (excluding X and Y and latent variables).
This baseline operates under the assumption that the model "knows" the typical size of an adjustment set, but has no information about which variables are causally relevant. This makes it a useful lower-bound comparison for models that do have access to the graph structure.

Distribution-Based Random Baseline

The second baseline uses a distribution of adjustment set lengths from the ground truth data to randomly sample a set size. For each DAG, a set size is sampled according to this distribution, and then a random subset of that size is selected from the observed nodes (again excluding X and Y and latent).

3.6 Summary of Methodological Choices

This chapter outlines a systematic approach to evaluating causal reasoning in LLMs through the task of identifying valid adjustment sets from DAGs. The key methodological decisions can be summarized as follows:

- **Focus on Adjustment Sets**: Rather than attempting to identify causal effects using all possible identification strategies, this study specifically focuses on estimating causal effects by identifying the valid adjustment sets.
- **DAG Generation**: Synthetic DAGs were generated with different node sizes (4, 6, and 8) and randomized edges. The 8-node DAGs included one latent variable to test robustness in the presence of unobserved confounding.
- **Ground Truth via** `pcalg`: All valid adjustment sets were computed using `pcalg` in R and python's networkx package, making sure of correctness grounded in established causal algorithms.
- **LLMs and Prompts**: The performance of two LLMs — GPT-4o Mini and DeepSeek Reasoner — was evaluated under four different prompt templates, focusing on the effect of prompt complexity.
- **Baselines**: Randomized baselines were coded from scratch to ensure that model performance could be benchmarked against naive guessing strategies as there is no prior benchmark for the process.

4 Experiments & Results

This section presents a comprehensive evaluation of the performance of the two chosen large language models—GPT-4o-mini and DeepSeek Reasoner, in identifying canonical valid adjustment sets from directed acyclic graphs (DAGs). We compare these models against the two random baseline approaches and analyze their performance under the four chosen distinct prompting strategies - Simple-Prompt, Prompt with Instructions, Prompt with CoT and Multi-Step Decomposition. In addition to standard performance metrics such as precision, recall, F1 score, exact match, and accuracy, we also evaluated computational attributes such as latency and token usage. The results highlight the effectiveness of structured multi-step decomposition in improving both accuracy and efficiency, particularly for reasoning-focused models. Before the final results - the experimental pipeline and evaluation metrics are outlined below.

4.1 Experimental Pipeline:

To address the limitations of early experiments, the following pipeline was strictly followed:

- **API-based access**: All LLMs were programmatically queried via their respective APIs to ensure stateless and reproducible interactions.
- **Structured DAG encoding**: Graphs were represented as JSON-formatted adjacency lists, which facilitate reliable parsing and reasoning by LLMs—a representation format shown to outperform natural language or code representations in recent empirical evaluations [Zhu et al., 2024].
- **Ground truth computation**: The canonical and all valid adjustment sets were computed using R's `pcalg` package [Kalisch et al., 2012], with graphs encoded via Python's `networkx` library.
- **Metadata storage**: Each DAG was stored along with metadata, including treatment and outcome nodes, any latent variable(s), the canonical adjustment set, and the list of all valid adjustment sets, to support both ground-truth computation and prompt construction.

This design helped to create a more systematic experimentation pipeline that could be applied to different types of DAGs and models.

Input to the Model

Each model received the full structure of a DAG encoded as a JSON-formatted edge list in a clean textual format, along with explicit specifications of the treatment variable X and outcome variable Y. When applicable, the prompt also indicated the presence of latent (unobserved) variables. For simplicity and consistency, the graph nodes were labeled using numbered variables (e.g., `X0, X1, X2`).

Output Format and Evaluation

The models were instructed to return a valid adjustment set, or multiple sets if applicable, using a consistent list notation such as {`X1, X3`}. In cases where no valid adjustment set existed,

the models were asked to state this explicitly. The predicted outputs were then compared to the ground truth adjustment sets computed using `pcalg` in R and Python. An output was considered correct if it exactly matched the canonical valid adjustment set for that DAG, and accurate if it matched any valid adjustment set for the given conditions.

4.2 Evaluation Metrics

To quantitatively evaluate the performance of the LLMs in identifying the canonical valid adjustment sets, five metrics were used.

Precision

Precision measures the proportion of variables in the model's predicted set that are part of the ground truth canonical adjustment set.

$$\text{Precision} = \frac{|\hat{Z} \cap Z^*|}{|\hat{Z}|},$$

where $\hat{Z}$ is the model's predicted adjustment set, and Z^* is the valid ground truth canonical adjustment set.

Recall

Recall captures the proportion of ground truth canonical adjustment set variables that are correctly identified by the model:

$$\text{Recall} = \frac{|\hat{Z} \cap Z^*|}{|Z^*|}.$$

F1 Score

The F1 score is the harmonic mean of precision and recall, and provides a more balanced measure that penalizes both false positives and false negatives:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Exact Match & Overall Accuracy

A prediction was marked correct (Exact Match) if:

- The predicted set exactly matched the ground truth canonical valid adjustment set (exact set match), or
- The model correctly stated that no valid adjustment set exists when that is true.

A prediction was marked Accurate - if the predicted set was "valid" for the given graph and treatment/outcome pair.
Exact match shows the proportion of examples for which the model gave a completely correct answer. Whereas "Overall accuracy" shows the proportion of examples for which the LLM

gave us a "valid adjustment set", not necessarily "canonical". Each of these metrics provides information into a different aspect of the model's performance, from partial reasoning ability (via F1) to complete structural correctness (via exact match).

4.3 Final Results:

The final experimental results are summarized in Table 1 below. This table presents a comprehensive comparison of two Large Language Models, GPT-4o-mini and DeepSeek Reasoner, against two baseline models across Directed Acyclic Graphs (DAGs) of 4, 6, and 8 nodes. Performance is measured using F1, Exact Match, and Accuracy scores, evaluating four distinct prompting strategies: Simple Prompt, Prompt with Instructions, Chain-of-Thought, and Multi-Step Decomposition.

