Causal Bounds in Quasi-Markovian Graphs

Madhumitha Shridharan [1] **Garud Iyengar** [1]

Abstract

We consider the problem of computing bounds for causal queries on quasi-Markovian graphs with unobserved confounders and discrete valued observed variables, where identifiability does not hold. Existing non-parametric approaches for computing such bounds use multilinear programming (MP) formulations that are often intractable for existing solvers when the degree of the polynomial objective is greater than two. Hence, one often has to resort to either fast approximate heuristics which are not guaranteed to contain the true query value, or more accurate but computationally intensive procedures. We show how to construct an equivalent MP with a polynomial objective of lower degree. In particular, the degree of the objective in the new MP is equal to *only* the number of C-components that are intervened upon, instead of the *total* number of C-components. As a result, we can compute exact bounds for significantly larger causal inference problems as compared to what is possible using existing techniques. We also propose a very efficient Frank-Wolfe heuristic that produces very high quality bounds, and scales to large multilinear problems of higher degree.

1. Introduction

Many practical applications of causal inference involve variables that are important for the identification of causal effects, but are either unknown or unmeasured, i.e. *unobserved* confounders (Imbens & Rubin, 2015; Pearl, 2009). It is impossible to accurately identify causal effects in the presence of unobserved confounders. However, it is possible to obtain bounds on such effects.

There have been multiple such attempts to bound causal effects in small graphs with special properties. To bound the causal effect of a treatment on a target variable, Richardson et al. (2014) invoke additional (untestable) assumptions and assess how inference about the treatment effect changes as these assumptions are modified. In small graphs where any two observed variables are neither adjacent in the graph, nor share a latent parent, Evans (2012) present a graphical approach to bound causal effects. Geiger & Meek (2013) leverage Tarski's procedure for quantifier elimination in the first order theory of real numbers to bound causal effects in a model under specific parametric assumptions. Kilbertus et al. (2020) and Zhang & Bareinboim (2021) develop procedures for computing causal bounds in extensions of the instrumental variable model to the continuous setting. Finkelstein & Shpitser (2020) develop a method for obtaining bounds on causal queries using probability rules and the conditions on counterfactuals implied by causal graphs.

While fewer in number, there have also been attempts to bound causal effects in large general graphs. Poderini et al. (2020) propose a technique, originally developed in the field of quantum foundations, to compute bounds in large graphs with multiple instruments and observed variables. Finkelstein et al. (2021) propose a method for partial identification in a class of measurement error models. Sachs et al. (2020; 2022) identify a large problem class for which linear programming (LP) can be used to compute causal bounds, and develop an algorithm for formulating the objective function and the constraints of the corresponding LP. However, such LP formulations quickly become intractable for existing solvers because the size of the LP grows exponentially in the number of edges in the underlying causal graph. Shridharan & Iyengar (2022) extend this work by showing that this LP can be significantly pruned by carefully considering the structure of the causal query, and show how to compute bounds for significantly larger causal inference problems as compared to what is possible using existing techniques. Zhang et al. (2022) and Duarte et al. (2021) propose general polynomial programming based approaches to solve general causal inference problems, but their procedures are either computationally intensive, or are complex to implement.

In this work, we extend the class of large graphs for which causal effects can be efficiently bounded. In particular, we focus on a class of causal graphs known as quasi-markovian causal graphs, where every variable has only a single unob-

[1]Department of Industrial Engineering and Operations Research, Columbia University, New York, USA. Correspondence to: Madhumitha Shridharan <ms6143@columbia.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

served confounder as parent. Although this assumption may seem restrictive, we choose to focus on this setting for the following reasons:

- Quasi-Markovian models, both large and small, are applicable in multiple important applications: the family of graphs which generalize the instrumental variable setting, see e.g. (Sachs et al., 2022), the well-studied setting where multiple confounded treatments influence an outcome (Ranganath & Perotte, 2019; Janzing & Schölkopf, 2018; D'Amour, 2019; Tran & Blei, 2017) and Markov Decision Processes (Kallus & Zhou, 2020).
- Removing the quasi-Markovian assumption puts us in the general setting with discrete variables which requires polynomial programming (Zhang et al., 2022; Duarte et al., 2021). As shown in our experiments in Section 5.2.2, existing algorithms for the general setting like that in Duarte et al. (2021) are computationally inefficient even for small graphs. These problems are indeed very challenging to solve, and developing efficient algorithms for this setting is a promising direction of future research. Developing better heuristics for the quasi-Markovian case as we have done is a step in this direction.

Similar to Zaffalon et al. (2020a), we assume that the input data given is such that bounds for causal queries on the quasi-markovian graph can be computed using disjoint multilinear programming i.e. optimization problems with a polynomial objective and separable linear constraints (De Campos et al., 1994; Zhang et al., 2022; de Campos & Cozman, 2004). The construction of the multilinear program relies on a type of clustering of variables in the causal graph, known as *confounded components*, or C-components (Tian & Pearl, 2002). In particular, the degree of the polynomial objective is the number of C-components in the causal graph. However, even the disjoint bilinear programming problem is NP-hard in general, and NP-hard to even approximate within any finite factor (Chen et al., 2009; Housni et al., 2022). Hence, techniques which leverage problem structure to reduce the complexity of the problem, and efficient heuristics to obtain effective approximate solutions are critical in our context.

Recently, Zaffalon et al. (2020a) showed that causal bounds can be obtained in small quasi-markovian graphs by applying variable elimination in credal networks. In their experiments on large quasi-markovian graphs, they apply the ApproxLP algorithm on the multilinear programming formulations to obtain approximate solutions (Antonucci et al., 2015). However, these bounds are non-exact, even for graphs with only one variable in the intervention set. In this work, we show that exact bounds for queries in quasi-markovian graphs can be computed by solving simpler multilinear programs where the degree of the multilinear objective function is equal to the number of C-components that are intervened upon, not the *total* number of C-components in the causal graph. In particular, bounds for causal queries where only one C-component is intervened upon (like the examples in Zaffalon et al. (2020a)) can, in fact, be computed *exactly* by formulating and solving an appropriate linear program. Similarly, bounds for causal queries where two C-components are intervened upon can be computed exactly by formulating and solving an appropriate quadratic program via readily available solvers.

For bounds of causal queries where three or more C-components are intervened upon, de Campos & Cozman (2004) transformed the multilinear program into a quadratic program by grouping terms and introducing new variables and equalities. However, such quadratic programs are still innately complex and cannot be solved by off-the-shelf solvers in reasonable time frames. Several variations of branch-and-bound techniques have been proposed as well; unfortunately they all suffer from serious memory and computational efficiency issues unless the multilinear program is very simple (Duarte et al., 2021; de Campos & Cozman, 2004; Cano et al., 2007).

Approaches like that in (Zhang et al., 2022) compute approximate bounds by avoiding solving the optimization problem altogether. They posit a model for the data generating process of the causal graph and use Collapsed Gibbs Sampling to sample from the posterior of the query given observation samples. However, the collapsed Gibbs Sampling procedure proposed is prone to model misspecification, is complex to implement in practice, and requires computation of several complete conditionals and the development of methods to sample from them. Furthermore, extensive empirical evaluations are missing, so it is unclear how accurate the approximation is on different sets of samples or how long Gibbs Sampling takes to converge in practice.

Hence, novel heuristics which exploit problem structure are required for higher degree multilinear programs. Utilizing a previously established result regarding vertex optimality in multilinear programs, we develop a heuristic based on the Frank Wolfe algorithm which is both fast and valid, always containing the true query value in empirical experiments. In each iteration of the heuristic, we decompose the multilinear program into multiple smaller linear programs which can be solved in parallel. Furthermore, each of these smaller linear programs can be further pruned by applying techniques in (Shridharan & Iyengar, 2022). As a consequence, we significantly increase the size of quasi-markovian graphs for which the multilinear program method remains tractable.

Our heuristic is perhaps most similar to the ApproxLP algorithm proposed in (Antonucci et al., 2015). This algorithm iteratively minimizes blocks of variables in the multilin-

ear program while holding others fixed until convergence, which is equivalent to a popular heuristic in nonlinear programming known as Block Coordinate Descent (BCD) (Lyu, 2020). Antonucci et al. (2015) discuss two methods to select the block of variables to minimize in each iteration. The first selects the first block (according to some random order) which improves the objective (Random ApproxLP), while the second greedily selects the block which leads to the best improvement in objective value (Greedy ApproxLP). However, convergence to a stationary point is not guaranteed in either version of BCD, and there are examples known to cycle (Powell, 1973). On the other hand, our heurtistic has better theoretical guarantees and is guaranteed to converge to a stationary point, including local minimums. Our heuristic consistently computes valid bounds i.e. the bounds always contain the true value of the query. While the ApproxLP heuristic updates only a single block of variables in each iteration, multiple blocks of variables are simultaneously updated in our heuristic. Hence, we show that on average, our heuristic converges close to the bounds computed by both variations of ApproxLP for all examples, but is significantly faster than both. The difference in speed between these algorithms is most pronounced for large graphs.

To summarize, our main contributions are:

(i) In Section 4, we show that the multilinear program for computing bounds in quasi-markovian graphs can be reduced to a simpler multilinear program with a polynomial objective of lower degree. In particular, the degree of the objective in the new multilinear program is equal to only the number of C-components which are intervened upon in the causal inference problem, instead of the total number of C-components. As a result, any causal inference problem on a quasi-Markovian graph with at most two variables in the intervention set can be solved via a simple linear or quadratic program using readily available optimization solvers like Gurobi (Gurobi Optimization, LLC, 2023).

(ii) In Section 5, we present a method to obtain causal bounds when three or more C-components are intervened upon, and the simplified multilinear program has a polynomial objective of degree at least three. In Section 5.1, we utilize a previous result on vertex optimality in multilinear programs to develop a heuristic based on the Frank Wolfe algorithm which is fast, scalable to large graphs, always contains the true query value in empirical experiments and is guaranteed to converge to a stationary point. Furthermore, while the ApproxLP heuristic updates only a single block of variables in each iteration, multiple blocks of variables are simultaneously updated in our heuristic. In Sections 5.2.2 and 5.2.1, we benchmark our heuristic against the most recent alternatives proposed in literature to our knowledge: both variations of Approx LP and the branch-and-bound algorithm in Duarte et al. (2021). On average, our heuristic converges to bounds close to that of both versions of ApproxLP for all examples, but is significantly faster, especially for large graphs. Furthermore, our heuristic outperforms our best-case interpretation of the algorithm presented in Duarte et al. (2021) in that it computes bounds of much higher quality in the same time.

The organization of the rest of this paper is as follows. In Sections 2 and 3 we introduce the formalism in Zaffalon et al. (2020a;b). In Section 4 we introduce our main structural result for reducing the degree of the polynomial objective in the multilinear program. In Section 5 we present the Frank Wolfe heuristic and benchmark it against the most recent alternatives proposed in literature to our knowledge. Section 6 discusses possible extensions.

