Multilinear and Linear Programs for Partially Identifiable Queries in Quasi-Markovian Structural Causal Models

João P. Arroyo[1] **João G. Rodrigues**[1] **Daniel Lawand**[1] **Denis D. Mauá**[1] **Junkyu Lee**[2] **Radu Marinescu**[2]
Alex Gray[3] **Eduardo R. Laurentino**[4] **Fabio G. Cozman**[1]

[1] Universidade de São Paulo, São Paulo, Brazil
[2]IBM Research – J. L.: Yorktown Heights, USA; R. M.: Ireland
[3]Centaur Institute, USA
[4]Instituto de Ciência e Tecnologia Itaú, São Paulo, Brazil

Abstract

We investigate partially identifiable queries in a class of causal models. We focus on acyclic Structural Causal Models that are quasi-Markovian (that is, each endogenous variable is connected with at most one exogenous confounder). We look into scenarios where endogenous variables are observed (and a distribution over them is known), while exogenous variables are not fully specified. This leads to a representation that is in essence a Bayesian network where the distribution of root variables is not uniquely determined. In such circumstances, it may not be possible to precisely compute a probability value of interest. We thus study the computation of tight probability bounds, a problem that has been solved by multilinear programming in general, and by linear programming when a single confounded component is intervened upon. We present a new algorithm to simplify the construction of such programs by exploiting input probabilities over endogenous variables. For scenarios with a single intervention, we apply column generation to compute a probability bound through a sequence of auxiliary linear integer programs, thus showing that a representation with polynomial cardinality for exogenous variables is possible. Experiments show column generation techniques to be superior to existing methods.

1 INTRODUCTION

Structural Causal Models (SCMs) offer a representation where some variables are associated with deterministic mechanisms while other variables are associated with marginal probabilities. We refer to the former variables as endogenous ones, and to the latter variables as exogenous ones.

It is often the case that observational data determines the probability distribution of endogenous variables, but not the distribution of exogenous variables. In fact, the structure of exogenous variables may not be specified, and all that one may know are bounds on the cardinalities of those variables. In such a setting, the causal diagram that captures the connections between variables can be viewed as a Bayesian network where the (marginal) distribution of root variables is not specified [Zaffalon et al., 2020]. This is the sort of representation we explore in this paper.

Given such a causal model, one may be interested in some probability value (or some function of probability values) under interventions. When such a query leads to a single precise number, we say the query is identifiable. Pearl's do-calculus can be used to determine when a query is identifiable [Pearl, 2009].

When identifiability fails, one can still bound probability values given a distribution over the observed variables. Hopefully, one may then produce probability intervals that are sufficiently informative to make decisions — when the observable variables are discrete, this can be achieved without additional assumptions on the unknown mechanisms at play [Balke and Pearl, 1997].

In this paper we focus on the computation of tight probabilistic bounds when identifiability fails; that is, we focus on the partially identifiable setting. We abuse language by using "SCM" to also refer to models where the marginal distribution over exogenous variables is not known precisely, but rather imprecisely induced by a distribution over endogenous variables.

We restrict interest to *quasi-Markovian* SCMs; that is, to SCMs where each endogenous variable has at most one exogenous parent. This family of SCMs is quite expressive; for instance, it contains Balke and Pearl's imperfect compliance example and many of their extensions in the literature. Moreover, quasi-Markovian SCMs can be used to approximate non-quasi-Markovian ones [Zhang et al., 2022].

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

There have been several relevant proposals to cast partially identifiable queries with quasi-Markovian SCMs as nonlinear programs that, in some cases, reduce to linear programs. For example, Balke and Pearl [1994] showed that linear programming can bound the causal effect of an intervention in the so-called instrumental variable model. Tian and Pearl [2000] then showed how to write a linear program to compute the probability of necessity and sufficiency in two-variable binary models. Sachs et al. [2023] extended Balke and Pearl [1994]'s models to a larger class, for which they showed that causal effect inferences can be cast as linear programs; the size of these linear programs has been studied, and in many cases reduced, by Shridharan and Iyengar [2023a]. Zhang et al. [2022] described techniques that bound the cardinality of non-observed variables and that can be used to approximate bounds on causal effects through linear programming. Duarte et al. [2024] instead focused on the general multilinear programs that produce anytime probability bounds (in the sense that the algorithm can be stopped at any given time and still yield approximate bounds). Zaffalon et al. [2024] translated the computation of causal inferences to inference in credal networks, and looked at approximate solutions based on sequences of linear programs and on parametric learning (a credal network is, in essence, a Bayesian network where conditional probability distributions are only known to belong to specified sets of distributions [Cozman, 2000]).

Recently, Shridharan and Iyengar [2023b] derived a significant result for quasi-Markovian SCMs. In short, for those SCMs, tight bounds on causal effects can be computed by multilinear programs whose degree is restricted to the number of intervened confounded components — and hence to linear programming when a single confounded component is intervened upon! It is within the context of Shridharan and Iyengar's work that we make our contributions.

We first present a new proof for the main result by Shridharan and Iyengar [2023b], a proof that hopefully shows this clever result to be in essence a simple one from a mathematical point of view (Section 3). We then present a new algorithm that shows how to exploit information in an input distribution over endogenous variables to simplify the construction of such multilinear/linear programs (Section 4). Our new algorithm exploits Pearl's do-calculus to build up a simplified objective function.

One practical challenge is that bounds on the cardinality of exogenous variables may be very large, thus leading to large multilinear/linear programs. To avoid this explosion, we introduce column generation techniques; that is, we show how to build a sequence of basis changes that lead to the desired bound (Section 5). Each change of basis uses a common master linear program and an auxiliary program that, despite the presence of polynomial terms in its specification, can be reduced to a linear integer program. We also show how to build a *single* linear integer program, that directly produces desired bounds. We present empirical evidence that our proposed techniques are superior to existing approaches.

2 BACKGROUND

We write random variables using capital letters (e.g., X) and sets of random variables using boldface (e.g., $\mathbf{X}$). The support of X is denoted as $\mathrm{val}(X)$, and $\mathrm{val}(\mathbf{X})$ is the direct product of the support of each variable in the set. A probability value is denoted by $\Pr(\mathbf{X} = \mathbf{x})$, or $\Pr(\mathbf{x})$ when appropriate. A distribution is denoted by $\Pr(\mathbf{X})$. We consider graphs whose nodes are random variables, and refer indiscriminately to nodes and random variables. We write $\mathrm{Pa}(X)$ to denote the parents of X in a graph $\mathcal{G}$. We write $\mathcal{G}_{\underline{\mathbf{X}}}$ to denote the graph obtained from $\mathcal{G}$ by removing all edges leaving nodes in the node set $\mathbf{X}$, and $\mathcal{G}_{\overline{\mathbf{X}}}$ to denote the graph obtained by removing edges entering nodes in $\mathbf{X}$.