Model	Prompting Strategy	4-node DAGs			6-node DAGs			8-node DAGs		
		F1	Exact	Acc.	F1	Exact	Acc.	F1	Exact	Acc.
Baselines	Baseline 1 (Fixed)	0.30	0.33	0.33	0.26	0.24	0.27	0.17	0.17	0.20
	Baseline 2 (Dist.)	0.35	0.37	0.37	0.35	0.29	0.33	0.21	0.23	0.27
GPT-4o-mini	Simple Prompt	0.61	0.37	0.49	0.64	0.32	0.44	0.45	0.29	0.39
	Prompt w/ Instructions	0.75	0.49	0.64	0.74	0.41	0.59	0.55	0.37	0.51
	Chain-of-Thought	0.75	0.55	0.61	0.70	0.48	0.55	0.50	0.40	0.44
	Multi-Step Decomp.	**0.93**	**0.85**	**0.93**	**0.89**	**0.74**	**0.86**	**0.83**	**0.75**	**0.81**
DeepSeek Reasoner	Simple Prompt	0.71	0.48	0.60	0.75	0.50	0.62	0.60	0.48	0.59
	Prompt w/ Instructions	0.84	0.65	0.79	0.84	0.59	0.71	0.67	0.53	0.60
	Chain-of-Thought	0.81	0.72	0.81	0.79	0.50	0.55	0.62	0.52	0.57
	Multi-Step Decomp.	**0.95**	**0.93**	**0.99**	**0.92**	**0.86**	**0.95**	**0.90**	**0.83**	**0.92**

Table 1: Final Results - All Baselines & Prompts with F1, Exact Match, and Accuracy

Analysis: The comparison against the randomized baselines clearly highlights the limitations of random selection strategies, which achieved relatively low performance across all DAG sizes, with F1 scores rarely exceeding 0.35 and accuracies remaining below 0.37. In contrast, GPT-4o-mini, which is just a general purpose language model, even when guided only by a simple prompt, consistently outperforms these baselines.

Comparison of results with the baselines show that even minimal task framework provides LLMs with a significant advantage over random selection, underlining the importance of semantic reasoning and context in identifying valid adjustment sets.

Across both models, performance improved generally as the DAG size decreased, with the highest F1 scores and Exact Match observed for the 4-node graphs. For both GPT-4o-mini and DeepSeek Reasoner, more structured prompting strategies outperformed the Simple Prompt as well as the randomized baseline strategies. In particular, the *Prompt with Instructions* strategy consistently achieved stronger results than Simple Prompt, and the Chain-of-Thought approach performed comparably. These findings show that the addition of explicit causal reasoning instructions improves the models' ability to identify valid adjustment sets.

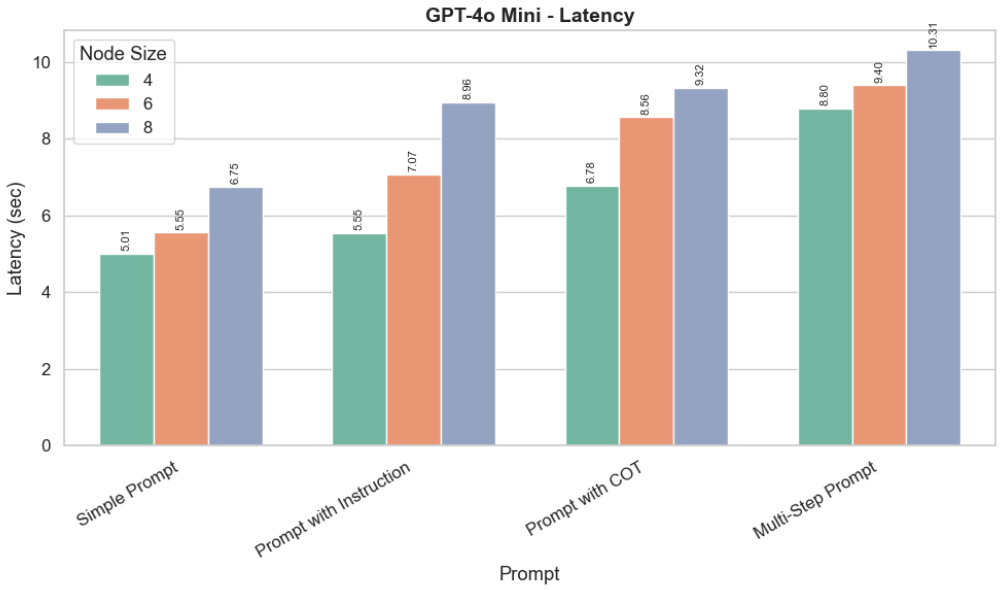
The Multi-Step Decomposition approach achieved the strongest overall performance across all DAG sizes and models, outperforming single-prompt, zero-shot (prompt with instructions) and

one-shot strategies (chain of thought). The overall pipeline results in Table 1 reflect overall F1, Exact match and Accuracy. Both GPT-4o-mini and DeepSeek Reasoner achieved markedly higher F1, Exact Match, and Accuracy under this approach, with DeepSeek Reasoner reaching close to a perfect scores on 4-node DAGs and having good performances on larger graphs.
This improvement comes from the nature of the errors that the LLMs were observed to make in prior approaches. When reasoning is carried out in a single pass, the model often fails to recursively track ancestor or descendant relationships. Another error noted was the hallucination of edges that do not exist in the graph. By splitting the task into smaller subtasks (e.g., first identifying ancestors, then descendants, finally applying the adjustment criterion), these mistakes are surfaced and can be limited at each stage by explicit prompt updates.
A second advantage is that the multi-step approach avoids compounding errors that occur when the LLM relies on its own generated output across steps in a single chain of reasoning. For example, when reasoning internally, the model was observed to introduce inconsistencies in its sets of ancestors or conditioning variables between steps, as seen in the example given in Appendix B. In contrast, the decomposition strategy explicitly stores the intermediate outputs and reuses them as inputs for subsequent prompts, forcing consistency across stages.
These factors together make the multi-step approach substantially more effective, as it distributes the reasoning load of the LLM across multiple simpler steps, and ensures better consistency in intermediate results shown in Table 2 in the Appendix section. This explains why the multi-step prompts achieve the highest F1, Exact Match, and Accuracy scores, while also narrowing the gap between the general-purpose and reasoning-optimized models.