2. Causal Inference Problems in Quasi-Markovian Graphs

Let G denote the causal graph. Let $V_1, \ldots, V_n$ denote the variables in G in topologically sorted order, and $N = \{1, \ldots, n\}$ denote the set of indices of the variables. We assume that each variable $V_i \in \{0, 1\}$. Later in Section 3 we discuss why our proposed techniques automatically apply to the case where V_i takes discrete values. We use lower case letters for the values of the variables, and the notation $V_i = v_i$ denotes that the value of variable V_i is $v_i \in \{0, 1\}$. For any subset of indices $S \subseteq N$, we define $V_S := \{V_i : i \in S\}$. The notation $V_S = v_S$ denotes that the variable $V_i = v_i$, for all $i \in S$, for some $v \in \{0, 1\}^{|N|}$. Let $U_1, \ldots, U_m$ denote the mutually independent unobserved confounders in G. We consider the following class of causal graphs, known as *Quasi-Markovian* causal graphs. (Zaffalon et al., 2020a;b).

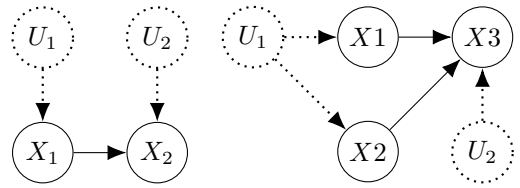
Definition 2.1 (Quasi-Markovian Causal Graphs (Zaffalon et al., 2020a))**.** For $i \in N$, let $pa(V_i)$ denote all parents of V_i in the causal graph, and let $\overline{pa}(V_i)$ denote only parents which are observed. A Quasi-Markovian Causal Graph is a causal graph where each variable has only a single unobserved confounder as parent. That is, for every $i \in N$,

$$pa(V_i) = (U_k, \overline{pa}(V_i))$$

where $k \in \{1, \ldots, m\}$.

We present examples of quasi-markovian causal graphs in Figure 1 (Zaffalon et al., 2020a).

Our framework relies on a type of clustering of observed variables in the causal graph, known as *confounded components* (Tian & Pearl, 2002). Let a bi-directed arrow



(a) (b)

Figure 1: Examples of Quasi-Markovian Causal Graphs

$V_i \leftrightarrow V_j$ between variables V_i, V_j represent the sequence $V_i \leftarrow U_k \rightarrow V_j$ where U_k is a unobserved confounder parent shared by V_i, V_j. A consequetive sequence of bi-directed arrows is known as a bi-directed path.

Definition 2.2 (Confounded Components (Tian & Pearl, 2002))**.** For a causal graph G, a subset $C \subseteq N$ is a C-component if any pair $V_i, V_j, i, j \in C$ is connected by a bi-directed path in G.

A C-component C is maximal if there does not exist any other C-component in the causal diagram G containing C. Let $C_1, \dots C_m \subseteq \{1 \dots n\}$ denote the maximal C-components in G corresponding to $U_1, \dots U_m$ i.e. C_k denotes the indices of the observed children of U_k in G.

Like Zaffalon et al. (2020a), we assume that the input data is of form $\mathbb{P}(V_{C_k}|\overline{pa}(V_{C_k}))$. Our goal is to compute bounds for the causal query $\mathcal{Q} = \mathbb{P}(V_O = q_O|do(V_I = q_I))$, i.e. the probability of the output event $V_O = q_O$ given an intervention $do(V_I = q_I)$. We show how to use this data to construct an optimization problem with a polynomial objective and linear constraints (i.e. a *multilinear program*) to compute causal bounds for the query.

3. Formulating the multilinear program

Each unobserved confounder U_k can potentially be very high dimensional with an unknown structure and dimension. As in (Balke & Pearl, 1994), we can circumvent this difficulty by modeling the *impact* of the confounder rather than the confounder directly via *response function variables*. For $i \in C_k$ the variable $V_i = f(\overline{pa}(V_i), U_k)$ for some unknown but fixed function $f : \{0,1\}^{|\overline{pa}(V_i)| \times U_k} \mapsto \{0,1\}$. Therefore, the confounder U_k impacts the relationship between $\overline{pa}(V_i)$ and V_i by selecting a function $f_i \in \mathcal{F}_i = \{f : f \text{ is a function from } \overline{pa}(V_i) \mapsto V_i\}$. Since each variable $V_j \in \overline{pa}(V_i)$ takes values in $\{0,1\}$ and $V_i \in \{0,1\}$, the cardinality of the set $|\mathcal{F}_i| = 2^{2^{|\overline{pa}(V_i)|}}$. Therefore, the elements of $\mathcal{F}_i$ can be indexed by $r_{V_i} \in R_{V_i} = \{1, \dots, |\mathcal{F}_i|\}$ i.e. $f_{r_{V_i}}$ denotes the r_{V_i}-th function from $\mathcal{F}_i$.

Let the set $R_{C_k} = \prod_{i \in C_k} R_{V_i}$ index all possible mappings from $\overline{pa}(V_i) \mapsto V_i$ for all $i \in C_k$. Thus, the response function variable $r_{C_k} \in R_{C_k}$ can be used to model the impact of U_k by denoting the function from all possible functions mapping $\overline{pa}(V_{C_k})$ to V_{C_k} selected by U_k. Hence, the unknown distribution over the high dimensional U_k can be equivalently modeled via the distribution $\boldsymbol{q}_{r_{C_k}} \in \mathbb{R}_+^{|R_{C_k}|}$ over the set R_{C_k}.

Note that although we work with causal graphs with binary variables in this paper, generalizing all subsequent results to categorical variables is straightforward. The critical property that we exploit is that the set of response function variables index possible mappings between variables. Therefore, our approach can be extended to general categorical variables by suitably defining response function variables. For example, suppose in Figure 1(a), $X_1, X_2 \in \{0, \dots, m\}$. Then the response function variable $r_{X_2} \in \{0, \dots, m^m - 1\}$. All subsequent results generalize.

Note that $\overline{pa}(V_{C_k})$ and r_{C_k} uniquely determine the values of variables V_{C_k}. For $r_{C_k} \in R_{C_k}$, let $F_T^k(V_S = v_S, r_{C_k})$ denote the value of $V_T \subseteq V_{C_k}$ when $V_S = v_S$ provided it is well defined. For $S \subseteq N$, let $P_S \subseteq N$ denote the indices of $\overline{pa}(V_S)$. As discussed, setting $\overline{pa}(V_{C_k}) = v_{P_{C_k}}$ and choosing $r_{C_k} \in R_{C_k}$ completely defines the values for V_{C_k}, i.e. $F_{C_k}^k(\overline{pa}(V_{C_k}) = v_{P_{C_k}}, r_{C_k})$ is well defined.

Let $R_{v_{C_k}.v_{P_{C_k}}} := \{r_{C_k} : F_{C_k}^k(V_{P_{C_k}} = v_{P_{C_k}}, r_{C_k}) = v_{C_k}\}$ and $p_{v_{C_k}.v_{P_{C_k}}} := \mathbb{P}(V_{C_k} = v_{C_k}|V_{P_{C_k}} = v_{P_{C_k}})$. Then, for each unobserved confounder U_k, we have

$$p_{v_{C_k}.v_{P_{C_k}}} = \sum_{r_{C_k} \in R_{v_{C_k}.v_{P_{C_k}}}} \boldsymbol{q}_{r_{C_k}}$$

Let $R = \Pi_{k=1}^m R_{C_k}$. For $r \in R$, let $F_T(V_S = v_S, r)$ denote the value of $V_T \subseteq V_N$ when $V_S = v_S$ provided it is well defined. The set

$$R_{\mathcal{Q}} = \{r \in R : F_O(V_I = q_I, r) = q_O\} \tag{1}$$

denotes the set of r values consistent with the query $\mathcal{Q} = \mathbb{P}(V_O = q_O|do(V_I = q_I))$. Hence,

$$\begin{aligned} \mathbb{P}(V_O = q_O|do(V_I = q_I)) &= \sum_{r \in R_{\mathcal{Q}}} \boldsymbol{q}_r \\ &= \sum_{r \in R_{\mathcal{Q}}} \Pi_{k=1}^m \boldsymbol{q}_{r_{C_k}} \end{aligned} \tag{2}$$

where (2) follows from mutual independence of unobserved confounders.

Lower and upper bounds for the causal query can then be obtained by solving the following pair of multilinear programs with polynomial objectives and linear constraints:

$$\begin{aligned} \min / \max_{\boldsymbol{q}} \quad & g(\boldsymbol{q}) := \textstyle\sum_{r \in R_{\mathcal{Q}}} \Pi_{k=1}^m \boldsymbol{q}_{r_{C_k}} \\ \text{s.t.} \quad & \textstyle\sum_{r_{C_k} \in R_{v_{C_k}.v_{P_{C_k}}}} \boldsymbol{q}_{r_{C_k}} \\ & = p_{v_{C_k}.v_{P_{C_k}}}, \forall k, v_{P_{C_k}}, v_{C_k} \\ & \boldsymbol{q} \geq 0. \end{aligned} \tag{3}$$

Fix $k \in \{1, \ldots, m\}$. Recall for $V_{P_{C_k}} = v_{P_{C_k}}$, $r_{C_k} \in R_{C_k}$ uniquely determines the value of V_{C_k}. Hence, for fixed $v_{P_{C_k}}$, $\cup_{v_{C_k}} R_{v_{C_k}.v_{P_{C_k}}}$ is a partition of R_{C_k}. Thus, the constraint $\sum_{r_{C_k} \in R_{C_k}} \boldsymbol{q}_{r_{C_k}} = \sum_{v_{C_k}} \sum_{r_{C_k} \in R_{v_{C_k}.v_{P_{C_k}}}} \boldsymbol{q}_{r_{C_k}} = \sum_{v_{C_k}} p_{v_{C_k}.v_{P_{C_k}}} = 1$ is implied by the other constraints in the multilinear program, and therefore, is not explicitly added to the multilinear program.

In Appendix C, we present the intuition behind response function variables and the formulation of the multilinear program via an example.

4. Simplifying the Multilinear Program

The degree of the polynomial objective in (3) is the number of C-components in the causal graph. The main result in this section is Theorem 4.1, which shows that this multilinear program is equivalent to a simpler one with polynomial objective of the same degree as the number of C-components which are intervened upon in the causal inference problem.