A *Structural Causal Model* (SCM) is a tuple $(\mathcal{G}, \mathbf{V}, \mathbf{U}, \mathcal{F}, \Pr(\mathbf{U}))$ where $\mathcal{G}$ is an directed graph whose node set is $\mathbf{V} \cup \mathbf{U}$, where $\mathbf{V}$ are the inner nodes, called *endogenous*, and $\mathbf{U}$ are the root nodes, called *exogenous*; $\mathcal{F}$ is a set of functions $f_V : \mathrm{val}(\mathrm{Pa}_{\mathcal{G}}(V)) \to \mathrm{val}(V)$ called *mechanisms*, one for each node V in $\mathcal{G}$; and $\Pr(\mathbf{U})$ is a probability distribution over the exogenous random variables $\mathbf{U}$ [Galles and Pearl, 1998, Halpern, 2000].

We restrict ourselves here to acyclic graphs. For such models, we have for any endogenous variable $V \in \mathbf{V}$ that

$$\Pr(V = v | \mathrm{Pa}(V) = \pi) = [\![f_V(\pi) = v]\!],$$

where $[\![\theta]\!]$ denotes the Iverson bracket (i.e., it is 1 if statement θ holds, and 0 otherwise). We assume exogenous nodes are independent, hence $\Pr(\mathbf{u}) = \prod_{U \in \mathbf{U}} \Pr(U = u)$. Consequently,

$$\Pr(\mathbf{v}, \mathbf{u}) = \prod_{V} [\![f_V(\pi) = v]\!] \prod_{U} \Pr(U = u), \qquad (1)$$

for values of v, π and u that are consistent with $\mathbf{v}$ and $\mathbf{u}$. The latter expression is multilinear on the probabilities of exogenous variables. Any marginal probability is therefore a multilinear expression of $\{\Pr(U = u) : u \in \mathrm{val}(U), U \in \mathbf{U}\}$.

A *confounded component* (for short, c-component) of a directed graph $\mathcal{G}$ is a set of endogenous nodes in a maximal connected component of the undirected version of graph $\mathcal{G}_{\underline{\mathbf{V}}}$ (i.e., the graph obtained by removing endogenous-to-endogenous edges) [Tian, 2002].

An SCM is *quasi-Markovian* if every endogenous variable has at most one exogenous variable as parent. If in addition every exogenous variable has exactly one child, then the model is said to be Markovian. Figure 1 (a) shows the graph of a quasi-Markovian SCM (that is not Markovian), taken from a rather simple example by [Sachs et al., 2023]. The model has two c-components: $\{X, W\}$ and $\{Z, Y\}$.

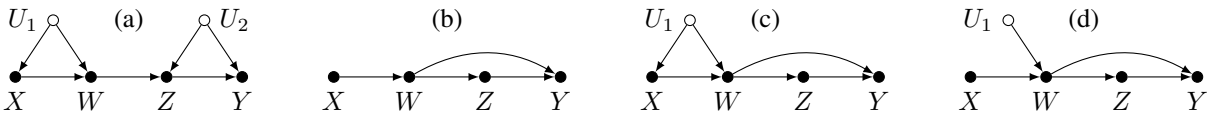

Figure 1: (a) An example proposed by Sachs et al. [2023]. (b) The factorization of the (marginal) distribution for endogenous variables. (c) A semi-marginal graph that marginalizes U_2. (d) The intervened semi-marginal graph for $\mathrm{do}(X = x)$.

Consider a c-component $\mathbf{C}$ of a quasi-Markovian SCM, and let U be the (single) exogenous parent of nodes in $\mathbf{C}$, and $\mathbf{W_C}$ denote the union of the variables in $\mathbf{C}$ and all of their endogenous parents. A key result by Tian [2002] is that:

$$\Pr(\mathbf{V} = \mathbf{v}) = \prod_{\mathbf{C}} Q_{\mathbf{C}}(\mathbf{w_C}), \tag{2}$$

where $\mathbf{w_C}$ is the configuration of $\mathbf{W_C}$ that is consistent with $\mathbf{v}$, and

$$Q_{\mathbf{C}}(\mathbf{W_C}) = \sum_{u} \Pr(u) \prod_{V \in \mathbf{C}} [\![f_V(\mathrm{Pa}(V)) = V]\!]. \tag{3}$$

For $V \in \mathbf{C}$, let $\mathbf{W}_V$ denote the variables that are topologically smaller than V in $\mathbf{W_C}$. Tian [2002] also showed that:

$$Q_{\mathbf{C}}(\mathbf{W_C}) = \prod_{V \in \mathbf{C}} \Pr(V|\mathbf{W}_V). \tag{4}$$

As c-components form a partition of $\mathbf{V}$, we have that:

$$\Pr(\mathbf{V}) = \prod_{V \in \mathbf{V}} \Pr(V|\mathbf{W}_V)\,. \tag{5}$$

For example, for the graph in Figure 1 (a), Equation (5) implies $\Pr(W, X, Y, Z) = \Pr(X)\Pr(W|X)\Pr(Z|W)\Pr(Y|W, Z)$ (as captured by Figure 1 (b)).

As Zaffalon et al. [2024] noted, Equations (3) and (4) lead to necessary and sufficient linear constraints over $\Pr(U)$:

$$\prod_{V \in \mathbf{C}} \Pr(V|\mathbf{W}_V) = \sum_{u} \Pr(u) \prod_{V \in \mathbf{C}} [\![f_V(\mathrm{Pa}(V)) = V]\!]. \tag{6}$$

Note that there is one constraint per configuration of $\mathbf{W_C}$.

When endogenous random variables are categorical, one can always extend a causal graph into a partially specified SCM, that is, an SCM without a fixed exogenous distribution $\Pr(\mathbf{U})$. That process is known as *canonicalization* [Zhang et al., 2022], and essentially, consists of enumerating all possible mechanisms via values of the exogenous variables. For quasi-Markovian graphs, canonicalization reduces each exogenous variable U to a categorical random variable whose state space has cardinality $\prod_{V \in \mathbf{C}} |\mathrm{val}(V)|^{|\mathrm{val}(\mathrm{Pa}(V))|}$, where $\mathbf{C}$ is the corresponding c-component. Each value $u \in \mathrm{val}(U)$ specifies a mechanism $f_V : \mathrm{Pa}_{\mathcal{G}}(V) \to V$ for each V in the c-component. Thus, we assume without loss of generality that every exogenous variable is categorical.

A simple *intervention* $\mathrm{do}(\mathbf{X} = \mathbf{x})$ modifies an SCM by substituting f_X with $[\![X = x]\!]$, where x is the corresponding value in $\mathbf{x}$, for every $X \in \mathbf{X}$. Graphically, interventions are represented by means of surgery of $\mathcal{G}$, producing $\mathcal{G}_{\overline{\mathbf{X}}}$.

An intervention induces a new (post-intervention) distribution over any variable set $\mathbf{Y}$, denoted as $\Pr(\mathbf{Y}|\mathrm{do}(\mathbf{x}))$. The goal of causal inference is to estimate expressions involving such probabilities using the constraints shared by the non-intervened and intervened SCMs. We are interested in the calculation of population-level causal effects that can be written as linear combinations of post-intervention probabilities, such as the average treatment effect (ATE) and the conditional average treatment effect (CATE). For the sake of exposition, we concentrate on simple inferences such as $\Pr(\mathbf{Y}|\mathrm{do}(\mathbf{x}))$; however, the same algorithms and results apply for linear combinations of such probabilities.