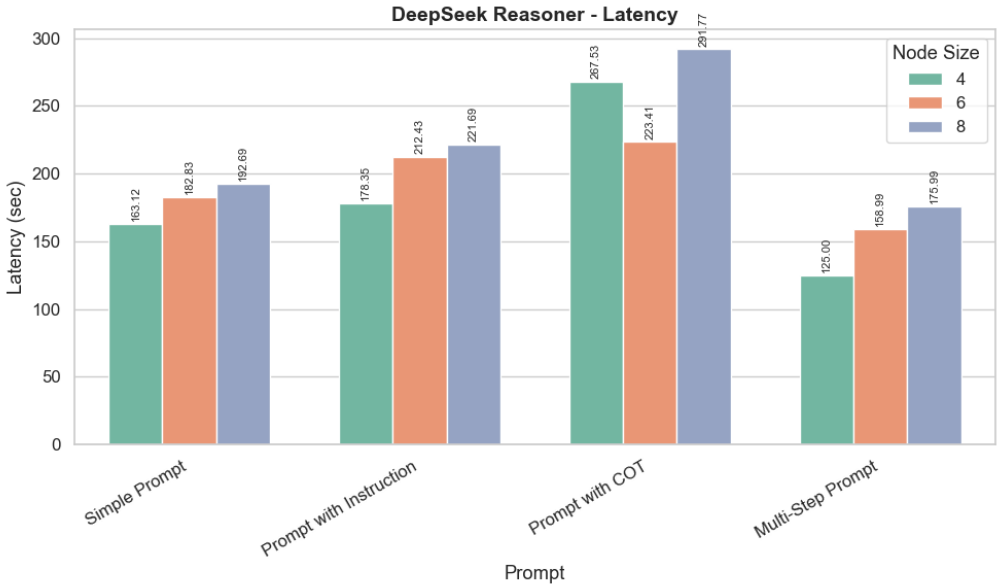
4.4 LLM Metrics Across Prompts

This section evaluates how different prompting strategies influence the computational behavior of the two models we have used in this study. Specifically, we analyze latency, performance (F1 score), and completion token usage across all the different prompting techniques. These metrics provide insight into the trade-offs between efficiency and accuracy, and highlight the key differences between speed-optimized models like GPT-4o-mini and reasoning-focused models like DeepSeek Reasoner.

4.4.1 Latency across prompting techniques:



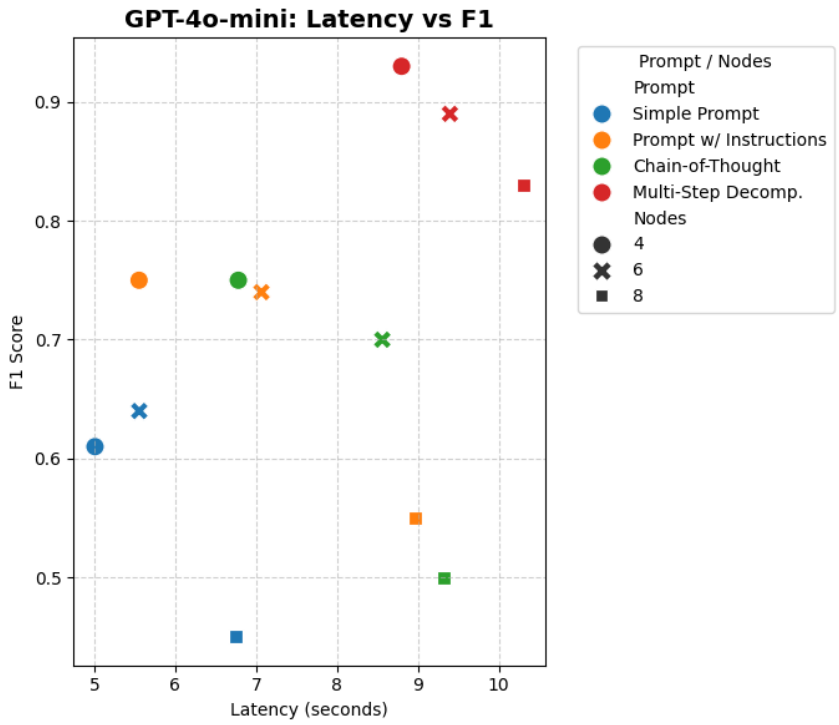
(a) Average Latency per request across all prompting techniques — GPT-4o-mini



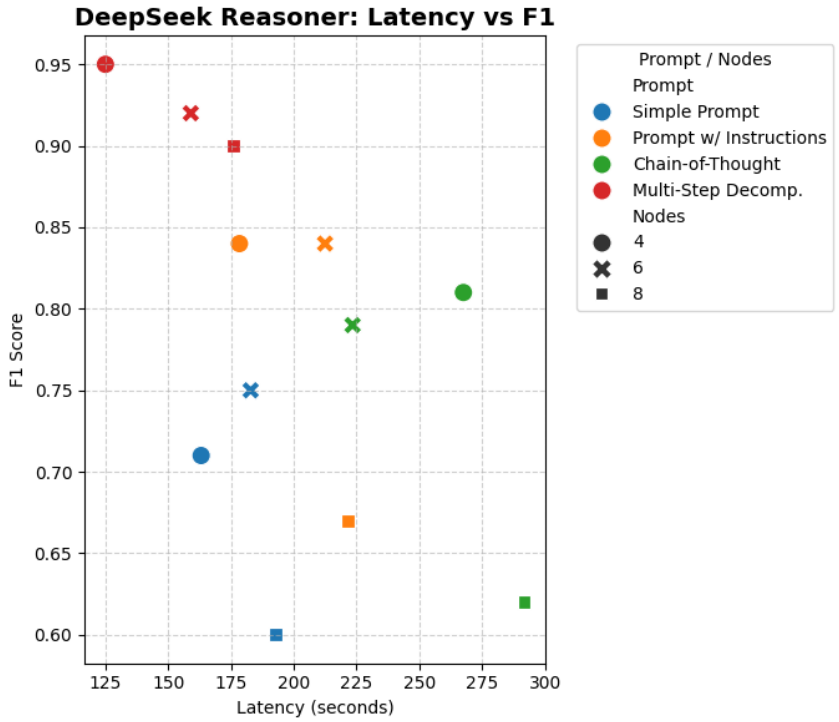
(b) Average Latency per request across all prompting techniques — DeepSeek Reasoner