Theorem 4.1 (Simplifying the Multilinear Program)**.** *Consider a causal inference problem with query $\mathbb{P}(V_O|do(V_I))$ on a quasi-markovian graph with m C-components. Let ℓ be the number of C-components which contains at least one variable in the intervention set I. Then, the problem of computing bounds is an optimization problem with linear constraints involving the given ℓ C-components and an objective that is a polynomial of degree ℓ.*

Proof Sketch. Let $Z = N \setminus (I \cup O)$ i.e. the indices of variables in the graph which are neither intervened upon nor observed in the query. Therefore $N = Z \cup O \cup I$ is a partition of N. We can rewrite the query as

$$\mathbb{P}\big(V_O \mid do(V_I)\big) = \sum_{v_Z} \mathbb{P}\big(V_O, V_Z = v_Z \mid do(V_I)\big)$$

For fixed v_Z, we show via induction that

$$\mathbb{P}\big(V_O, V_Z = v_Z \mid do(V_I)\big) = \prod_{k=1}^{m} \mathbb{P}\left(\begin{array}{l|l} V_{O\cap C_k}, & do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), \\ V_{Z\cap C_k} = v_{Z\cap C_k} & V_{Z\cap\overline{pa}(V_{C_k})} = v_{Z\cap\overline{pa}(V_{C_k})}, \\ & V_{O\cap\overline{pa}(V_{C_k})} \end{array} \right) \quad (4)$$

The result is a special case of the C-component factorization result first shown in (Tian, 2002).

Suppose $V_{I\cap C_k} = \emptyset$ i.e. no variable in the k-th C-component is in the intervention set. Then, it follows that

$$\begin{aligned} (V_{O\cap C_k}, V_{Z\cap C_k}) &= V_{C_k}, \\ (V_{I\cap\overline{pa}(V_{C_k})}, V_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})}) &= V_{\overline{pa}(V_{C_k})}. \end{aligned}$$

Therefore, the term

$$\mathbb{P}\left(V_{O\cap C_k}, V_{Z\cap C_k} \,\middle|\, \begin{array}{l} do(V_{I\cap\overline{pa}(V_{C_k})}), \\ V_{Z\cap\overline{pa}(V_{C_k})}, \\ V_{O\cap\overline{pa}(V_{C_k})} \end{array} \right) = \mathbb{P}(V_{C_k}|V_{\overline{pa}(V_{C_k})})$$

is a constant completely defined by the data.

Fix $k \in \{1, \ldots, m\}$ such that $V_{I\cap C_k} \neq \emptyset$. Consider every term of form

$$\mathbb{P}\left(V_{O\cap C_k}, V_{Z\cap C_k} \,\middle|\, \begin{array}{l} do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), \\ V_{Z\cap\overline{pa}(V_{C_k})}, \\ V_{O\cap\overline{pa}(V_{C_k})} \end{array} \right) \quad (5)$$

We can rewrite this term as:

$$\begin{aligned} &\mathbb{P}\left(V_{O\cap C_k}, V_{Z\cap C_k} \,\middle|\, \begin{array}{l} do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), \\ V_{Z\cap\overline{pa}(V_{C_k})}, \\ V_{O\cap\overline{pa}(V_{C_k})} \end{array} \right) \\ &= \sum_{r_{C_k}} \mathbb{P}\left(\begin{array}{l|l} V_{O\cap C_k}, & do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), \\ V_{Z\cap C_k} & V_{Z\cap\overline{pa}(V_{C_k})}, \\ & V_{O\cap\overline{pa}(V_{C_k})}, r_{C_k} \end{array} \right) \mathbb{P}(r_{C_k}) \\ &= \sum_{r_{C_k}} \mathbb{P}\left(\begin{array}{l|l} V_{O\cap C_k}, & do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), \\ V_{Z\cap C_k} & V_{Z\cap\overline{pa}(V_{C_k})}, \\ & V_{O\cap\overline{pa}(V_{C_k})}, r_{C_k} \end{array} \right) q_{r_{C_k}} \end{aligned}$$

Fix r_{C_k}. Every variable in $\overline{pa}(V_{C_k})$ is either in I, Z or O. Hence, in the term

$$\mathbb{P}\left(V_{O\cap C_k}, V_{Z\cap C_k} \,\middle|\, \begin{array}{l} do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), \\ V_{Z\cap\overline{pa}(V_{C_k})}, \\ V_{O\cap\overline{pa}(V_{C_k})}, r_{C_k} \end{array} \right)$$

every variable in $\overline{pa}(V_{C_k})$ is either intervened upon or conditioned on. Since r_{C_k} is also conditioned on, each term of this form is a deterministic constant. Hence, (5) is linear in $q_{r_{C_k}}$. Now, the result follows from (4) and the two observations above.

□

Corollary 4.2. *The following results follow from Theorem 4.1:*

(a) A causal inference problem with only one variable in the intervention set can be solved by a linear program with linear constraints involving only the C-component containing that variable.

(b) A causal inference problem with two variables in the intervention set can be solved either by a linear program (if both are in the same C-component), or a quadratic program (if both are in different C-components).

The impact of Corollary 4.2 is that any causal inference problem on a quasi-markovian graph can be solved via a simple linear or quadratic program using readily available optimization solvers like Gurobi (Gurobi Optimization, LLC, 2023), as long as it contains at most two variables in the intervention set.

5. Higher Degree Multilinear Programs

We now present methods to obtain causal bounds when three or more C-components are intervened upon, and the simplified multilinear program has a polynomial objective of degree at least three. It has been previously established that there always exists an extreme point of the constraint polyhedron which is optimal in the multilinear program, as implied by Lemma 5.1 and Theorem 5.2 (see Appendix for proofs).

Lemma 5.1 ((Falk & Hoffman, 1976))**.** *The minimum value of a real-valued continuous concave function $f(x)$ over a bounded closed convex set P is achieved at an extreme point of P.*

Theorem 5.2 (Vertex Optimal (Lukatskii & Shapot, 2001))**.** *Suppose P_i is a bounded closed convex set for all $i = 1, \ldots, m$. Let $P = \times_{i=1}^{m} P_i$ and $ext(P) = \times_{i=1}^{m} ext(P_i)$. Let $f : P \mapsto \mathbb{R}$ be component-wise concave, i.e. $f(q_i, q_{-i}) : P_i \mapsto \mathbb{R}$ is a concave function of q_i for every fixed value of q_{-i}. Then*

$$\min_{q \in P} f(q) = \min_{q \in ext(P)} f(q).$$

We utilize this result to develop a heuristic based on the Frank Wolfe algorithm which is fast, scalable to large graphs and always contains the true query value in empirical experiments.

5.1. Frank Wolfe Heuristic

The Frank-Wolfe (FW) optimization algorithm proposed in (Frank & Wolfe, 1956) (also known as conditional gradient method (Demianov & Rubinov, 1970)), is a popular first-order method to solve constrained optimization problems of form

$$\min_{\boldsymbol{x} \in \mathcal{M}} f(\boldsymbol{x})$$

where $f : R^d \to R$ is a continuously differentiable convex function over the domain $\mathcal{M}$ that is convex and compact. However, the algorithm is known to provide good solutions even on non-convex objectives with a Lipschitz continuous gradient, with Lacoste-Julien (2016) showing that it is guaranteed to converge to a stationary point at a rate of $O(\frac{1}{\sqrt{t}})$. Its speed and scalability in machine learning applications has led to a surge in popularity, see (Jaggi, 2013; Lacoste-Julien & Jaggi, 2015) and references therein. See also (Lan, 2013) for a related survey.

In each iteration, the Frank–Wolfe algorithm considers a linear approximation of the objective function, and moves towards a minimizer of this linear function (taken over the same domain), and hence only requires access to a linear minimization oracle over the feasible set. Let $\boldsymbol{q}^{(t)}$ denote the iterate at iteration t, and A_k denote the polyhedron represented by constraints corresponding to C_k i.e.

$$A_k = \left\{ \begin{array}{ll} \boldsymbol{q}_{r_{C_k}} : & \sum_{r_{C_k} \in R_{v_{C_k} \cdot v_{P_{C_k}}}} \boldsymbol{q}_{r_{C_k}} \\ & = p_{v_{C_k} \cdot v_{P_{C_k}}}, \forall v_{P_{C_k}}, v_{C_k} \\ & \boldsymbol{q}_{r_{C_k}} \geq \boldsymbol{0} \end{array} \right\}$$

We discuss the algorithm for the minimization problem which computes lower bounds, but the implementation of the algorithm for the maximization problem is analogous. The procedure is initialized by generating a random feasible solution $\boldsymbol{q}^{(0)}$. In each iteration t, we compute:

$$\boldsymbol{s}^{(t)} \coloneqq \text{argmin}_{\boldsymbol{s} \in \bigcap_{k=1}^{m} A_k} \boldsymbol{s}^\top \nabla g(\boldsymbol{q}^{(t)})$$

Equivalently, since the constraints are separable, for every $k = 1 \ldots, m$, we have

$$\begin{aligned} &\boldsymbol{s}^{(t)}_{r_{C_k}} = \\ &\text{argmin}_{\boldsymbol{s}_{r_{C_k}} \in A_k} g(\boldsymbol{q}^{(t)}_{r_{C_1}}, \ldots, \boldsymbol{s}, \ldots, \boldsymbol{q}^{(t)}_{r_{C_m}}) \end{aligned} \tag{6}$$

Note that for each $k = 1 \ldots, m$, (6) is a linear program. Hence, to compute the vertex to move towards in each iteration, we decompose the optimization problem in each step into m linear programs which can be solved in parallel. Furthermore, each linear program can be further pruned using techniques for computing bounds in a single C-component. We refer the interested reader to Shridharan & Iyengar (2022) for details. Note that while the ApproxLP heuristic updates only a single block of variables in each iteration, multiple blocks of variables are simultaneously updated in our heuristic (Antonucci et al., 2015).

A popular technique used to determine step sizes in the Frank Wolfe algorithm is line search, where the step size at iteration t, γ_t, is given by:

$$\gamma_t \coloneqq \text{argmin}_{\gamma \in [0,1]} g(\boldsymbol{q}^{(t)} + \gamma(\boldsymbol{s}^{(t)} - \boldsymbol{q}^{(t)}))$$

Since g is coordinate-wise concave on the line segment, the optimal solution must lie on one of the two vertices. Hence,

$$\gamma_t = \text{argmin}_{\gamma \in \{0,1\}} g(\boldsymbol{q}^{(t)} + \gamma(\boldsymbol{s}^{(t)} - \boldsymbol{q}^{(t)}))$$

That is, in each iteration, we either stay at the current vertex $\boldsymbol{q}^{(t)}$ or take the full step to the new vertex $\boldsymbol{s}^{(t)}$ if it improves

the objective value. Iterations are repeated until convergence, and the entire procedure is repeated T times, each with a random initialization of the first iterate $\boldsymbol{q}^{(0)}$.

Algorithm 1 presents our complete heuristic. In each iteration, $\boldsymbol{q}^{(t,n-1)}$ and $g_{n-1}^{(t)}$ denote the solution and objective value from the previous iteration, while $\boldsymbol{q}^{(t,n)}$ and $g_{n}^{(t)}$ denote the solution and objective value from the current iteration. When we can no longer improve the objective value i.e. $g_{n-1}^{(t)} = g_{n}^{(t)}$, the algorithm terminates.