3 COMPUTING PROBABILITY BOUNDS

Suppose we have a partially specified quasi-Markovian SCM $(\mathcal{G}, \mathbf{V}, \mathbf{U}, \mathcal{F})$ and an *input* distribution $\widehat{\Pr}(\mathbf{V})$. In the remainder of this paper we assume endogenous variables are binary; this substantially simplifies the presentation by reducing the number of necessary indexes (extension to categorical variables should be clear).

We are interested in computing $\Pr(\mathbf{Y} = \mathbf{y}|\mathrm{do}(\mathbf{X} = \mathbf{x}))$, abbreviated $\Pr(\mathbf{y}|\mathrm{do}(\mathbf{x}))$. We do not assume identifiability; that is, the input distribution may not be sufficient to constrain $\Pr(\mathbf{y}|\mathrm{do}(\mathbf{x}))$ to a point value; for this reason, we wish to compute the *lower probability* defined as

$$\underline{\Pr}(\mathbf{y}|\mathrm{do}(\mathbf{x})) := \inf \Pr(\mathbf{y}|\mathrm{do}(\mathbf{x})),$$

where the infimum is taken over the set of all extensions of the given partially specified SCM whose induced joint distributions satisfies the set of constraints given by Equation (6), with

$$\Pr(V|\mathbf{W}_V) = \widehat{\Pr}(V|\mathbf{W}_V).$$

Zaffalon et al. [2024] have shown this optimization problem to lead to tight bounds. Note that the fact that the feasible region is a closed polytope allows us to replace infimum by minimum in the expression above, so

$$\underline{\Pr}(\mathbf{y}|\mathrm{do}(\mathbf{x})) = \min \Pr(\mathbf{y}|\mathrm{do}(\mathbf{x})).$$

One might as well be interested in the *upper probability*, by taking maximum instead of minimum; the necessary changes should be obvious.

As shown by Shridharan and Iyengar [2023b], the multilinear expression for $\Pr(\mathbf{y}|\mathrm{do}(\mathbf{x}))$ need only involve probabilities $\Pr(U = u)$ for exogenous variables connected with c-components intervened by $\mathrm{do}(\mathbf{X} = \mathbf{x})$.[1] Thus, the degree of the multilinear expression is equal to the number of intervened c-components. This result was proved by them by explicitly developing the relevant factors in the multilinear expression; we now present a shorter argument directly based on Tian's factorization given by Expression (2).

To start, let $\mathbf{Z} = \mathbf{V} \backslash \{\mathbf{X}, \mathbf{Y}\}$. Thus $\Pr(\mathbf{y}|\mathrm{do}(\mathbf{x}))$ is equal to

$$\sum_{\mathbf{z}} \Pr(\mathbf{y}, \mathbf{z}, \mathbf{x}|\mathrm{do}(\mathbf{x})) = \sum_{\mathbf{z}} \prod_{\mathbf{C}} Q_{\mathbf{C}}^{\mathrm{do}(\mathbf{x})}(\mathbf{w}_{\mathbf{C}}),$$

where the latter equality follows from Expression (2) with each $\mathbf{w}_{\mathbf{C}}$ being consistent with $\mathbf{y}$, $\mathbf{z}$ and $\mathbf{x}$. To proceed, let $\mathcal{C}_0$ denote the collection of c-components of the non-intervened model that *do not* contain intervened variables. Note that a c-component $\mathbf{C} \in \mathcal{C}_0$ does *not* change under the intervention, as an intervention $\mathrm{do}(\mathbf{X} = \mathbf{x})$ modifies only the mechanisms related to $X \in \mathbf{X}$. As a consequence, in the expression above, each term $Q_{\mathbf{C}}^{\mathrm{do}(\mathbf{x})}$ that refers to a c-component $\mathbf{C} \in \mathcal{C}_0$ equals the analogous term $Q_{\mathbf{C}}$ in the factorization of the non-intervened SCM (as both are given by identical instances of Expression (3)). Moreover, by Equation (4) $Q_{\mathbf{C}}$ can be written as a product of empirical probabilities $\widehat{\Pr}(v|\mathbf{w}_V)$ for each $V \in \mathbf{C}$. Similarly, for each $\mathbf{C}$ not in $\mathcal{C}_0$ the term $Q_{\mathbf{C}}^{\mathrm{do}(\mathbf{x})}$ in the expression above differs from the analogous term $Q_{\mathbf{C}}$ only with respect to the mechanisms for the intervened variables, which become $[\![X = x]\!]$. Connecting all observations above, we get:

$$\Pr(\mathbf{y}|\mathrm{do}(\mathbf{x})) = \sum_{\mathbf{z}} \prod_{\mathbf{C} \in \mathcal{C}_0} \prod_{V \in \mathbf{C}} \widehat{\Pr}(v|\mathbf{w}_V) \times \prod_{\mathbf{C} \notin \mathcal{C}_0} \sum_{u} \Pr(u) \prod_{V \in \mathbf{C} \backslash \mathbf{X}} [\![f_V(\pi) = v]\!], \quad (7)$$

where all assignments inside the outer summation must agree with the assignments for $\mathbf{Y} = \mathbf{y}$, $\mathbf{Z} = \mathbf{z}$, and $\mathbf{X} = \mathbf{x}$. Now define $\mathbf{U}^*$ to be the set of exogenous variables that are parents of intervened c-components. Then we have that

$$\underline{\Pr}(\mathbf{y}|\mathrm{do}(\mathbf{x})) = \min_{\Pr(U): U \in \mathbf{U}^*} \Pr(\mathbf{y}|\mathrm{do}(\mathbf{x})),$$

where $\Pr(\mathbf{y}|\mathrm{do}(\mathbf{x}))$ is given by Expression (7), subject to the set of constraints given by Equation (6). As that defines a multilinear program, we conclude our proof of Shridharan and Iyengar [2023b]'s main result.

It is actually possible to convey this latter result in a rather visual way by using graphs. A simple example should clarify the idea. Consider an SCM with graph as in Figure 1 (a), and an intervention $\mathrm{do}(X = x)$. We may marginalize all exogenous variables (Figure 1 (b)), or just U_2 (Figure 1 (c)); the latter produces what we call a *semi-marginal graph*. Equation (7) denotes the factorization given by Equation (1) in the graph in Figure 1 (d). In the latter, the exogenous variable U_1, associated with the intervened c-component $\{W, X\}$ of the original graph, is kept while the exogenous U_2 in the non-intervened c-component $\{Z, Y\}$ is discarded — note the edge $W \to Y$ is included to account for the missing exogenous variable.

That is, we can take the directed graph, create a topological order for the variables, and for each c-component $\mathbf{C}$ that is not intervened,

- remove the corresponding exogenous variable, and
- connect each variable V in $\mathbf{C}$ with the variables in $\mathbf{W}_V$ (recall that $\mathbf{W}_V$ contains the variables that are topologically smaller than V among the variables in $\mathbf{C}$ and the variables that are parents of variables in $\mathbf{C}$);

and then remove edges corresponding to the interventions as usual. We refer to this latter type graph as the *intervened semi-marginal* graph, that is, the graph obtained by marginalizing exogenous variables connected with non-intervened c-components and performing surgery on the intervened variables. Additional intervened semi-marginal graphs can be found in the Supplementary Material.