Figure 2: Latency statistics for GPT-4o-mini and DeepSeek Reasoner across all prompting techniques (average per request).

4.4.2 Latency vs Performance



(a) F1 score vs Average Latency per request — GPT-4o-mini



(b) F1 score vs Average Latency per request — DeepSeek Reasoner

Figure 3: F1 score vs Average Latency for GPT-4o-mini and DeepSeek Reasoner across all prompting techniques (average per request).

Analysis:
The results highlight a clear trade-off between latency and performance across the two models. For **GPT-4o-mini**, latency remained relatively stable across prompting strategies, as seen in Figure 3(a). The difference between simple, instruction-based, chain-of-thought, and multi-step prompts was only about 1–2 seconds, with average response times ranging from 5 to 10 seconds across DAG sizes. This shows that GPT-4o-mini's average latency per request is largely unaffected by the complexity of the given task, and is instead more directly influenced by prompt length. Prompts containing larger DAGs (8 nodes) took longer than prompts with smaller DAGs (4 nodes) by ≈1-2 seconds. This behavior shows GPT-4o-mini's optimization for faster inference, where latency is minimally impacted by reasoning depth. Importantly, despite the stable response time, the multi-step decomposition strategy still gave a substantial performance boost, achieving F1 scores as high as 0.93 and accuracies above 0.90. This shows that decomposition significantly improved correctness, without imposing any substantial latency cost.

In contrast, **DeepSeek Reasoner** showed a markedly different latency profile as seen in Figure 3(b). Unlike GPT-4o-mini, where latency differences across prompts were minimal, DeepSeek's response time was highly sensitive to the prompt complexity. Simple and instruction-based prompts already required 160–220 seconds, and chain-of-thought peaked above 250 seconds, showing that more complex prompts imposed a substantial computational burden. Interestingly, the multi-step decomposition strategy resulted in the lowest latency of all prompting techniques (≈120–150 seconds across DAG sizes), despite requiring three separate queries.

This highlighted a key distinction: as a reasoning-focused LLM, DeepSeek Reasoner generates large amounts of intermediate reasoning text as mentioned in [DeepSeek], which increases the inference time. This can be seen more clearly in the next section. However, when the task is decomposed into simpler subtasks, the reasoning load per query is reduced, helping the model to deliver responses faster and more efficiently. Importantly, this efficiency gain came without any compromise in performance. Multi-step decomposition delivered the highest overall results, with F1 scores between 0.90 and 0.95 and accuracies up to 0.99 across DAG sizes.
Taken together, these results highlight an important insight:

- For speed-focused models (GPT-4o-mini), decomposition slightly increases latency but significantly improves accuracy.
- For reasoning-focused models (DeepSeek Reasoner), decomposition simultaneously reduces latency and improves accuracy, demonstrating that structured multi-step prompting is computationally efficient and also yields better performance.

4.4.3 Completion Tokens across Prompting techniques:

Analysis of Completion Tokens

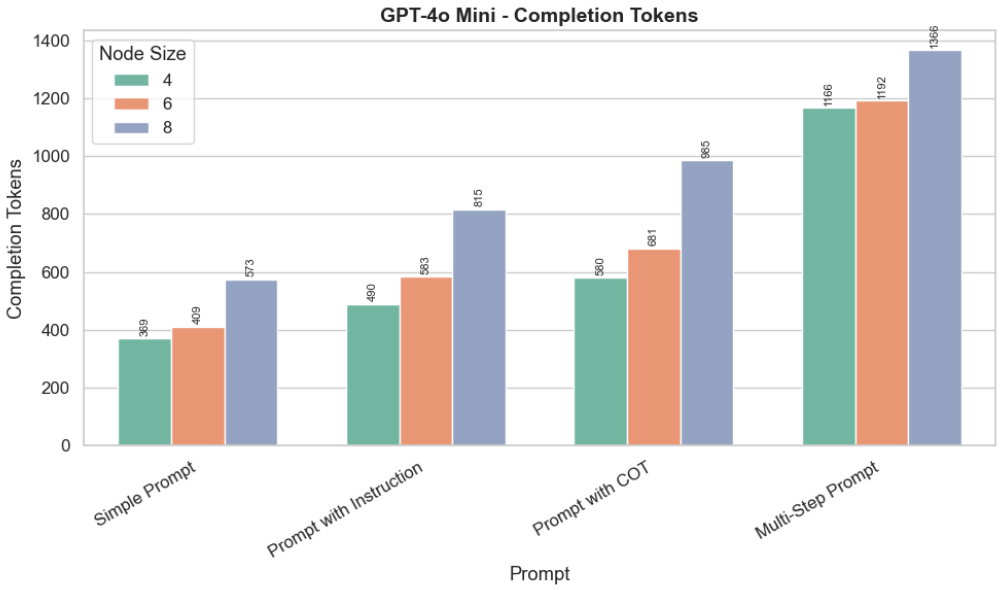
Completion token usage gives us additional insight into how different prompting strategies affect the computational load of the models (Figure 4).