Algorithm 1 Frank Wolfe Heuristic for Lower Bound

for $t = 1, \dots, T$ **do**
  Initialize $g_{n-1}^{(t)} = 1.1$.
  for $k = 1, \dots, m$ **do**
    Randomly initialize $\boldsymbol{q}_{r_{C_k}}^{(t,n-1)} \in A_k$
  Initialize $g_{n}^{(t)} \coloneqq g(\boldsymbol{q}^{(t,n-1)})$.
  while $g_{n}^{(t)} < g_{n-1}^{(t)}$ **do**
    $g_{n-1}^{(t)} \coloneqq g_{n}^{(t)}$
    for $k = 1, \dots, m$ **do**
      Compute $\boldsymbol{s}_{r_{C_k}}^{(n)} \coloneqq \min_{\boldsymbol{s} \in P_k} g(\boldsymbol{q}_{r_{C_1}}^{(t,n-1)}, \dots, \boldsymbol{s}, \dots, \boldsymbol{q}_{r_{C_m}}^{(t,n-1)})$
    if $g(\boldsymbol{s}^{(n)}) < g(\boldsymbol{q}^{(t,n-1)})$ **then**
      $g_{n}^{(t)} \coloneqq g(\boldsymbol{s}^{(n)})$
      $\boldsymbol{q}^{(t,n)} \coloneqq \boldsymbol{s}^{(n)}$
    else
      $g_{n}^{(t)} \coloneqq g(\boldsymbol{q}^{(t,n-1)})$
      $\boldsymbol{q}^{(t,n)} \coloneqq \boldsymbol{q}^{(t,n-1)}$
return $\min_{t=1,\dots,T} g_{n}^{(t)}$

Although the interval produced by the heuristic will lie inside the true tight interval, we show that in empirical experiments, the interval produced by our heuristic always contains the true query value of the causal inference problem. Furthermore, Theorems 5.3 and 5.4 show that our objective function has a Lipschitz continuous gradient, and as a result, our heuristic is guaranteed to converge to a stationary point. This offers a compelling alternative to other branch-and-bound methods which are guaranteed to contain the true query value, but are computationally intensive and non-scalable.

Theorem 5.3 (Lipschitz Continuous Gradient)**.** *There exists $L < \infty$ such that $\nabla g(\boldsymbol{q})$ is L-Lipschitz continuous on the feasible set $\bigcap_{k=1}^{m} A_k$.*

Proof. See Appendix B. □

Theorem 5.4 (Convergence of Frank-Wolfe (Lacoste-Julien, 2016))**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ denote a continuously differentiable function that is potentially non-convex, but is L-Lipschitz continuous over the compact convex domain $\mathcal{M}$. Then the Frank Wolfe Algorithm with exact line search converges to an approximate stationary point with gap at most ϵ in $O(\frac{1}{\epsilon^2})$ iterations, where the gap $g(\boldsymbol{x})$ of a stationary point $\boldsymbol{x}$ is defined as:*

$$g(\boldsymbol{x}) \coloneqq \max_{s \in \mathcal{M}} \langle s - \boldsymbol{x}, -\nabla f(\boldsymbol{x}) \rangle$$

5.2. Numerical Experiments

We evaluate our heuristic on queries for the quasi-markovian graphs in Figure 1g, 1h and 1j in Zaffalon et al. (2020a). We selected these graphs since they are the largest among the family of graphs in Figure 1 in Zaffalon et al. (2020a), hence showing that the algorithm is scalable to large causal graphs and outputs valid bounds despite their complexity. We evaluate our heuristic on queries with 3, 6 and 10 C-components intervened upon. Note that we appropriately modify the graphs in (Zaffalon et al., 2020a) by adding extra edges to ensure that the graphs remain connected after performing the intervention in the query. Overall, we evaluate our heuristic on 9 causal inference problems. The details of the problems are in the appendix.

We evaluate the performance of our heuristic on each causal inference problem as follows. For each problem, we randomly generate 50 values of $\boldsymbol{q}$ as our ground truth. Then for each value, we compute the corresponding true value of the query, and input data distribution $\mathbb{P}(V_{C_k}|V_{P_{C_k}}), k = 1, \dots, m$. We then run Algorithm 1 on the input data distribution with $T = 10$, and check if the output bounds contain the true value of the query . We find that for every causal inference problem, and every ground truth $\boldsymbol{q}$ generated for each problem, the bounds computed by Algorithm 1 was always valid i.e. always contained the true query value. In Sections 5.2.2 and 5.2.1, we benchmark Algorithm 1 against the most recent alternatives proposed in literature to our knowledge: both variations of Approx LP and the branch-and-bound algorithm in Duarte et al. (2021).

5.2.1. Benchmarking against Approx LP

We compare the speed and accuracy of Algorithm 1 with both variations of ApproxLP, and found that for the same number of random restarts ($T = 10$), Algorithm 1 converges to bounds close to that of both versions of ApproxLP on average for all examples, but is significantly faster, especially for large graphs. Table 1 compares the runtime of the Frank Wolfe Heuristic with both versions of the ApproxLP algorithm where m denotes the number of C-components intervened on in the example, and t_{FW}, t_{GAP}, t_{RAP} denote the average runtime of each heuristic in seconds.

Let $\alpha_L^{FW}, \alpha_L^{GAP}, \alpha_L^{RAP}$ denote the lower bounds computed by the Frank Wolfe, Greedy ApproxLP (GAP), and Ran-

Figure	m	t_{FW}(s)	t_{RAP}(s)	t_{GAP}(s)
2 (a)	3	0.2	0.2	0.3
2 (b)	3	0.18	0.23	0.26
2 (c)	3	0.19	0.25	0.29
3 (a)	6	3.5	4.1	5.2
3 (b)	6	0.7	1.1	1.2
3 (c)	6	1.3	2.1	2.1
4 (a)	10	45.3	60.7	69.2
4 (b)	10	17.5	37.8	31.2
4 (c)	10	39.5	75.3	72.7
Extension of 4 (b)	16	1921	5032	3929

Table 1: Average runtime of the Frank Wolfe vs Approx LP Heuristics in each example. m denotes the number of C-components intervened upon in the example. t_{FW}, t_{GAP}, t_{RAP} denote the average runtime of each heuristic in seconds. We evaluated our heuristic on causal inference problems with 3, 6 and 10 C-components, and their details are in the appendix. We also computed average runtimes on 20 ground truth values in an extension of the graph and query in Figure 4 (b) to 16 C-components with $T = 1$.

dom ApproxLP (RAP) heuristics. Let $\alpha_U^{FW}, \alpha_U^{GAP}, \alpha_U^{RAP}$ denote the upper bounds computed by the Frank Wolfe, Greedy ApproxLP, and Random ApproxLP heuristics. Then $\epsilon_L^{GAP} = \frac{\alpha_L^{GAP} - \alpha_L^{FW}}{\alpha_L^{GAP}}$ and $\epsilon_L^{RAP} = \frac{\alpha_L^{RAP} - \alpha_L^{FW}}{\alpha_L^{RAP}}$ i.e. the relative improvement in the lower bound of Frank Wolfe compared to GAP and RAP. Similarly, $\epsilon_U^{GAP} = \frac{\alpha_U^{FW} - \alpha_U^{GAP}}{\alpha_U^{RAP}}$ and $\epsilon_L^{RAP} = \frac{\alpha_U^{FW} - \alpha_U^{RAP}}{\alpha_U^{RAP}}$. Note that positive values of relative improvements indicate that Frank Wolfe outperformed the Approx LP heuristic, and vice versa. Table 2 reports the mean and 95-percent confidence interval of $\epsilon_L^{RAP}, \epsilon_U^{RAP}, \epsilon_L^{GAP}$ and ϵ_U^{GAP} over 50 ground truth samples.

On average, the bounds computed by Frank Wolfe are close to those computed by the Approx LP heuristics for all examples. Recall further that in every sample, Frank Wolfe computed valid bounds which contained the true query value.

5.2.2. Benchmarking against Duarte et al. (2021)

Algorithm 2 in Duarte et al. (2021) is guaranteed to compute valid bounds on termination at any point. However, it is also computationally inefficient even for small graphs. In this section, we benchmark Algorithm 1 against our best-case interpretation of Algorithm 2 in Duarte et al. (2021) by terminating the algorithm at the time Algorithm 1 terminates.

Note that it is not clear how certain critical sub-procedures in

Figure	mean ϵ_L^{RAP}	mean ϵ_U^{RAP}	mean ϵ_L^{GAP}	mean ϵ_U^{GAP}
2(a)	0.0000 ±0.0000	0.0000 ±0.0000	0.0000 ±0.0000	0.0000 ±0.0000
2(b)	0.0000 ±0.0000	-0.0002 ±0.0004	0.0000 ±0.0000	0.0075 ±0.0073
2(c)	0.0000 ±0.0000	-0.0002 ±0.0004	0.0000 ±0.0000	0.0001 ±0.0002
3(a)	0.0000 ±0.0000	0.0000 ±0.0000	0.0000 ±0.0000	0.0000 ±0.0000
3(b)	-0.0071 ±0.0128	-0.0005 ±0.0007	-0.0071 ±0.0128	0.0121 ±0.0069
3(c)	0.0000 ±0.0000	0.0000 ±0.0000	0.0000 ±0.0000	0.0005 ±0.0007
4(a)	0.0000 ±0.0000	0.0000 ±0.0001	0.0000 ±0.0001	0.0000 ±0.0001
4(b)	-0.0091 ±0.0070	-0.0024 ±0.0025	-0.0091 ±0.0070	0.0173 ±0.0179
4(c)	0.0000 ±0.0000	-0.0003 ±0.0005	0.0000 ±0.0000	0.0000 ±0.0007
Extension of 4 (b)	-0.0031 ±0.0047	-0.0018 ±0.0037	-0.0031 ±0.0047	0.0115 ±0.0228

Table 2: Mean relative improvements of Frank Wolfe over the Approx LP heuristic in each example. $\epsilon_L^{GAP} = \frac{\alpha_L^{GAP} - \alpha_L^{FW}}{\alpha_L^{GAP}}$ and $\epsilon_L^{RAP} = \frac{\alpha_L^{RAP} - \alpha_L^{FW}}{\alpha_L^{RAP}}$ denote the relative improvement in the lower bound of Frank Wolfe compared to GAP and RAP. Similarly, $\epsilon_U^{GAP} = \frac{\alpha_U^{FW} - \alpha_U^{GAP}}{\alpha_U^{RAP}}$ and $\epsilon_L^{RAP} = \frac{\alpha_U^{FW} - \alpha_U^{RAP}}{\alpha_U^{RAP}}$. Note that positive values of relative improvements indicate that Frank Wolfe outperformed the Approx LP heuristic, and vice versa.

Algorithm 2 in Duarte et al. (2021) were implemented (e.g. how the branch b chosen in the sub-procedure is partitioned, how the LP relaxation $D_{b'}$ is obtained for each subpartition of the branch, etc). Since the authors mention that they *"employ a variation of the spatial branch-and-bound method, combined with a piecewise linear envelope, implemented using a variety of optimization frameworks that include SCIP"*, we benchmark our heuristic against the SCIP solver for nonlinear programming. Furthermore, in order to ensure that any difference in performance is not due to the choice of solver, we solved the linear programming sub-procedures of our heuristic using SCIP.