4 EXPLOITING INPUT DISTRIBUTIONS

Suppose we have a quasi-Markovian causal graph $\mathcal{G}$, an input distribution $\widehat{\Pr}(\mathbf{V})$, target variables $\mathbf{Y}$ and a *single* intervention $\mathrm{do}(X = x)$. Then Shridharan and Iyengar [2023b]'s result show us that $\Pr(\mathbf{y}|\mathrm{do}(x))$ can be cast as a linear program whose objective function, given by Equation (7), contains $\mathrm{val}(U \cup \mathbf{Z})$ terms. We can instead simplify the expression for instance by running symbolic variable elimination [Koller and Friedman, 2009] with the factors defined by the semi-marginal graph. Doing so leads to an expression whose number of terms is given by $\mathrm{val}(U)$ times an exponential in the graph's treewidth (which measures the connectivity of the causal graph).

However, we can do better. As we now show, we may require only a smaller set of factors, and less exogenous variables, by taking a different route.

As a preliminary point, note that any node that is not in the ancestral graph of $\mathbf{Y}$ can be discarded, as it will not affect the result of the causal inference. This can be seen from the intervened semi-marginal graph, which in that case will have such a node d-separated from $\mathbf{Y}$ or barren. That holds even for intervened $X \in \mathbf{X}$; we thus assume that each X is an ancestor of some $Y \in \mathbf{Y}$ in the following.

Now consider simplifications that exploit the input distribution. To do so, we take the input distribution as an oracle that can yield any conditional distribution $\widehat{\Pr}(\mathbf{R}|\mathbf{S})$, for any

[1] This assumes that the optimization is feasible.

$\mathbf{R}, \mathbf{S} \subseteq \mathbf{V}$. In practice, efficient data structures can be used to obtain such probabilities from e.g. a dataset (note: "local" probabilities are needed anyway to handle Expression (7)).

We present an algorithm that produces a sequence of expressions for auxiliary values $\Pr(\mathbf{Y}_t|\mathrm{do}(x))$; the objective function can be built at the end by collecting such expressions, leading to an expression that potentially is exponentially more succinct that the one given by Expression (7).

Starting with the input target set, $\mathbf{Y}_0 = \mathbf{Y}$, at each step $t = 0, 1, \ldots$, the algorithm selects a variable $Y_t \in \mathbf{Y}_t$ that has no descendants in $\mathbf{Y}_t$; that is, the algorithm selects variables $Y_0 < Y_1 < \cdots$ in reverse topological order. It then defines a next target set $\mathbf{Y}_{t+1}$ and produces a symbolic expression relating $\Pr(\mathbf{Y}_t|\mathrm{do}(x))$ and $\Pr(\mathbf{Y}_{t+1}|\mathrm{do}(x))$, according to the following cases.

C1: Y_t **is in** $\mathbf{U}$ **(hence** Y **is not a descendant of** X**).**
Then build $\mathbf{Y}_{t+1} = \mathbf{Y}_t \setminus \{Y_t\}$ and output

$$\Pr(\mathbf{y}_t|\mathrm{do}(x)) = \Pr(\mathbf{y}_{t+1}|\mathrm{do}(x)) \Pr(Y_t = u)$$

for each assignment $\mathbf{y}_t$ of $\mathbf{Y}_t$, where u agrees with $\mathbf{y}_t$. If $\mathbf{Y}_{t+1} = \emptyset$ then set $\Pr(\mathbf{y}_{t+1}|\mathrm{do}(x)) = 1$.

C2: $Y_t \in \mathbf{V}$ **is not a descendant of** X**.**
Then build $\mathbf{Y}_{t+1} = \mathbf{Y}_t \setminus \{Y_t\}$ and output

$$\Pr(\mathbf{y}_t|\mathrm{do}(x)) = \Pr(\mathbf{y}_{t+1}|\mathrm{do}(x))\widehat{\Pr}(y|\mathbf{y}_{t+1})$$

for each assignment $\mathbf{y}_t$ of $\mathbf{Y}_t$.

C3: Y_t **is a descendant of** X **and they are in the same c-component.**
Then define $\mathbf{Z}_t = \mathrm{Pa}(Y_t) \setminus (\mathbf{Y}_t \cup \{X\})$, and $\mathbf{Y}_{t+1} = (\mathbf{Y}_t \cup \mathbf{Z}_t) \setminus \{Y_t\}$. For each assignment $\mathbf{y}_t$ of $\mathbf{Y}_t$, output

$$\Pr(\mathbf{y}_t|\mathrm{do}(x)) = \sum_{\mathbf{z}_t} [\![f_{Y_t}(\pi) = y_t]\!] \Pr(\mathbf{y}_{t+1}|\mathrm{do}(x)),$$

where π is the assignment of the parents of Y_t that agrees with $\mathbf{z}_t$, $\mathbf{y}_t$ and x, and y_t agrees with $\mathbf{y}_t$.

C4: Y_t **is a descendant of** X **but they are not in the same c-component.**
Define $U_t = \mathrm{Pa}(X)$. Find a set of endogenous variables $\mathbf{W}_t$ such that (i) Y_t and X are d-separated by $\mathbf{S}_t$ in $\mathcal{G}_{\underline{X}}$ (the graph where edges leaving X are removed), and (ii) Y_t and U_t are d-separated by $\mathbf{S}_t \cup \{X\}$ in $\mathcal{G}$, where $\mathbf{S}_t = (\mathbf{W}_t \cup \mathbf{Y}_t) \setminus \{Y_t, U_t\}$. Let $\mathbf{Z}_t = \mathbf{W}_t \setminus \mathbf{Y}_t$. Build $\mathbf{Y}_{t+1} = (\mathbf{Y}_t \cup \mathbf{Z}_t) \setminus \{Y_t\}$ and output

$$\Pr(\mathbf{y}_t|\mathrm{do}(x)) = \sum_{\mathbf{z}_t} \widehat{\Pr}(y_t|x, \mathbf{s}_t) \Pr(\mathbf{y}_{t+1}|\mathrm{do}(x)),$$

where the assignments on the right hand side agree with the assignments on the left hand side.

Each one of these cases produces a new expression; as noted previously, these expressions can be collected to generate a single linear expression for $\Pr(\mathbf{y}|\mathrm{do}(x))$ that contains optimization variables $\Pr(u)$ (for each u).

Figure 2: Graphs for quasi-Markovian SCMs, used in examples. None of them are handled by techniques by Sachs et al. [2023].

Before we prove the algorithm correct, consider a few examples that convey its behavior.