For **GPT-4o-mini**, the number of completion tokens on average per request, scaled predictably with prompt type and length as seen in Figure 4(a). Simple prompts required approximately 370–570 tokens, while instruction-based prompts consumed around 490–815 tokens. Chain-of-thought prompting further increased the output size to nearly 1,000 tokens for 8-node DAGs. Multi-step decomposition had the largest token counts overall, since three separate prompts were used and aggregated across all steps, totals exceeded 1,200 tokens for larger DAGs per request on average. Importantly, the differences between prompting strategies here are primarily attributable to prompt length instead of reasoning complexity which is consistent with GPT-4o-mini's optimization for efficient and quick responses.

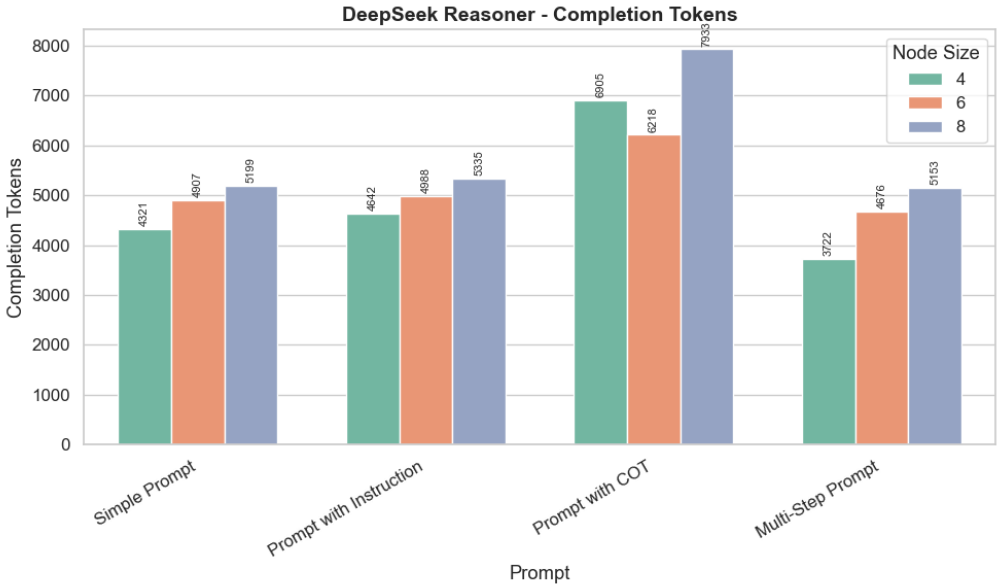
In contrast, **DeepSeek Reasoner** generated substantially larger completion tokens across all settings as seen in Figure 4(b). Even simple prompts required 4,000–5,000 tokens on average, with chain-of-thought prompts peaking between 6,000 and 8,000 tokens. This reflects DeepSeek's reasoning-focused design: as it includes both *reasoning_content* and final *content* in its responses which result in them being much longer compared to GPT-4o-mini. Interestingly, the multi-step decomposition strategy again demonstrated efficiency advantages similar to the latency metrics. Even though three separate prompts were issued, the combined completion length ($\approx$3,500–5,000 tokens) was comparatively lower than the single long outputs generated under chain-of-thought prompting. This showed that decomposing the task into smaller subtasks not only reduced latency (as discussed earlier) but also restricted response length, leading to a more efficient use of tokens.

Taken together, these results reinforce the difference between the two models:

- For **GPT-4o-mini**, completion token counts remain relatively low and depend mainly on prompt length over reasoning complexity.
- For **DeepSeek Reasoner**, completion token counts are substantially higher due to the model's reasoning traces, but decomposition mitigates this overhead by distributing reasoning load over simpler sub-tasks.



(a) Average Completion tokens per request across all prompting techniques — GPT-4o-mini



(b) Average Completion tokens per request across all prompting techniques — DeepSeek Reasoner

Figure 4: Completion token statistics for GPT-4o-mini and DeepSeek Reasoner across all prompting techniques (average per request).

5 Conclusion

This thesis studied the capability of large language models (LLMs) to identify canonical adjustment sets for causal inference tasks, with a specific focus on how prompting strategies influence performance. The study compared a general-purpose model (GPT-4o mini) and a reasoning-optimized model (DeepSeek Reasoner) across multiple prompting strategies: simple prompt, instruction-based prompt, chain-of-thought (CoT) prompt, and a multi-step decomposition approach. Experiments were conducted on directed acyclic graphs (DAGs) of different sizes and optimal edge densities. Baseline models, including fixed-length and distribution-based random set generation, were coded for reference performance.

The observed results show that the prompting strategy has a measurable impact on both accuracy and efficiency. In general, structured prompts that explicitly guide the model's reasoning process, such as instruction-enhanced and CoT prompts, achieved higher alignment with ground-truth adjustment sets as compared to the simple prompts. The Multi-Step decomposition approach, which distributes the reasoning load over sequential subtasks, gave the overall best results and also notably reduced average latency and limited token usage in later stages.

DeepSeek Reasoner provided more detailed, extensive outputs and superior reasoning, but it has high latency and uses more tokens. Conversely, GPT-4o mini is much faster and more efficient, though it lacks the reasoning depth required for higher accuracy. This trade-off shows that the best model depends on whether speed or reasoning is prioritized.

Across DAG sizes and edge probabilities, results showed that larger and denser graphs tend to increase task complexity, affecting both model accuracy and processing time.

In conclusion, this study showed that LLMs can be useful for causal reasoning tasks, but their effectiveness is sensitive to prompt design, reasoning strategy, and the underlying model architecture.