Tables 3 and 4 list the difference in bounds computed by Algorithm 1 and the SCIP solver at the time when Algorithm 1 terminates. t_{FW} denotes the runtime in seconds of Algorithm 1 averaged over 50 ground truth samples for figures 2(a), 2(b) and 2(c). Recall our query is a probability, and so, is guaranteed to lie in $[0, 1]$. Algorithm 1 almost always produces informative bounds (i.e. in almost 100% of samples, $\alpha_L^{FW} > 0$ and $\alpha_U^{FW} < 1$). However, the SCIP primal lower bound α_L^P was finite in approximately 90% of samples for 2(b) and 2(c), and in none of the samples for

Figure	t_{FW}(s)	finite α_U^P (%)	average ϵ_L	finite α_U^P (%)	average ϵ_U
2 (a)	3.58	0	–	0	–
2 (b)	1.62	84	0.05 $\pm$0.06	36	0.01 $\pm$0.02
2 (c)	2.33	88	0.04 $\pm$0.04	34	0.01 $\pm$0.03

Table 3: Difference in bounds computed by Algorithm 1 and the primal bounds computed by the SCIP solver at the time Algorithm 1 terminates. t_{FW}(s) is the runtime in seconds of the FW heuristic averaged over samples for figures 2(a), 2(b) and 2(c). Note that the SCIP primal lower bound was finite in approximately 90% of samples for 2(b) and 2(c), and in none of the samples for 2(a). Let ϵ_L denote the relative improvement of Algorithm 1 compared to the SCIP primal lower bound. The average relative improvement of Algorithm 1 on those samples with finite SCIP primal lower bound was positive. The SCIP primal upper bound α_U^P was finite in only approximately 40 percent of samples for 2(b) and 2(c), and none for 2(a). The average relative improvement of Algorithm 1 over the SCIP primal upper bound on samples with finite upper bound was also positive.

Figure	t_{FW} (s)	$\alpha_L^D > 0$ (%)	$\alpha_U^D < 1$(%)	$\alpha_L^{FW} > 0$ (%)	$\alpha_U^{FW} < 1$ (%)
2 (a)	3.58	0	0	100	100
2 (b)	1.62	0	0	98	98
2 (c)	2.33	0	0	100	100

Table 4: Difference in bounds computed by Algorithm 1 and the dual bounds computed by the SCIP solver at the time Algorithm 1 terminates. t_{FW}(s) is the runtime in seconds of the FW heuristic averaged over samples for figures 2(a), 2(b) and 2(c). Recall our query is a probability, and so, is guaranteed to lie in $[0, 1]$. Algorithm 1 almost always produces informative bounds (i.e. in almost 100% of samples, $\alpha_L^{FW} > 0$ and $\alpha_U^{FW} < 1$). However, for every sample of examples 2(a)-(c) the SCIP dual bounds $\alpha_L^D \leq 0$ and $\alpha_U^D \geq 1$ in the alotted time.

2(a). Let $\epsilon_L^P = \frac{\alpha_L^P - \alpha_L^{FW}}{\alpha_L^P}$ denote the relative improvement of the lower bound computed by Algorithm 1 compared to the SCIP primal lower bound. The average relative improvement of Algorithm 1 on those samples with finite SCIP primal lower bound was positive. The SCIP primal upper bound α_U^P was finite in only approximately 40 percent of samples for 2(b) and 2(c), and none for 2(a). The average relative improvement of Algorithm 1 over the SCIP primal upper bound on samples with finite upper bound was also positive. For every sample of examples 2(a)-(c) the SCIP dual bounds $\alpha_L^D \leq 0$ and $\alpha_U^D \geq 1$ in the alotted time. This is completely uninformative for our setting.

Thus, in summary, it is clear that our Algorithm 1 outperforms the SCIP solver (our best-case interpretation of the algorithm presented in Duarte et al. (2021)) in that it computes bounds of much higher quality in the same time. This confirms the fact that although Duarte et al. (2021) flexibly accommodates a variety of simple models, their algorithm does not scale to large graphs with multiple C-components. Indeed, developing algorithms that can scalably compute bounds which are guaranteed to be valid in large graphs is challenging; our work offers a step in this direction.

6. Conclusion

We show that the multilinear program for computing bounds for causal queries in quasi-markovian graphs can be reduced to a simpler multilinear program with a polynomial objective of lower degree. In particular, the degree of the objective in the new multilinear program is equal to only the number of C-components which are intervened upon in the causal inference problem, instead of the total number of C-components. As a result, any causal inference problem on a quasi-Markovian graph with at most two variables in the intervention set can be solved via a simple linear or quadratic program using readily available optimization solvers.

We also present a method to obtain causal bounds when three or more C-components are intervened upon, and the simplified multilinear program has a polynomial objective of degree at least three. We utilize a previous result on vertex optimality in multilinear programs to develop a heuristic based on the Frank Wolfe algorithm which is fast, scalable to large graphs, always contains the true query value in empirical experiments and is guaranteed to converge to a stationary point. Furthermore, while the ApproxLP heuristic updates only a single block of variables in each iteration, multiple blocks of variables are simultaneously updated in our heuristic. We benchmark our heuristic against the most recent alternatives proposed in literature to our knowledge: both variations of Approx LP and the branch-and-bound algorithm in Duarte et al. (2021). On average, our heuristic converges to bounds close to that of both versions of ApproxLP for all examples, but is significantly faster, especially for large graphs. Furthermore, our heuristic outperforms our best-case interpretation of the algorithm presented in Duarte et al. (2021) in that it computes bounds of much higher quality in the same time.

Acknowledgements

We thank Junzhe Zhang for useful discussions about causal bounds in quasi-markovian causal graphs.

References

Antonucci, A., De Campos, C. P., Huber, D., and Zaffalon, M. Approximate credal network updating by linear programming with applications to decision making. *International Journal of Approximate Reasoning*, 58:25–38, 2015.

Balke, A. and Pearl, J. Counterfactual probabilities: Computational methods, bounds and applications. In *Uncertainty Proceedings 1994*, pp. 46–54. Elsevier, 1994.

Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Cano, A., Gómez, M., Moral, S., and Abellán, J. Hill-climbing and branch-and-bound algorithms for exact and approximate inference in credal networks. *International Journal of Approximate Reasoning*, 44(3):261–280, 2007. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2006.07.020. URL https://www.sciencedirect.com/science/article/pii/S0888613X06000971. Reasoning with Imprecise Probabilities.

Chen, X., Deng, X., and Teng, S.-H. Settling the complexity of computing two-player nash equilibria. *Journal of the ACM (JACM)*, 56(3):1–57, 2009.

D'Amour, A. On multi-cause approaches to causal inference with unobserved counfounding: Two cautionary failure cases and a promising alternative. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 3478–3486. PMLR, 16–18 Apr 2019. URL https://proceedings.mlr.press/v89/d-amour19a.html.

de Campos, C. P. and Cozman, F. G. Inference in credal networks using multilinear programming. In *Proceedings of the Second Starting AI Researcher Symposium*, pp. 50–61, 2004.

De Campos, L. M., Huete, J. F., and Moral, S. International journal of uncertainty, fuzziness and knowledge-based °y***** vol. 2, no. 2 (1994) 167–196. *World*, 2(2):167–196, 1994.

Demianov, V. F. and Rubinov, A. M. *Approximate methods in optimization problems*. Number 32. Elsevier Publishing Company, 1970.

Duarte, G., Finkelstein, N., Knox, D., Mummolo, J., and Shpitser, I. An automated approach to causal inference in discrete settings. *arXiv preprint arXiv:2109.13471*, 2021.

Evans, R. J. Graphical methods for inequality constraints in marginalized DAGs. In *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–6, 2012. doi: 10.1109/MLSP.2012.6349796.

Falk, J. E. and Hoffman, K. R. A successive underestimation method for concave minimization problems. *Mathematics of operations research*, 1(3):251–259, 1976.

Finkelstein, N. and Shpitser, I. Deriving bounds and inequality constraints using logical relations among counterfactuals. In *Conference on Uncertainty in Artificial Intelligence*, pp. 1348–1357. PMLR, 2020.

Finkelstein, N., Adams, R., Saria, S., and Shpitser, I. Partial identifiability in discrete data with measurement error. In de Campos, C. and Maathuis, M. H. (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1798–1808. PMLR, 27–30 Jul 2021. URL https://proceedings.mlr.press/v161/finkelstein21b.html.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2): 95–110, 1956.

Geiger, D. and Meek, C. Quantifier elimination for statistical problems, 2013. URL https://arxiv.org/abs/1301.6698.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com.

Housni, O. E., Foussoul, A., and Goyal, V. Lp-based approximations for disjoint bilinear and two-stage adjustable robust optimization. In Aardal, K. and Sanità, L. (eds.), *Integer Programming and Combinatorial Optimization*, pp. 223–236, Cham, 2022. Springer International Publishing. ISBN 978-3-031-06901-7.

Imbens, G. W. and Rubin, D. B. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

Jaggi, M. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pp. 427–435. PMLR, 2013.

Janzing, D. and Schölkopf, B. Detecting confounding in multivariate linear models via spectral analysis. *Journal of Causal Inference*, 6(1):20170013, 2018. doi: doi:10.1515/jci-2017-0013. URL https://doi.org/10.1515/jci-2017-0013.

Kallus, N. and Zhou, A. Confounding-robust policy evaluation in infinite-horizon reinforcement learning, 2020. URL https://arxiv.org/abs/2002.04518.

Kilbertus, N., Kusner, M. J., and Silva, R. A class of algorithms for general instrumental variable models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 20108–20119. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/e8b1cbd05f6e6a358a81dee52493dd06-Paper.pdf.

Lacoste-Julien, S. Convergence rate of frank-wolfe for non-convex objectives, 2016.

Lacoste-Julien, S. and Jaggi, M. On the global linear convergence of frank-wolfe optimization variants. *Advances in neural information processing systems*, 28, 2015.

Lan, G. The complexity of large-scale convex programming under a linear optimization oracle. *arXiv preprint arXiv:1309.5550*, 2013.

Lukatskii, A. and Shapot, D. Problems in multilinear programming. *Computational Mathematics and Mathematical Physics*, 41(5):638–648, 2001.

Lyu, H. Convergence and complexity of block coordinate descent with diminishing radius for nonconvex optimization, 2020. URL https://arxiv.org/abs/2012.03503.

Pearl, J. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 052189560X.

Poderini, D., Chaves, R., Agresti, I., Carvacho, G., and Sciarrino, F. Exclusivity graph approach to instrumental inequalities. In Adams, R. P. and Gogate, V. (eds.), *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 1274–1283. PMLR, 22–25 Jul 2020. URL https://proceedings.mlr.press/v115/poderini20a.html.