Example 1. Consider the graph in Figure 1 (a), an intervention $\mathrm{do}(x)$ and target $\mathbf{Y}_0 = \{Y\}$. At step $t = 0$, the algorithm selects Y. To apply C4, we find a set $\mathbf{W}_0$ that d-separates Y and X in $\mathcal{G}_{\underline{X}}$, and also d-separates Y and U_1 in $\mathcal{G}$. The only set satisfying such conditions is $\mathbf{W}_0 = \{W\}$. The algorithm produces

$$\Pr(y|\mathrm{do}(x)) = \textstyle\sum_w \widehat{\Pr}(y|x, w) \Pr(w|\mathrm{do}(x)).$$

Then the algorithm moves to $\mathbf{Y}_1 = \{W\}$, selects W, which satisfies C3, and produces

$$\Pr(w|\mathrm{do}(x)) = \textstyle\sum_{u_1} [\![f_W(x, u_1) = w]\!] \Pr(u_1|\mathrm{do}(x)).$$

Last, the algorithm takes $\mathbf{Y}_2 = \{U_1\}$, satisfying C1, and produces: $\Pr(u_1|\mathrm{do}(x)) = \Pr(u_1)$. Collecting all expressions, we get a linear objective function:

$$\textstyle\sum_w \widehat{\Pr}(y|x, w) \sum_{u_1} [\![f_W(x, u_1) = w]\!] \Pr(u_1)\,. \qquad \square$$

Example 2. Consider the graph in Figure 2 (left). Given intervention $\mathrm{do}(x)$ and target $\mathbf{Y}_0 = \{Y\}$, the algorithm selects variables in the ordering Y, Z and U_1, creates sets $\mathbf{Y}_1 = \{Z, U_1\}$, $\mathbf{Y}_2 = \{U_1\}$, and outputs, respectively:

$$\Pr(y|\mathrm{do}(x)) = \sum_{z,u_1 : f_Y(z,x,u_1) = y} \Pr(z, u_1|\mathrm{do}(x)). \quad \text{[C3]}$$

$$\Pr(z, u_1|\mathrm{do}(x)) = \widehat{\Pr}(z|x) \Pr(u_1|\mathrm{do}(x)). \quad \text{[C4]}$$

$$\Pr(u_1|\mathrm{do}(x)) = \Pr(u_1). \quad \text{[C1]}$$

Collecting all expressions, we get a linear objective function *that notably does not mention* R:

$$\textstyle\sum_{z,u_1} [\![f_Y(z, x, u_1) = y]\!] \widehat{\Pr}(z|x) \Pr(u_1)\,. \qquad \square$$

Example 3. Consider a graph as in Figure 2 (left), but with the edge $R \to Z$ replaced by a sequence $R \to R_1 \to \cdots \to R_n \to Z$, where each R_i is also connected to U_1. Then Expression (7) generates an expression with over $|\mathrm{val}(U_1)|2^n$ terms, whereas our algorithm generates the same expression as in the previous example. And if we replace each R_i by a subgraph with high treewidth, the same example shows that our algorithm can produce expressions that are exponentially smaller then by running symbolic variable elimination on the intervened semi-marginal graph. $\square$

Now consider correctness:

Theorem 1. *The previous algorithm generates a linear program that, when optimized subject to constraints given by Equation* (6)*, computes tight bounds for* $\Pr(\mathbf{y}|\mathrm{do}(x))$.

The proof requires the following lemma.

Lemma 1. *Suppose a quasi-Markovian graph $\mathcal{G}$ with endogenous nodes X, Y and $\mathbf{Z}$ is such that: (i) X and Y have no common exogenous parent; (ii) Y is a descendant of X; (iii) there are no descendants of Y in $\mathbf{Z}$. Then there is a set of endogenous variables $\mathbf{W}$ that are ancestors of $\mathbf{Z} \cup \{Y\}$ and such that X and Y are d-separated by $\mathbf{W} \cup \mathbf{Z}$ in $\mathcal{G}_{\underline{X}}$. In addition, $\mathbf{W} \cup \mathbf{Z} \cup \{X\}$ also d-separates Y and U in $\mathcal{G}$, where $U = \mathrm{Pa}(X) \cap \mathbf{U}$.*

Proof of Lemma 1. Consider the moral graph $\mathcal{M}$ of the ancestors of $\mathbf{Z}$ and Y in $\mathcal{G}_{\underline{X}}$, which also include the ancestors of X by Assumption (ii). Then, X and Y are d-separated by some superset of $\mathbf{Z}$ in $\mathcal{G}_{\underline{X}}$ iff there is no undirected path in $\mathcal{M}$ containing no endogenous nodes (other than X and Y) [Koller and Friedman, 2009]. First note that $\mathcal{M}$ has no edge X–Y, since X has no outgoing edges in $\mathcal{G}_{\underline{X}}$, Y is descendant of X by Assumption (ii) and by Assumption (iii) X and Y have no common child in $\mathcal{M}$. Thus consider a path between X and Y containing U. By Assumption (i) and because $\mathcal{G}$ is quasi-Markovian, there is no connection X–U–Y in that path. Similarly, we cannot have X–U–U', with U' being another exogenous node. Hence, any path must have at least one endogenous node on which we can condition to block it. Now the existence of an active path between Y and U in $\mathcal{G}$ would imply an active path between X and Y in $\mathcal{M}$, hence it cannot exist. □

Proof of Theorem 1. We will prove that each case follows from probability laws and Pearl's do-calculus [Pearl, 2009]. Thus consider Y_t that satisfies either C1 or C2. According to the variable selection rule, either case only occurs if there are no descendants of X in $\mathbf{Y}_t$. We thus have

$$\begin{aligned}\Pr(\mathbf{y}_t|\mathrm{do}(x)) &= \Pr(y_t, \mathbf{y}_{t+1}|\mathrm{do}(x)) \\ &= \Pr(\mathbf{y}_{t+1}|\mathrm{do}(x)) \Pr(y_t|\mathrm{do}(x), \mathbf{y}_{t+1}).\end{aligned}$$

Rule 3 of do-calculus states that:

$$\Pr(y_t|\mathrm{do}(x), \mathbf{y}_{t+1}) = \Pr(y_t|\mathbf{y}_{t+1}),$$

whenever Y_t and X are d-separated by $\mathbf{Y}_{t+1}$ in $\mathcal{G}_{\overline{\mathbf{X}}}$. Because $\mathbf{Y}_t$ does not contain descendants of $\mathbf{X}$, any path from Y_t to X in $\mathcal{G}_{\overline{X}}$ goes through some collider which has no descendants in $\mathbf{Y}_{t+1}$. Now, if $Y_t \in \mathbf{U}$, then Y_t is also d-separated from $\mathbf{Y}_{t+1}$ and $\Pr(y_t|\mathbf{y}_{t+1}) = \Pr(y_t)$.