6 Limitations and Future Work

While this study showed the potential of large language models for causal reasoning, several limitations must be acknowledged. Firstly, the use of the Identification (ID) algorithm [Shpitser and Pearl, 2006] was not fully explored. The ID algorithm can derive identifying formulas in cases where no valid adjustment set exists, such as in front-door scenarios. Focusing only on adjustment sets restricts the generality of the findings. Second, we did not formulate causal inference tasks in natural language as done in benchmarks such as CLadder [Jin et al., 2023], where problems are expressed as text-based queries instead of graph-based prompts. This would allow for the testing of models in a setting that is closer to human reasoning tasks. Finally, the experimental DAGs were limited in size (up to eight nodes), which does not reflect the complexity of real-world causal structures such as those found in biological or economic systems. Larger graphs with more intricate dependencies remain an open challenge.

The results of this study show several promising directions for future research. The multi-step decomposition strategy can be extended by splitting the reasoning process into even smaller subtasks, further reducing per-step complexity and improving accuracy, depending fully on the task at hand. The reliability of the LLM outputs can be improved through self-consistency mechanisms [Wang et al., 2022], where multiple candidate solutions are generated and a consensus (or majority vote) is used to decide the final answer. A more adaptive approach can be explored in which the LLM itself is asked to evaluate the problem, propose a solution strategy, and generate the intermediate steps required to breakdown the problem. This helps the model to do self-directed reasoning and adjust its strategies based on the complexity of the problem. In line with recent perspectives on LLM evaluation [Chang et al., 2024], future work can explore self-evaluation as an important component where the model critiques and validates its own reasoning. This can serve as an internal benchmark to avoid errors. Finally, the prompting framework introduced here can be extended to other causal reasoning tasks beyond adjustment sets, such as causal discovery, mediation analysis, or counterfactual reasoning, where structured multi-step reasoning is equally critical.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Palaash Agrawal, Shavak Vasania, and Cheston Tan. Can llms perform structured graph reasoning tasks? In *International Conference on Pattern Recognition*, pages 287–308. Springer, 2025.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Tom B. Brown, Benjamin Mann, Nick Ryder, , et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1877–1901, 2020.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

Yining Chen, Jiaheng Wu, Yue Pan, Minje Chen, Guangkuo Li, Chenjie Deng, Chen Zhang, Yang Zhang, and Jianchao Wu. Calm: A large-scale causal language model benchmark. *arXiv preprint arXiv:2401.07720*, 2024.

Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, , et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Diego Colombo, Marloes H Maathuis, Markus Kalisch, and Thomas S Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321, 2012.

A Philip Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 41(1):1–15, 1979.

DeepSeek. Reasoning model. URL `https://api-docs.deepseek.com/guides/reasoning_model`.

DeepSeek-AI. DeepSeekMath: Pushing the limits of mathematical reasoning in open large language models. *arXiv preprint arXiv:2402.03300*, 2024.

Qingxiu Dong, Lei Dai, Juncheng Lin, , et al. A survey on in-context learning. *arXiv preprint arXiv:2210.05221*, 2022.

Guido W Imbens and Donald B Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, 2015.

Fan Jin, Jingbo Zhang, Hong Zhang, Weiyun Li, Qi Chen, Xu Han, Qi Shen, and Yuqiao Zhang. Cladder: A causal inference benchmark for large language models. *arXiv preprint arXiv:2311.12759*, 2023.

Karl G. Jöreskog. A general method for estimating a linear structural equation system. *Structural Equation Models in the Social Sciences*, pages 85–112, 1973.

Markus Kalisch, Martin Mächler, Diego Colombo, Marloes H Maathuis, and Peter Bühlmann. Causal inference using graphical models with the r package pcalg. *Journal of statistical software*, 47:1–26, 2012.

Takeshi Kojima, Shixiang Gu, Machel Xu, Vafa Nitisukul, , et al. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:25330–25345, 2022.

Oliver Kramer and Jill Baumann. Unlocking structured thinking in language models with cognitive prompting. *arXiv preprint arXiv:2410.02953*, 2024.

Patrick Lewis, Wen-tau Yih, Abhishek Goyal, , et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Pengfei Liu, Weizhe Yuan, Jin Fu, , et al. Pre-train, prompt, and predict: A systematic survey of prompting methods in language models. *ACM Computing Surveys*, 2023.

OpenAI. Gpt-4o mini. URL `https://platform.openai.com/docs/models/gpt-4o-mini`.

OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1988.

Judea Pearl. A probabilistic calculus of actions. In *Uncertainty in artificial intelligence*, pages 454–462. Elsevier, 1994.

Judea Pearl. Causal inference in statistics: An overview. 2009a.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009b.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017.

Donald B Rubin. Estimating causal effects of treatments in randomized and non-randomized studies. *Journal of Educational Psychology*, 66(5):688–701, 1974.

Ilya Shpitser and Judea Pearl. Identification of joint interventional distributions in recursive semi-markovian causal models. In *AAAI*, pages 1219–1226, 2006.

Peter Spirtes, Christopher Meek, and Thomas S. Richardson. An algorithm for causal inference in the presence of latent variables and selection bias. In Clark Glymour and Gregory F. Cooper, editors, *Computation, Causation and Discovery*, pages 211–252. MIT Press, 1999.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Thang Trivedi, Saining Zhou, Hong He, Zichao Liu, , et al. Solving olympiad geometry problems without human demonstrations. *Nature*, 625(7994):305–310, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Yi Tay, Rishi Bommasani, , et al. Chain of thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Sewall Wright. Correlation and causation. *Journal of Agricultural Research*, 20(7):557–585, 1921.

Tong Xiao and Jingbo Zhu. Foundations of large language models. *arXiv preprint arXiv:2501.09223*, 2025.

Edward Xie, Aleksander Rzepka, Yanda Du, , et al. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Machine Learning (ICML)*, 2022.

Fan Zhou, Kaiyuan Li, Jiaxin Liu, Meng Wu, Yanan Wang, Fan Wei, Zilin Peng, Dawei Yin, Bing Wu, and Bin Zhu. Causalbench: A benchmark for causal reasoning in large language models. *arXiv preprint arXiv:2403.01186*, 2024.