Powell, M. J. On search directions for minimization algorithms. *Mathematical programming*, 4(1):193–201, 1973.

Ranganath, R. and Perotte, A. Multiple causal inference with latent confounding, 2019.

Richardson, A., Hudgens, M. G., Gilbert, P., and Fine, J. Nonparametric bounds and sensitivity analysis of treatment effects. *Stat Sci*, 29(4):596–618, 2014. doi: 10.1214/14-STS499.

Sachs, M. C., Gabriel, E. E., and Sjolander, A. Symbolic computation of tight causal bounds. *Biometrika*, 103(1): 1–19, 2020.

Sachs, M. C., Jonzon, G., Sjölander, A., and Gabriel, E. E. A general method for deriving tight symbolic bounds on causal effects. *Journal of Computational and Graphical Statistics*, pp. 1–10, 2022.

Shridharan, M. and Iyengar, G. Scalable computation of causal bounds. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20125–20140. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/shridharan22a.html.

Tian, J. *Studies in causal reasoning and learning*. PhD thesis, 2002. URL http://ezproxy.cul.columbia.edu/login?url=https://www.proquest.com/dissertations-theses/studies-causal-reasoning-learning/docview/304802697/se-2. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2022-10-29.

Tian, J. and Pearl, J. *A general identification condition for causal effects*. eScholarship, University of California, 2002.

Tran, D. and Blei, D. M. Implicit causal models for genome-wide association studies, 2017.

Zaffalon, M., Antonucci, A., and Cabañas, R. Structural causal models are (solvable by) credal networks. In Jaeger, M. and Nielsen, T. D. (eds.), *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pp. 581–592. PMLR, 23–25 Sep 2020a. URL https://proceedings.mlr.press/v138/zaffalon20a.html.

Zaffalon, M., Antonucci, A., and Cabañas, R. Causal expectation-maximisation, 2020b. URL https://arxiv.org/abs/2011.02912.

Zhang, J. and Bareinboim, E. Bounding causal effects on continuous outcome. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):12207–12215, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17449.

Zhang, J., Tian, J., and Bareinboim, E. Partial counterfactual identification from observational and experimental data. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 26548–26558. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/zhang22ab.html.

A. Causal Inference Problems in Experiments

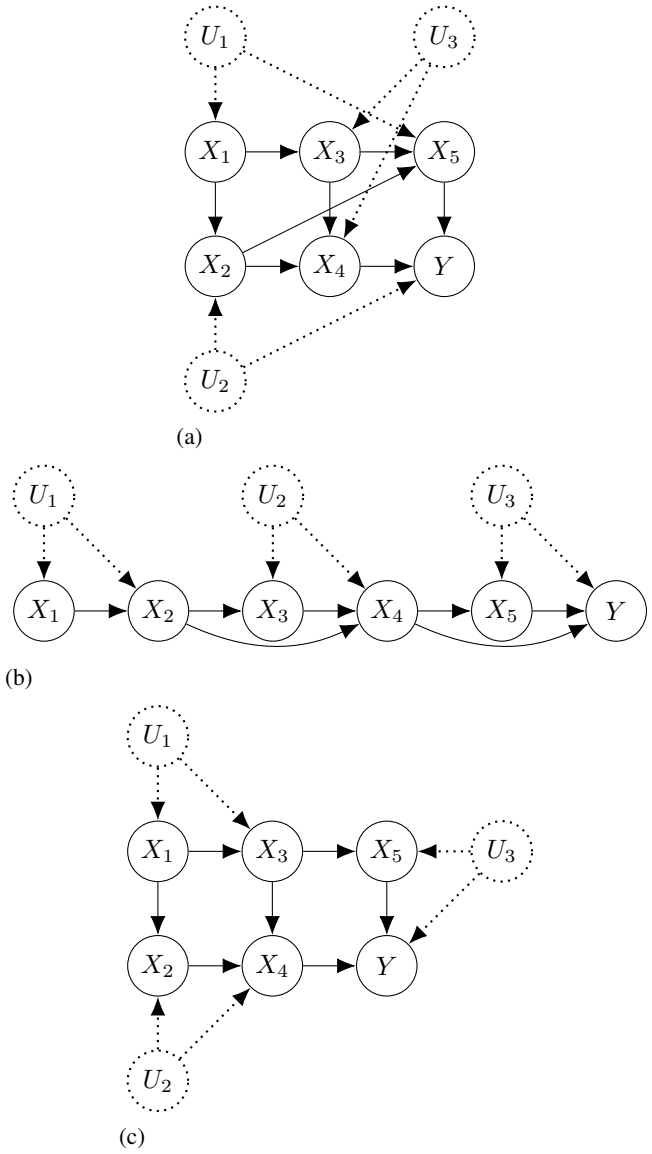
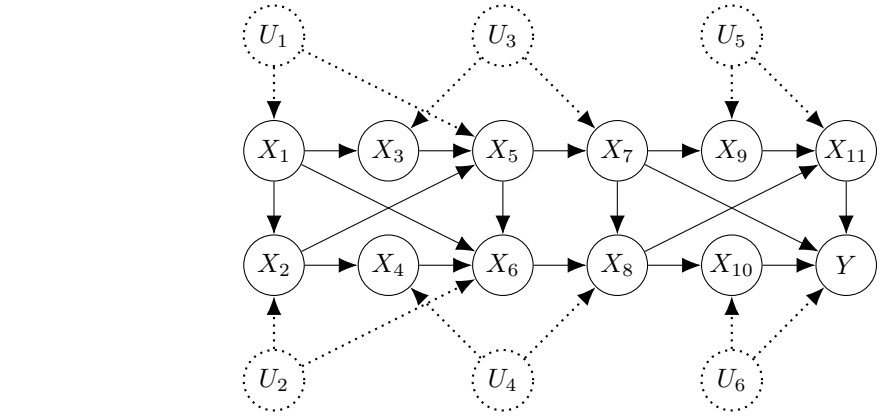
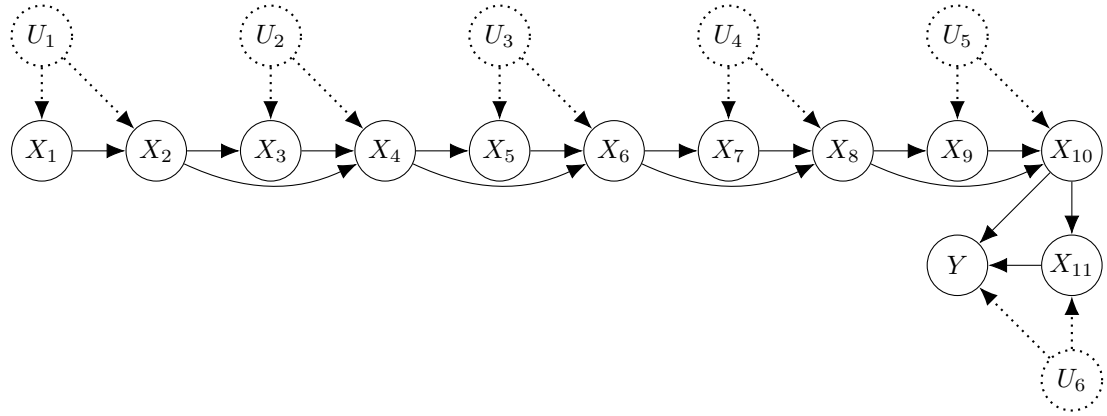


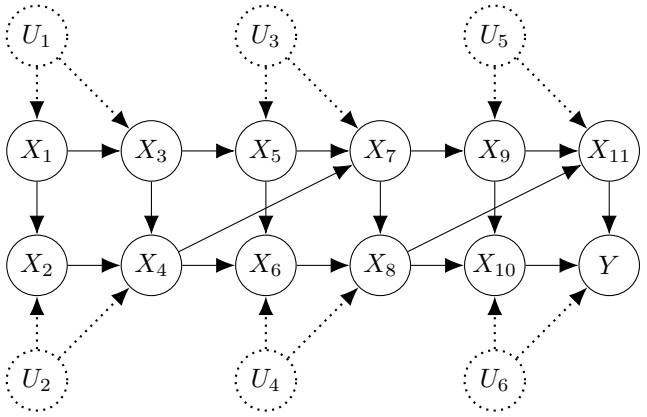
Figure 2: Quasi-Markovian Graphs with 3 C-components, of which all 3 C-components were intervened upon in our evaluation of Algorithm 1. The corresponding query for each graph was (a) $\mathbb{P}(Y = 1|do(X_1 = 1, X_2 = 1, X_3 = 1))$ (b) $\mathbb{P}(Y = 1|do(X_1 = 1, X_3 = 1, X_5 = 1))$ (c) $\mathbb{P}(Y = 1|do(X_1 = 1, X_2 = 1, X_5 = 1))$.



(a)

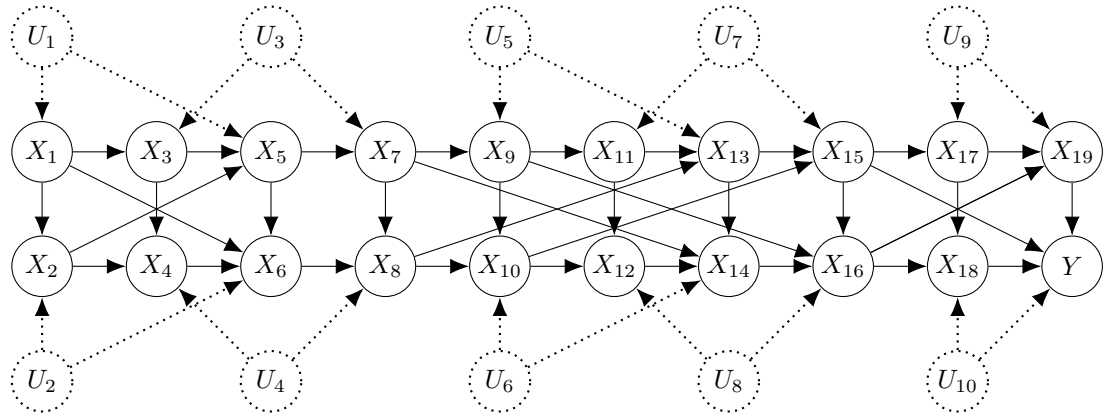


(b)