Now consider Y_t that satisfies C3. Let $\mathbf{Z}' = \mathrm{Pa}(Y_t) \setminus \mathbf{Y}_t$; that is, unlike $\mathbf{Z}_t$, $\mathbf{Z}'$ includes X if $X \in \mathrm{Pa}(Y_t)$. Similarly, let $\mathbf{y}'$ be $\mathbf{y}_{t+1}$ possibly extended with x if $X \in \mathrm{Pa}(Y_t)$. Denote by π an assignment of the parents of $\mathrm{Pa}(Y_t)$ that is consistent with $\mathbf{y}'$. Usual probabilistic manipulation, and d-separation between a node and its nondescendants nonparents given its parents, leads to $\Pr(\mathbf{y}_t|\mathrm{do}(x)) = \sum_{\mathbf{z}'} \Pr(y_t|\pi, \mathrm{do}(x)) \Pr(\mathbf{y}'|\mathrm{do}(x))$. Now, because $\Pr(X = x|\mathbf{y}_{t+1}, \mathrm{do}(x)) = 1$, we obtain $\Pr(\mathbf{y}_t|\mathrm{do}(x)) = \sum_{\mathbf{z}_{t+1}} \Pr(y_t|\pi, \mathrm{do}(x)) \Pr(\mathbf{y}_{t+1}|\mathrm{do}(x))$. Using the fact that a parent set defines a backdoor set, the intervened conditional probability of a variable given its parents is identified with its non-intervened probability, which is just the corresponding mechanism. So,

$$\Pr(\mathbf{y}_t|\mathrm{do}(x)) = \sum_{\mathbf{z}_t} [\![f_{Y_t}(\mathbf{z}_t, \mathbf{w}_t) = y_t]\!] \Pr(\mathbf{y}_{t+1}|\mathrm{do}(x)).$$

At last, consider Y_t that satisfies C4. Note that X and Y_t have no common parent (as they are assumed in distinct c-components), and there can be no descendants of Y_t in $\mathbf{Y}_{t+1}$ (because we only add ancestors and we process variables in reverse topological order). Hence, we can apply Lemma 1 to show that there is a subset $\mathbf{W}_t$ of the endogenous ancestors of $\mathbf{Y}_t$ such that $\mathbf{S}_t$ d-separates Y_t and X in $\mathcal{G}_{\underline{X}}$, and such that $\mathbf{S}_t \cup \{X\}$ d-separates Y_t and $\mathbf{U}_t$ in $\mathcal{G}$. By probability laws, we have that $\Pr(\mathbf{y}_t|\mathrm{do}(x)) = \sum_{\mathbf{z}_t} \Pr(y_t|\mathbf{y}_{t+1}, \mathrm{do}(x)) \Pr(\mathbf{y}_{t+1}|\mathrm{do}(x))$. It follows from Rule 2 of the do-calculus that

$$\Pr(y_t|\mathbf{y}_{t+1}, \mathrm{do}(x)) = \Pr(y_t|x, \mathbf{y}_{t+1}),$$

because Y_t and X are d-separated by $\mathbf{Y}_{t+1}$ in the graph obtained by removing edges leaving X. Now since $\mathbf{S}_t \cup \{X\}$ d-separates also Y_t and U_t, we can ignore u_t from the right hand side above (if it exists), producing

$$\Pr(\mathbf{y}_t|\mathrm{do}(x)) = \sum_{\mathbf{z}_t} \widehat{\Pr}(y_t|x, \mathbf{v}_t) \Pr(\mathbf{y}_{t+1}|\mathrm{do}(x)),$$

and concluding the proof. □

All the results in this section are still correct when we have multiple interventions in the same c-component, as such a case can essentially be reduced to a single-intervention case.

5 EXPLOITING COLUMN GENERATION

We now suppose that a single c-component is intervened upon; that is, $\underline{\Pr}(\mathbf{y}|\mathrm{do}(x))$ is produced by a linear program as described in the previous section. Denote by $\mathbf{C}^*$ the intervened c-component, by $\mathbf{W}^*$ the union of the (endogenous) variables in $\mathbf{C}^*$ and their parents, and by U^* the exogenous variable connected to $\mathbf{C}^*$. For simplicity, we assume all endogenous variables are binary.

There are 2^M constraints (6), where $M := |\mathbf{W}^*|$ (one constraint per configuration of $\mathbf{W}^*$). Canonicalization may lead to a large cardinality for U^* (one value per possible mechanism), given by $\prod_{V \in \mathbf{C}^*} 2^{2^{|\mathrm{Pa}(V)|}}$ (Section 2). Depending

on the edges among variables in $\mathbf{W}^*$, the cardinality of U^* may be of order 2^{2^M}. We assume here that $|\mathbf{W}^*|$ is relatively small, say 4 or 5 variables. Even then, note that 2^{2^M} is already larger than 4 billions for $|\mathbf{W}^*| = 5$, hence we should not expect that a direct formulation of the linear program is practical.

We can write down the constraints (6) in matrix form as $\mathbf{Ap} = \hat{\mathbf{q}}$, where the vector $\mathbf{p}$ contains the optimizing variables $p_u = \Pr(U^* = u)$, indexed by values of U^*. And the vector $\hat{\mathbf{q}}$ comes from the input empirical distribution. Matrix $\mathbf{A}$ only contains zeros and ones; there are as many rows as constraints, and as many columns as values of U^*.

However, only a square matrix is actually needed at any given iteration of the revised simplex algorithm [Bertsimas and Tsitsiklis, 1997]; that is, we can keep a $2^M \times 2^M$ matrix in memory, where each column is defined by a mechanism (hence we only need to find 2^M mechanisms). We can thus resort to column generation to sequentially find the relevant columns.

As a digression, note that Shridharan and Iyengar [2023a] have shown that, in some particular cases, the number of columns of $\mathbf{A}$ can be reduced to $2^{|\mathbf{C}^*|2^{|\cup_{V\in\mathbf{C}^*}\mathrm{Pa}(V)\setminus\mathbf{C}^*|}}$. We assume that those particular cases are treated, whenever they apply and reduce costs, before our techniques are run.

5.1 COLUMN GENERATION

Column generation searches for a column at each iteration of the revised simplex method, by searching u that minimizes the reduced cost [Bertsimas and Tsitsiklis, 1997]:

$$\gamma_u - \mathbf{d} \cdot \mathbf{a}_u, \tag{8}$$

where γ_u is the uth coefficient of the objective function, $\mathbf{d}$ is the current vector of dual costs, and $\mathbf{a}_u$ is the uth column of matrix $\mathbf{A}$. We have used subscripts u because each coefficient of the objective function, and each column of $\mathbf{A}$, is fixed when we select a value u of the exogenous variable U^*. Note that the vector $\mathbf{d}$ is usually available as a call to the linear solver of choice, so we assume it is available. In our context, there are as many dual costs as there are constraints; as constraints in our problem can be indexed by the configuration $\mathbf{w}$ of $\mathbf{W}^*$, we write $\mathbf{d}_\mathbf{w}$ for each entry of $\mathbf{d}$. Likewise, the vector $\mathbf{a}_u$ has as many entries as there are constraints; so we write $\mathbf{a}_{u,\mathbf{w}}$ for the entry of $\mathbf{a}_u$ corresponding to the configuration $\mathbf{w}$ of $\mathbf{W}^*$.

Consider first the term,

$$\mathbf{d} \cdot \mathbf{a}_u = \sum_{\mathbf{w}} \mathbf{d}_\mathbf{w} \mathbf{a}_{u,\mathbf{w}},$$

where the summation runs over the values $\mathbf{w}$ of $\mathbf{W}^*$. Note that u is not fixed at this point, as we are searching for it; however, $\mathbf{w}$, for each term in the summation, is fixed.