Kerui Zhu, Bo-Wei Huang, Bowen Jin, Yizhu Jiao, Ming Zhong, Kevin Chang, Shou-De Lin, and Jiawei Han. Investigating instruction tuning large language models on graphs. *arXiv preprint arXiv:2408.05457*, 2024.

A Prompt Templates

This appendix contains the exact prompt templates used in the experiments. All prompts include placeholders for dynamically inserting the DAG JSON, treatment variable, and outcome variable.

A.1 Simple Prompt

```
You are given a DAG with treatment node x and outcome node y.
Find the canonical valid adjustment set using the given input.

Input DAG:
{json.dumps(dag_json, indent=1)}
Treatment: {x}
Outcome: {y}

Output Format (Strict):

1) Reasoning:

2) Final Output:
Canonical Adjustment Set: ['X1', 'X2', ...]
# Or:
Canonical Adjustment Set: []
...
If no valid set exists, state: "No valid adjustment set exists."
```

A.2 Prompt with Instructions

```
You are given a DAG with treatment node x and outcome node y.
Find the canonical valid adjustment set using the following instructions:
• Identify descendants of X (De(X))
• Identify ancestors of Y excluding X (An(Y) \ X)
• Compute the forbidden set: descendants of nodes that are both in An(Y) and De(X)
• Exclude X and forbidden nodes from possible ancestors of Y
• If there is no valid adjustment set: return No valid adjustment set exists
• If adjustment set is empty, return Canonical Adjustment Set: []
• Else, list the canonical adjustment set

Input DAG:
{json.dumps(dag_json, indent=1)}
Treatment: {x}
Outcome: {y}

Output Format (Strict):

1) Reasoning:
```

```
2) Final Output:
Canonical Adjustment Set: ['X1', 'X2', ...]
# Or:
Canonical Adjustment Set: []
...
If no valid set exists, state: "No valid adjustment set exists."
```

A.3 Chain-of-Thought (CoT)

```
You are given a DAG with treatment node x and outcome node y.
Find the canonical valid adjustment set using the following instructions:
1) Identify descendants of X (De(X))
2) Identify ancestors of Y excluding X (An(Y) \ X)
3) Compute the forbidden set: descendants of nodes that are both in An(Y) and
   De(X)
4) Exclude X and forbidden nodes from possible ancestors of Y
5) If there is no valid adjustment set: return No valid adjustment set exists
6) If adjustment set is empty, return Canonical Adjustment Set: []
7) Else, list the canonical adjustment set

Give step-by-step reasoning to support your answer.

Example:
{
  "nodes": ["X","Y","Z","M","D","U"],
  "edges": [
    ["Z","X"], ["Z","Y"], # Z confounds X and Y
    ["U","X"], # U -> X only (instrument-like)
    ["X","M"], ["M","Y"], # X -> M -> Y (mediator M)
    ["M","D"] # D is a descendant of mediator
  ]
}

1) Reasoning:
- possDe(X): {M, D, Y}
- possAn(Y) \ {X}: {Z, M}
- Intersection: {M}
- Forbidden set (De of intersection): {M, D}
- Candidate Z*: {Z}
- Validation: passes (adjusting for Z blocks back-door paths; mediator M and
   its descendants are not adjusted for)

2) Final Output:
Canonical Adjustment Set: ['Z']

Input DAG:
{json.dumps(dag_json, indent=1)}
Treatment: {x}
```

```
Outcome: {y}

Output Format (Strict):

1) Reasoning:

2) Final Output:
Canonical Adjustment Set: ['X1', 'X2', ...]
# Or:
Canonical Adjustment Set: []
...
If no valid set exists, state: "No valid adjustment set exists."
```

A.4 Multi-Step Decomposition

A.4.1 Prompt 1:

```
Prompt 1:

    You are given a Directed Acyclic Graph (DAG) in JSON format.
    Each key is a node, and its value is a list of children it points to
        (i.e., **edges go from key to children**).

    Task:
    1. Identify all possible **descendants** of the treatment variable X
        (i.e., all nodes reachable by following outgoing edges from X). Do
        **not include X itself**.
    2. Identify all possible **ancestors** of the outcome variable Y (i.e.,
        all nodes from which there is a directed path to Y by following
        incoming edges recursively trace parents, then parents of parents, and
        so on). Do **not include Y itself**.
    3. Compute and return the **intersection** between these two sets.

    Instructions:
    - The DAG is defined as a dictionary where each key is a node, and the
        value is a list of its children.
    - If a node A has an entry like `"A": ["B"]`, this means there is a
        **directed edge from A to B**, i.e., A to B.
      - In this case: A is the **parent**, B is the **child**.
    - Only follow the directions given by the edges. Do **not** infer any
        reverse-direction edges.
    - **Do not confuse** the JSON structure for example, `"X1": ["X4"]` means
        **X1 points to X4** (X1 to X4), so **X1 is the parent**, and **X4 is
        the child**.

    Input:
    DAG (JSON): {json.dumps(dag_entry["dag"])}
    Treatment (X): {dag_entry["x"]}
    Outcome (Y): {dag_entry["y"]}
```

```
    Output Format (Strict):
    #Reasoning (step-by-step and concise)

    #Final Output:
    Possible descendants of X: [X1, X2, ...]
    Possible ancestors of Y: [Y1, Y2, ...]
    Intersection: [Z1, Z2, ...]
```

A.4.2 Prompt 2:

```
Prompt 2:
    You are given a Directed Acyclic Graph (DAG) represented in JSON,
    And the list of predicted intersection nodes (possAn(Y) & possDe(X)):

    Input:
    DAG (JSON): {json.dumps(dag_entry["dag"])}
    Predicted intersection nodes: {json.dumps(intersection_nodes)}

    Task:
    - Find all descendants (excluding themselves) of the predicted
        intersection nodes, using the DAG as reference.
    - If the predicted intersection nodes list is empty, simply output:
    Forbidden set: []

    Output Format (strict):
    #Reasoning: (concise)

    #Final Output:
    Forbidden set: [A, B, ...]
```