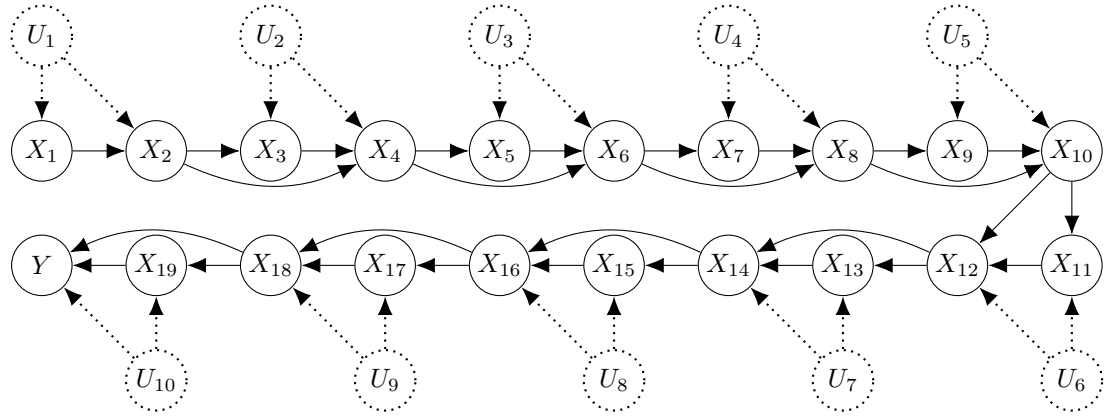


(c)

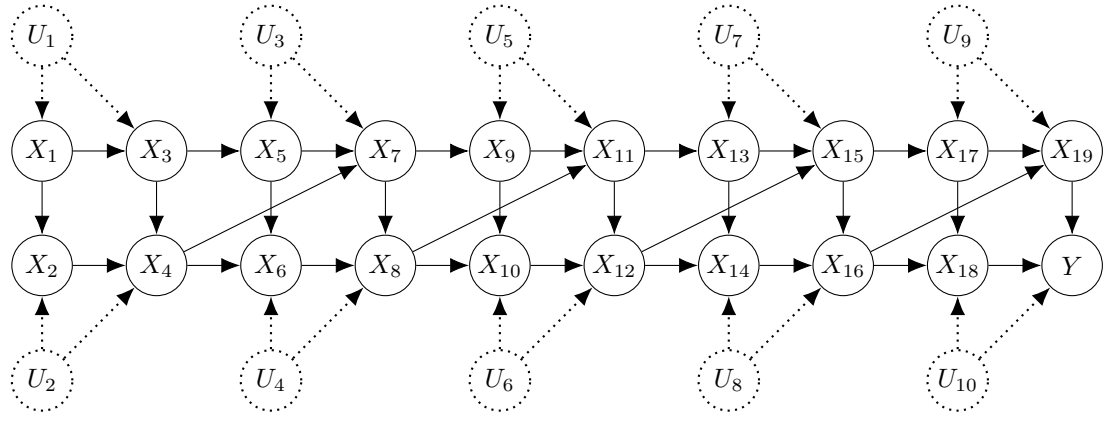
Figure 3: Quasi-Markovian Graphs with 6 C-components, of which all 6 C-components were intervened upon in our evaluation of Algorithm 1. The corresponding query for each graph was (a) $\mathbb{P}(Y = 1|do(X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 1, X_9 = 1, X_{10} = 1))$(b) $\mathbb{P}(Y = 1|do(X_1 = 1, X_3 = 1, X_5 = 1, X_7 = 1, X_9 = 1, X_{11} = 1))$ (c) $\mathbb{P}(Y = 1|do(X_1 = 1, X_2 = 1, X_5 = 1, X_6 = 1, X_9 = 1, X_{10} = 1))$.



(a)



(b)



(c)

Figure 4: Quasi-Markovian Graphs with 10 C-components, of which all 10 C-components were intervened upon in our evaluation of Algorithm 1. The corresponding query for each graph was (a) $\mathbb{P}(Y = 1|do(X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 1, X_9 = 1, X_{10} = 1, X_{11} = 1, X_{12} = 1, X_{17} = 1, X_{18} = 1))$(b) $\mathbb{P}(Y = 1|do(X_1 = 1, X_3 = 1, X_5 = 1, X_7 = 1, X_9 = 1, X_{11} = 1, X_{13} = 1, X_{15} = 1, X_{17} = 1, X_{19} = 1))$ (c) $\mathbb{P}(Y = 1|do(X_1 = 1, X_5 = 1, X_9 = 1, X_{13} = 1, X_{17} = 1, X_2 = 1, X_6 = 1, X_{10} = 1, X_{14} = 1, X_{18} = 1))$.

B. Proof of Results

Theorem 4.1 (Simplifying the Multilinear Program)**.** *Consider a causal inference problem with query $\mathbb{P}(V_O|do(V_I))$ on a quasi-markovian graph with m C-components. Let ℓ be the number of C-components which contains at least one variable in the intervention set I. Then, the problem of computing bounds is an optimization problem with linear constraints involving the given ℓ C-components and an objective that is a polynomial of degree ℓ.*

Proof. Let $Z = N \setminus (I \cup O)$ i.e. the indices of variables in the graph which are neither intervened upon nor observed in the query. Therefore $N = Z \cup O \cup I$ is a partition of N. We can rewrite the query as

$$\mathbb{P}\big(V_O \mid do(V_I)\big) = \sum_{v_Z} \mathbb{P}\big(V_O, V_Z = v_Z \mid do(V_I)\big)$$

For fixed v_Z, we claim

$$\mathbb{P}\big(V_O, v_Z \mid do(V_I)\big) = \prod_{k=1}^{m} \mathbb{P}\big(V_{O\cap C_k}, v_{Z\cap C_k} \mid do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), v_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})}\big) \tag{7}$$

First consider the base case of $m = 1$. Since $\overline{pa}(V_{C_k}) = \emptyset$ and $C_k = N$, (7) holds.

Suppose the result holds for a m C-component problem. Next, we show that the result holds for a $m+1$ C-component problem:

$$\begin{aligned}
&\mathbb{P}\big(V_O, v_Z \mid do(V_I)\big)\\
&\overset{(a)}{=} \mathbb{P}(V_{O\cap C_{m+1}}, v_{Z\cap C_{m+1}}|do(V_{I\cap\overline{pa}(C_{m+1})}, V_{I\cap C_{m+1}}), v_{Z\cap\overline{pa}(C_{m+1})}, V_{O\cap\overline{pa}(C_{m+1})})\\
&\mathbb{P}(V_{O\setminus C_{m+1}}, v_{Z\setminus C_{m+1}}, v_{Z\cap\overline{pa}(C_{m+1})}, V_{O\cap\overline{pa}(C_{m+1})}|do(V_{I\setminus C_{m+1}}))\\
&\overset{(b)}{=} \mathbb{P}(V_{O\cap C_{m+1}}, v_{Z\cap C_{m+1}}|do(V_{I\cap\overline{pa}(C_{m+1})}, V_{I\cap C_{m+1}}), v_{Z\cap\overline{pa}(C_{m+1})}, V_{O\cap\overline{pa}(C_{m+1})})\\
&\mathbb{P}(V_{O\setminus C_{m+1}}, v_{Z\setminus C_{m+1}}|do(V_{I\setminus C_{m+1}}))\\
&\overset{(c)}{=} \mathbb{P}(V_{O\cap C_{m+1}}, v_{Z\cap C_{m+1}}|do(V_{I\cap\overline{pa}(C_{m+1})}, V_{I\cap C_{m+1}}), v_{Z\cap\overline{pa}(C_{m+1})}, V_{O\cap\overline{pa}(C_{m+1})})\\
&\Pi_{k=1}^{m}\mathbb{P}\big(V_{O\cap C_k}, v_{Z\cap C_k} \mid do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), v_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})}\big)\\
&= \Pi_{k=1}^{m+1}\mathbb{P}\big(V_{O\cap C_k}, v_{Z\cap C_k} \mid do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), v_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})}\big)
\end{aligned}$$

where (a) follows from the conditional independence of $V_{C_{m+1}}$ from other variables in G given $\overline{pa}(V_{C_{m+1}})$, and (c) follows from the induction hypothesis. The result is a special case of the C-component factorization result first shown in (Tian, 2002).

Suppose $V_{I\cap C_k} = \emptyset$ i.e. no variable in the k-th C-component is in the intervention set. Then, it follows that

$$\begin{aligned}
(V_{O\cap C_k}, V_{Z\cap C_k}) &= V_{C_k},\\
(V_{I\cap\overline{pa}(V_{C_k})}, V_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})}) &= V_{\overline{pa}(V_{C_k})}.
\end{aligned}$$

Therefore, every term of form

$$\mathbb{P}(V_{O\cap C_k}, V_{Z\cap C_k}|do(V_{I\cap\overline{pa}(V_{C_k})}), V_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})}) = \mathbb{P}(V_{C_k}|V_{\overline{pa}(V_{C_k})})$$

is a constant completely defined by the data.

Fix $k \in \{1, \ldots, m\}$ such that $V_{I\cap C_k} \neq \emptyset$. Consider every term of form

$$\mathbb{P}(V_{O\cap C_k}, V_{Z\cap C_k}|do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), V_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})}). \tag{8}$$

We can rewrite this term as:

$$\begin{aligned}
&\mathbb{P}(V_{O\cap C_k}, V_{Z\cap C_k}|do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), V_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})})\\
&= \sum_{r_{C_k}} \mathbb{P}(V_{O\cap C_k}, V_{Z\cap C_k}|do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), V_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})}, r_{C_k})\mathbb{P}(r_{C_k})\\
&= \sum_{r_{C_k}} \mathbb{P}(V_{O\cap C_k}, V_{Z\cap C_k}|do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), V_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})}, r_{C_k})q_{r_{C_k}}
\end{aligned}$$

Fix r_{C_k}. Every variable in $\overline{pa}(V_{C_k})$ is either in I, Z or O. Hence, in the term

$$\mathbb{P}(V_{O\cap C_k}, V_{Z\cap C_k}|do(V_{I\cap\overline{pa}(V_{C_k})}, V_{I\cap C_k}), V_{Z\cap\overline{pa}(V_{C_k})}, V_{O\cap\overline{pa}(V_{C_k})}, r_{C_k})$$

every variable in $\overline{pa}(V_{C_k})$ is either intervened upon or conditioned on. Since r_{C_k} is also conditioned on, each term of this form is a deterministic constant. Hence, (8) is linear in $q_{r_{C_k}}$. Now, the result follows from (7) and the two observations above.

□

Lemma B.1 ((Falk & Hoffman, 1976)). *The minimum value of a real-valued continuous concave function $f(x)$ over a bounded closed convex set P is achieved at an extreme point of P.*

Proof. Since P is a bounded closed set and $f(x)$ is a continuous function, the minimum $\min\{f(x) : x \in P\}$ is achieved at some point x^*. Suppose x^* is not an extreme point. Then, $x^* = \sum_{i=1}^{M} \lambda_i v_i$ where $\lambda \geq 0$, $\sum_i \lambda = 1$, and $v_i \in \text{ext}(P)$ for all $i = 1, \ldots, M$. Then we have that

$$f(x^*) = f\big(\sum_{i=1}^{M} \lambda_i v_i\big) \geq \sum_{i=1}^{M} \lambda_i f(v_i) \geq \min\{f(v_i) : i = 1, \ldots, M\}$$

where the first inequality follows from the concavity of f, and the second from the properties of convex combinations. Thus, it follows that $v^* = \text{argmin}\{f(v_i) : i = 1, \ldots, M\}$ is an extreme point with $f(v^*) \leq f(x^*)$, and hence $f(v^*) = f(x^*)$.