We start by writing an implicit expression for $\mathbf{a}_u$. The strategy is to write a generic value u of U^* in binary notation as a sequence of bits; in fact, as a sequence of blocks of bits, one per mechanism f_V. The procedure is as follows.

1. Take each mechanism f_{V_i} in $\mathbf{C}^*$, for $i = 1, \ldots, |\mathbf{C}^*|$.
   (a) If f_{V_i} is a function only of U, introduce a single bit $b_1^i = f_{V_i}(U)$. That is, the output of $f_{V_i}(U)$ is simply the bit b_1^i of U.
   (b) If instead f_{V_i} is a function of U and a set of n_i variables in $\{V_1, \ldots, V_{i-1}\}$, then: for *each* one of the 2^{n_i} configurations of these variables, ordered themselves as binary numbers, introduce a bit b_j^i. Note that this step introduces 2^{n_i} bits.
2. Write $u = b_0^1 \cdots b_{2^{n_1}-1}^1 \cdots b_1^{|\mathbf{C}^*|} \cdots b_{2^{n_{|\mathbf{C}^*|}}-1}^{|\mathbf{C}^*|}$, with $n_i = 0$ when f_{V_i} does not depend on endogenous variables, for a generic value of U^*.

Example 4. Consider the quasi-Markovian model in Figure 2 (right), and focus on the c-component $\mathbf{C}^*$ associated with exogenous variable $U^* = U_1$. Order the variables in $\mathbf{W}^*$, W, X and Z, using lexicographic order (V_1 is W, V_2 is X, V_3 is Z). Now examine mechanisms associated with variables in $\mathbf{C}^*$. Mechanism f_X depends only on U_1, hence we write: $f_X(u) = b_0^2$. Mechanism f_W depends on X and Z (in this order) besides U_1, so we must code the four configurations of X, Z, ordered using binary notation:

$$\begin{array}{ll} f_W(0,0,u) = b_0^1, & f_W(0,1,u) = b_1^1, \\ f_W(1,0,u) = b_2^1, & f_W(1,1,u) = b_3^1. \end{array} \tag{9}$$

Thus a generic value u of U_1 is written as $b_0^1 b_1^1 b_2^1 b_3^1 b_0^2$. Consequently, there are $2^5 = 32$ values of U, agreeing with $\prod_{V\in\mathbf{C}^*} 2^{2^{|\mathrm{Pa}(V)|}} = 2^{2^0} \times 2^{2^2}$. □

We use these bits to build a "symbolic" version of each $\mathbf{a}_{u,\mathbf{w}}$, so as to leave $\mathbf{a}_u$ as a function of the sought for value u:

1. For each variable V_i in $\mathbf{C}^*$, take mechanism f_{V_i} and:
   (a) get the bit b_j^i that corresponds to the value of f_{V_i} evaluated at $(u, \mathbf{w})$;
   (b) if $v_i = 1$, then insert b_j^i into a list L^+;
   (c) if $v_i = 0$, then insert $(1 - b_j^i)$ into a list L^-.
2. Return $\mathbf{a}_{u,\mathbf{w}} = \prod_{b^+\in L^+} b^+ \prod_{b^-\in L^-} (1 - b^-)$.

Note that each b^+ and b^- is restricted to be integer in $\{0, 1\}$.

We must now eliminate the products of bits; we do so resorting to linear constraints. More precisely, we replace the product in the last step of the previous algorithm by the following set of $|\mathbf{C}^*| + 1$ linear constraints:

$$0 \le \mathbf{a}_{u,\mathbf{w}} \le b^+, \quad \text{for each } b^+ \in L^+,$$

$$0 \le \mathbf{a}_{u,\mathbf{w}} \le (1 - b^-), \quad \text{for each } b^- \in L^-,$$

$$1 - |\mathbf{C}^*| + \sum_{b^+\in L^+} b^+ + \sum_{b^-\in L^-} (1 - b^-) \le \mathbf{a}_{u,\mathbf{w}} \le 1.$$

Example 5. Consider again Figure 2 (right), where each u is written as bits $b_0^1 b_1^1 b_2^1 b_3^1 b_0^2$. A generic column $\mathbf{a}_u$ has an entry per configuration $\mathbf{w} = (wxz)$ of $\mathbf{W}^*$, as follows: *if* $w = b^1_{2x+z}$, *and* $x = b^2_0$, *then* $\mathbf{a}_{u,\mathbf{w}} = 1$; *otherwise*, $\mathbf{a}_{u,\mathbf{w}} = 0$. Abusing notation (equating "true" and 1, "false" and 0), we write:

$$\mathbf{a}_{u,\mathbf{w}} = (b^1_{2x+z} \leftrightarrow w) \wedge (b^2_0 \leftrightarrow x).$$

For binary variable V and any bit b, $(v \leftrightarrow b)$ is just b when $v = 1$, and is just $(1-b)$ when $v = 0$. Moreover, with binary variables we can reproduce conjunction using product. For instance if $wxz = 000$, we have $\mathbf{a}_{u,000} = (1 - b^1_0)(1 - b^2_0)$. That is, each element of the column $\mathbf{a}_{u,\mathbf{w}}$ is a product of bits (or negated bits) of u. Additional details can be found in the Supplementary Material (Section 7). □

To actually process the reduced cost (Expression (8)) within column generation, we still need to write γ_u, the coefficient of $\Pr(U^* = u)$ in the objective function, as a function of (the bits of) u. Note that $\gamma_u = \Pr(\mathbf{y}|\mathrm{do}(X = x), U^* = u)$. As described in Section 4, our proposed algorithm generates γ_u sequentially; in the end, γ_u is a summation with as many terms as there are configurations of $\mathbf{W}^* \backslash \{X\}$. Moreover, each term is a product of $|\mathbf{C}^*| - 1$ bits (or negated bits) in the binary notation for U^* (the same binary encoding discussed previously), as all mechanisms in $\mathbf{C}^*$ contribute, except the mechanism of the intervened variable.

We can thus use the same techniques discussed previously to code $\mathbf{a}_u$ to build γ_u as a function of the bits of U^*. An example should clarify the idea.

Example 6. Consider again the quasi-Markovian model in Figure 2 (right). By running the algorithm in the previous section, we obtain: $\Pr(y|\mathrm{do}(x)) = \sum_u \gamma_u \Pr(u)$, where

$$\gamma_u = \textstyle\sum_{w,z} \widehat{\Pr}(y|w,x,z)[\![f_W(x,z,u) = w]\!]\widehat{\Pr}(z|x).$$

Recall that $[\![f_W(x,z,u) = z]\!]$ is given by Expression (9). If we want $\Pr(Y = y|\mathrm{do}(X = 1))$; then γ_u is equal to:

$$\begin{array}{l}
\widehat{\Pr}(y|W=0,X=1,Z=0)\widehat{\Pr}(Z=0|X=1)(1-b^1_2)+\\
\widehat{\Pr}(y|W=0,X=1,Z=1)\widehat{\Pr}(Z=1|X=1)(1-b^1_3)+\\
\widehat{\Pr}(y|W=1,X=1,Z=0)\widehat{\Pr}(Z=0|X=1)b^1_2+\\
\widehat{\Pr}(y|W=1,X=1,Z=1)\widehat{\Pr}(Z=1|X=1)b^1_3,
\end{array}$$

a function of the bits of U_1. In a more complex example we might have to handle products of bits in the objective function, by adding linear integer constraints as before. □

By writing γ_u and $\mathbf{a}_u$ as depending on bits of U^*, we can find a value u that minimizes Expression (8). By iterating this process as usual in any implementation of column generation [Bertsimas and Tsitsiklis, 1997], we can reach $\underline{\Pr}(\mathbf{y}|\mathrm{do}(x))$ as desired.