A.4.3 Prompt 3:

```
Prompt 3:
    You are given:
    - The set of all ancestors of Y.
    - The descendants of nodes that appear on proper causal paths from X to Y.
    - the treatment and outcome

    Input:
    Ancestors of Y: {ancestors}
    Descendants of causal path nodes: {descendants}
    Treatment (X): {x}
    Outcome (Y): {y}

    Task:
    1. Remove from the Ancestor Set:
       - the treatment X and outcome Y (IF PRESENT ONLY)
       - Descendants of causal path nodes
    2. The resulting set is the canonical adjustment set.
```

```
    Notes:
    - An empty canonical adjustment set ([]) is valid.

    Output Format (Strict):

    1) Reasoning (concise):

    2) Final Output:
    Canonical Adjustment Set: ['X1', 'X2', ...]
    # Or:
    Canonical Adjustment Set: []
```

B Example of LLM Errors:

```
    [Step 1: Identify Descendants of Treatment Node 'x' (X3)
To find the descendants of X3, we trace all directed paths starting from X3 to
    find all the nodes it can reach. Following the convention that a node is a
    descendant of itself, we start with X3.

X3 is the starting node.

The direct children of X3 are: X5, X6, X7, and X8.

Next, we find the children of those nodes:

X5 has no children.

X6 has one child: X7.

X7 has one child: X8.

X8 has no children.

Combining all nodes reachable from X3 (and including itself), we get the full
    set of descendants.

\{Thus, the set of descendants of X3 is {X3, X5, X6, X7, X8}.}

Step 2: Identify Ancestors of Outcome Node 'y' (X5)
To find the ancestors of X5, we trace all paths backward from X5. A node is
    also considered an ancestor of itself.

X5 is the starting node.

We search the DAG for nodes that have X5 as a child:

X2: ["X5", "X7"] : X2 is a parent.
```

X3: ["X5", "X6", "X7", "X8"] : X3 is a parent.

Now we find the ancestors of these parents (X2 and X3):

X2 has no parents.

X3 has no parents.

Combining the node itself with all its parents, grandparents, etc., gives us the full set of ancestors.

Thus, the set of ancestors of X5 is {X2, X3, X5}.

Step 3: Intersection of the Two Sets
Finally, we identify the nodes that are common to both the set of descendants of X3 and the set of ancestors of X5.

\{Descendants of X3: {X2, X5, X6, X7, X8}}

Ancestors of X5: {X2, X3, X5}

\{The common nodes in both sets are X2, X3 and X5.}

Therefore, the intersection of the descendants of X3 and the ancestors of X5 is {X2, X3, X5}.]

As seen in the above example - there is an inconsistency between the predicted ancestor set in step 1 and step 3, this ultimately causes an incorrect intersection set - further compounding errors.

C Additional Result Tables:

C.1 Evaluation Metrics Across DAG Sizes

4-Node DAGs		
	GPT-4o mini	DeepSeek
Simple Prompt		
Precision	0.48	0.59
Recall	0.86	0.90
F1 Score	0.61	0.71
Exact Match	0.37	0.48
Accuracy	0.49	0.60
Prompt with Instruction		
Precision	0.68	0.79
Recall	0.84	0.89
F1 Score	0.75	0.84
Exact Match	0.49	0.65
Accuracy	0.64	0.79
Chain of Thought		
Precision	0.60	0.71
Recall	0.89	0.91
F1 Score	0.75	0.81
Exact Match	0.55	0.72
Accuracy	0.61	0.81

6-Node DAGs

	GPT-4o mini	DeepSeek
Simple Prompt		
Precision	0.59	0.71
Recall	0.71	0.81
F1 Score	0.64	0.75
Exact Match	0.32	0.50
Accuracy	0.44	0.62
Prompt with Instruction		
Precision	0.69	0.80
Recall	0.80	0.89
F1 Score	0.74	0.84
Exact Match	0.41	0.59
Accuracy	0.59	0.71
Chain of Thought		
Precision	0.65	0.74
Recall	0.77	0.86
F1 Score	0.70	0.79
Exact Match	0.48	0.50
Accuracy	0.55	0.55

8-Node DAGs

	GPT-4o mini	DeepSeek
Simple Prompt		
Precision	0.37	0.52
Recall	0.59	0.70
F1 Score	0.45	0.60
Exact Match	0.29	0.48
Accuracy	0.39	0.59
Prompt with Instruction		
Precision	0.47	0.59
Recall	0.69	0.81
F1 Score	0.55	0.67
Exact Match	0.37	0.53
Accuracy	0.51	0.60
Chain of Thought		
Precision	0.41	0.53
Recall	0.66	0.76
F1 Score	0.50	0.62
Exact Match	0.40	0.52
Accuracy	0.44	0.57

C.2 Multi-Step Decomposition Prompt - Breakdown Stats:

Model	Metric	4-node DAGs			6-node DAGs			8-node DAGs		
		P1	P2	P3	P1	P2	P3	P1	P2	P3
GPT-4o-mini	Precision	0.92	1.00	1.00	0.83	0.97	1.00	0.78	0.94	1.00
	Recall	0.89	1.00	0.99	0.80	0.99	0.96	0.80	0.99	0.94
	F1 Score	0.91	1.00	0.99	0.81	0.98	0.98	0.79	0.96	0.96
DeepSeek	Precision	0.95	1.00	1.00	0.95	1.00	1.00	0.90	1.00	1.00
	Recall	0.99	1.00	1.00	0.95	1.00	1.00	0.91	1.00	1.00
	F1 Score	0.97	1.00	1.00	0.95	1.00	1.00	0.90	1.00	1.00

Table 2: Breakdown of individual steps in the Multi-Step Decomposition Prompt, where P1, P2 and P3 are Prompts 1, 2 and 3 respectively