□

Theorem 5.2 (Vertex Optimal (Lukatskii & Shapot, 2001)). *Suppose P_i is a bounded closed convex set for all $i = 1, \ldots, m$. Let $P = \times_{i=1}^{m} P_i$ and ext$(P) = \times_{i=1}^{m}$ ext(P_i). Let $f : P \mapsto \mathbb{R}$ be component-wise concave, i.e. $f(q_i, q_{-i}) : P_i \mapsto \mathbb{R}$ is a concave function of q_i for every fixed value of q_{-i}. Then*

$$\min_{q\in P} f(q) = \min_{q\in ext(P)} f(q).$$

Proof. First consider the base case of $m = 1$. Lemma 5.1 establishes the result for this case.

Suppose the result holds for a m component problem. Next, we show that the result holds for the $m+1$ component problem:

$$\begin{aligned}
&\min_{(q_{-(m+1)}, q_{m+1})\in P} f(q_{-(m+1)}, q_{m+1})\\
&= \min_{q_{-(m+1)}\in\times_{i=1}^{m} P_i} \Big\{ \min_{q_{m+1}\in P_{m+1}} f(q_{-(m+1)}, q_{m+1}) \Big\},\\
&\overset{(a)}{=} \min_{q_{-(m+1)}\in\times_{i=1}^{m} P_i} \Big\{ \min_{q_{m+1}\in \text{ext}(P_{m+1})} f(q_{-(m+1)}, q_{m+1}) \Big\},\\
&\overset{(b)}{=} \min_{q_{m+1}\in \text{ext}(P_{m+1})} \Big\{ \min_{q_{-(m+1)}\in\times_{i=1}^{m} P_i} f(q_{-(m+1)}, q_{m+1}) \Big\},\\
&\overset{(c)}{=} \min_{q_{m+1}\in \text{ext}(P_{m+1})} \Big\{ \min_{q_{-(m+1)}\in \text{ext}(\times_{i=1}^{m} P_i)} f(q_{-(m+1)}, q_{m+1}) \Big\},
\end{aligned}$$

where (a) follows from Lemma 5.1 applied to the concave function $f(q_{-(m+1)}, q_{m+1})$ as a function of q_{m+1}, (b) follows from rearranging the order of the minimization, and (c) from the induction hypothesis.

□

Theorem 5.3 (Lipschitz Continuous Gradient)**.** *There exists $L < \infty$ such that $\nabla g(\boldsymbol{q})$ is L-Lipschitz continuous on the feasible set $\bigcap_{k=1}^{m} A_k$.*

Proof. We show that the hessian $\nabla^2 g(\boldsymbol{q})$ has a bounded spectral norm over the feasible set i.e. there exists $L < \infty$ such that $\|\nabla^2 g(\boldsymbol{q})\|_2 \leq L$ for $\boldsymbol{q} \in \bigcap_{k=1}^{m} A_k$. This implies that ∇g is L-Lipschitz continuous on $\bigcap_{k=1}^{m} A_k$ (Boyd et al., 2004).

Since $0 \leq \boldsymbol{q} \leq 1$ for all $\boldsymbol{q} \in \cap_{k=1}^{m} A_k$, it follows that for each element $\nabla^2 g(\boldsymbol{q})_{ij}$, the Hessian staisfies $0 \leq \nabla^2 g(\boldsymbol{q})_{ij} \leq |R|$.

Let $M = \sum_{k=1}^{m} |R_{C_k}|$ i.e. the dimension of $\boldsymbol{q}$. For fixed $\boldsymbol{q} \in \bigcap_{k=1}^{m} A_k$, let $\lambda \in \mathbb{R}$ denote any eigenvalue of $\nabla^2 g(\boldsymbol{q})$, and $x \neq \mathbf{0} \in \mathbb{R}^M$ denote the corresponding eigenvector. We have $\nabla^2 g(\boldsymbol{q}) x = \lambda x$, therefore, for all elements $i = 1, \ldots, M$, we have that

$$\begin{aligned} |\lambda x_i| &= \Big| \sum_{j=1}^{M} \nabla^2 g(\boldsymbol{q})_{ij} x_j \Big| \\ &\leq \left(\sum_{j=1}^{M} \nabla^2 g(\boldsymbol{q})_{ij} \right) \max_{j \in \{1,\ldots,M\}} |x_j| \\ &\leq M|R| \max_{j \in \{1,\ldots,M\}} |x_j| \end{aligned}$$

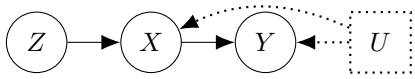
Since $|\lambda x_i| = |\lambda||x_i|$, the above inequality implies that

$$|\lambda| \max_{i \in \{1,\ldots,M\}} |x_i| \leq M|R| \max_{j \in \{1,\ldots,M\}} |x_j|$$

Hence, the spectral norm of $\nabla^2 g(\boldsymbol{q})$ is bounded by $L = M|R|$. □

C. Example for Section 3

Consider the following causal graph with $X, Y, Z \in \{0, 1\}$:



(U_1, U_2) are unobserved (potentially high dimensional and continuous) confounders. Suppose the input data consists of the distributions $\mathbb{P}(Z = z)$ and $\mathbb{P}(X = x, Y = y | Z = z)$, and the goal is to compute bounds for the causal query $\mathbb{P}(Y = 1 | do(X = 1))$.

We circumvent the difficulty of modeling the unobserved confounders by modeling the impact of these variables on the observed variables via response function variables.

- Confounder U_1 selects one value for the variable $Z \in \{0, 1\}$: Let r_Z denote the value for Z chosen by U_1.
- For each fixed value for the unknown confounder, the variable X is some function of Z; thus, confounder U_2 selects one function from the set $\mathcal{F} = \{f : f \text{ is a function } Z \to X\}$ with $|\mathcal{F}| = 2^2 = 4$: Let $r_X \in \{0, \ldots, 3\}$ index the elements of $\mathcal{F}$, and let f_{r_X} denote the r_X-th function from $\mathcal{F}$.
- Similarly, U_2 selects one function from the set $\mathcal{G} = \{g : g \text{ is a function X } \to Y\}$ with $|\mathcal{G}| = 4$: Let $r_Y \in \{0, \ldots, 3\}$ index the elements of $\mathcal{G}$ and let g_{r_Y} denote the r_Y-th function from $\mathcal{G}$.

  Thus, it is clear that the response function variables r_X, r_Y and r_Z can be used to model the impact of U_1 and U_2. In particular, the unknown distributions over U_1 and U_2 can be equivalently modeled via the distributions $\boldsymbol{q}^1_{r_Z} = \mathbb{P}(r_Z)$ and $\boldsymbol{q}^2_{r_X r_Y} = \mathbb{P}(r_X, r_Y)$.

For fixed $z \in \{0, 1\}$, (r_X, r_Y) maps $Z = z$ to $X = f_{r_X}(z)$ and $Y = g_{r_Y}(f_{r_X}(z))$ i.e. (r_X, r_Y) uniquely determines the realization of X and Y. Let

$$R_{xy.z} = \{(r_X, r_Y) : f_{r_X}(z) = x, g_{r_Y}(x) = y\}$$

denote the set of (r_X, r_Y) values which map $Z = z$ to $X = x, Y = y$. Then we have

$$\mathbb{P}(X = x, Y = y|Z = z) = \sum_{(r_X,r_Y)\in R_{xy.z}} \boldsymbol{q}^2_{r_X r_Y}, \qquad \mathbb{P}(Z = z) = \boldsymbol{q}^1_{r_Z}(z)$$

Let

$$R_{\mathcal{Q}} = \{(r_X, r_Y, r_Z) : g_{r_Y}(1) = 1\}$$

denote the set of response function variable values consistent with the query $\mathcal{Q} = \mathbb{P}(Y = 1|do(X = 1))$. Hence, from the independence of r_Z and (r_X, r_Y), it follows that

$$\mathbb{P}(Y = 1|do(X = 1)) = \sum_{(r_X,r_Y,r_Z)\in R_{\mathcal{Q}}} \boldsymbol{q}^1_{r_Z} \boldsymbol{q}^2_{r_X r_Y},$$

Thus, we have that the following multilinear programs give bounds on the query:

$$\begin{array}{rl} \max_q / \min_q & \sum_{(r_X,r_Y,r_Z)\in R_{\mathcal{Q}}} \boldsymbol{q}^1_{r_Z} \boldsymbol{q}^2_{r_X r_Y} \\ \text{s.t.} & \sum_{(r_X,r_Y)\in R_{xy.z}} \boldsymbol{q}^2_{r_X r_Y} = \mathbb{P}(X = x, Y = y|Z = z), \forall x, y, z \in \{0, 1\} \\ & \boldsymbol{q}^1_z = \mathbb{P}(Z = z), \forall z \in \{0, 1\} \\ & \boldsymbol{q}^1, \boldsymbol{q}^2 \geq 0, \end{array} \tag{9}$$

Note that the constraints $\mathbb{P}(Z = 0) = \boldsymbol{q}^1_0$ and $\mathbb{P}(Z = 1) = \boldsymbol{q}^1_1$ together imply $\boldsymbol{q}^1_0 + \boldsymbol{q}^1_1 = \mathbb{P}(Z = 0) + \mathbb{P}(Z = 1) = 1$, and so the constraint $\boldsymbol{q}^1_0 + \boldsymbol{q}^1_1 = 1$ is not explicitly added to the multilinear program.

Similarly, the constraint $\sum_{(r_X,r_Y)\in\{0,\dots,3\}^2} \boldsymbol{q}^2_{r_X r_Y} = 1$ is also implied by other constraints in the multilinear program. Fix $Z = 0$. Then, summing up the constraints

$$\sum_{(r_X,r_Y)\in R_{xy.0}} \boldsymbol{q}^2_{r_X r_Y} = \mathbb{P}(X = x, Y = y|Z = 0),$$

for all $x, y \in \{0, 1\}$, we have

$$\sum_{x,y} \sum_{(r_X,r_Y)\in R_{xy.0}} \boldsymbol{q}^2_{r_X r_Y} = \sum_{x,y} \mathbb{P}(X = x, Y = y|Z = 0) = 1$$

Now recall that fixing $Z = 0$, (r_X, r_Y) uniquely determine the value of (X, Y). Hence, $\bigcup_{x,y} R_{xy.0}$ is a partition of $\{0, \dots, 3\}^2$, the set of all possible values of (r_X, r_Y). Hence

$$\sum_{x,y} \sum_{(r_X,r_Y)\in R_{xy.0}} \boldsymbol{q}^2_{r_X r_Y} = \sum_{(r_X,r_Y)\in\{0,\dots,3\}^2} \boldsymbol{q}^2_{r_X r_Y}$$

Therefore, it follows that $\sum_{(r_X,r_Y)\in\{0,\dots,3\}^2} \boldsymbol{q}^2_{r_X r_Y} = 1$.