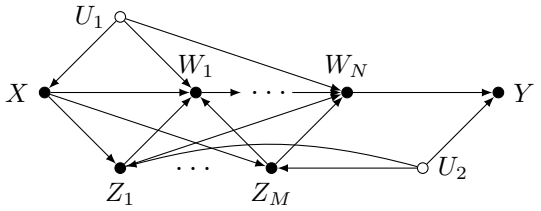

Figure 3: A parameterized expansion of the graph in Figure 2 (right). We have $|\mathbf{C}^*| = N + 1$ and $|\mathbf{W}^*| = N + M + 1$.

5.2 EXPERIMENTS

We have implemented the algorithm in the previous section,[2] relying on the Gurobi solver for master and auxiliary programs.[3] We report representative experiments here.

We start with the quasi-Markovian SCM depicted in Figure 2 (right). This SCM is based on a practical problem faced by one of the authors; namely, the evaluation of causes in a low-latency service pipeline. Binary treatment X signals activation of a newly deployed AI system. High processing requests are indicated by Z (whether or not average load exceeds, say, 80%); Z mediates the effect of X on tail latency W, and the latter may in turn trigger an incident indicated by Y. We also allow for a direct path from X to W, caused by database calls issued by the AI model. There are latent pressures on the pipeline: U_1 indicates heavy traffic induced by marketing campaigns, and U_2 refers to the degradation of an external API that affects both Z and Y.

We also consider a parameterized version of this SCM, where there are N variables W_i in the direct path between X and Y, all of them in the c-component connected with U_1, and M observed confounders Z_j, children of U_2 and parents of all W_i. The template is depicted in Figure 3. For $N = M = 1$ we obtain the graph in Figure 2 (right). No pair (M, N) can be handled by techniques by Sachs et al. [2023]. Additional details can be found in the Supplementary Material (Section 8).

We wish to compute bounds on $\Pr(y|\mathrm{do}(X = x))$. There are on the order of 2^{N2^M} optimization variables in the corresponding linear program. To run column generation, we build auxiliary linear integer programs that produce values of U_1 coded as $b^1_0 ... b^1_{2^{M+1}-1} ... b^N_{2^{M+1}-1} b^{N+1}_0$, where each block of bits $b^i_0 ... b^i_{2^{M+1}-1}$ corresponds to a variable W_i and bit b^{N+1}_0 corresponds to X.

Table 1 shows the comparison between the execution times with our column generation scheme (CG) and with just the direct linear program (LP) conveyed by Expression (7). The latter approach cannot handle several cases; as can be seen in the table, the running time of LP grows dramatically and

[2] It will be made publicly available if the paper is accepted.

[3] Gurobi version 12.0.2 (Linux): www.gurobi.com.

M	N	CG(s)	LP(s)	M	N	CG(s)	LP(s)
1	1	0.749	0.661	2	2	0.867	2494
1	2	0.271	1.55	2	3	94.5	-
1	3	1.81	106	2	4	4360	-
1	4	16.0	411	3	1	0.207	4207
1	5	1050	7174	3	2	6.64	-
2	1	0.216	4.65	3	3	3138	-

Table 1: Runs of Column Generation (CG) and direct Linear Programming (LP), in seconds, for several pairs M, N. Entries marked "-" mean that the LP solver did not finish within 3 hours of execution.

is, except for small problems, much larger than the running time of CG (all tests run in an AMD 32-cores machine). A particularly striking case is $M = 3$, $N = 1$, where CG is 20.000 times faster than LP.

5.3 BONUS: SOLUTION BY SINGLE PROGRAM

We have explored, in the previous section, an encoding for a *single* coefficient γ_u and a *single* column $\mathbf{a}_u$. However, we know that to optimize $\Pr(\mathbf{y}|\mathrm{do}(x))$ we must only find M such pairs coefficient/column — if we simultaneously find the M right columns (instead of one), we can obtain the desired probability bounds.

This leads to the following strategy:
1) Build M copies of the bits described in the previous section: for each bit b^i_j, we have M bits $b^{i,k}_j$.
2) Then use, for each $k \in \{1, \ldots, M\}$, the bits $b^{i,k}_j$ to build constraints for a column $\mathbf{a}_{u_k}$. So, we have M columns that depend on the bits.
3) Similarly, use, for each $k \in \{1, \ldots, M\}$, the bits $b^{i,k}_j$ to build a coefficient γ_{u_k}. So, we have M coefficients that depend on the bits.
4) Write down a single objective function as a sum $\sum_k \gamma_{u_k} \Pr(u_k)$ subject to $\mathbf{Ap} = \hat{\mathbf{q}}$ where $\mathbf{A}$ is now the square matrix with the M columns just built, $\mathbf{p}$ is a vector with the optimizing values $\Pr(u_k)$, and $\hat{\mathbf{q}}$ is defined as before. These expressions contains products of the form mp, where $m \in \{0, 1\}$ and p is a real (both optimizing variables). Replace each such product by a fresh variable $\alpha_{m,p}$ subject to $0 \leq \alpha_{m,p} \leq m$ and $p + m - 1 \leq \alpha_{m,p} \leq p$.
The minimum/maximum for the latter linear integer program is exactly the lower/upper bound we desire.

This strategy has been employed in probabilistic logic [Cozman and Fargoni di Ianni, 2015]. The advantage of a non-iterative strategy is simplicity of implementation. Past experience suggests that column generation tends to be faster, an issue to be verified empirically in our present context.

6 CONCLUSION

In this paper we have investigated the computation of probability bounds for quasi-Markovian SCMs subject to interventions. We described a shorter proof for Shridharan and Iyengar [2023b]'s key reduction to multilinear programming. We then proposed a new algorithm that exploits the presence of input probabilities when building linear programs in the presence of a single intervention. We presented a column generation scheme to solve such linear programs, and described an approach that computes lower or upper bounds using a single linear integer program. Our experiments showed that column generation offers significant improvements over direct linear programming.

Future work should extend our results to multiple interventions. It is also important to characterize the complexity of our algorithm, perhaps by connecting it with graph-theoretical quantities, and to explore tree decompositions or similar data structures to accelerate the construction of linear programs. Another promising avenue is to combine column generation with special cases that already reduce linear programs; for example, the cases discussed by Shridharan and Iyengar [2023a].

Looking forward, it will be valuable to combine binary (and categorical) variables with continuous ones, so as to expand practical application. That extension will likely require different techniques to handle continuous variables, even if continuous variables are restricted to say Gaussian distributions.

ACKNOWLEDGEMENTS

We thank the Center for Artificial Intelligence at Universidade de São Paulo (C4AI-USP), with support by the São Paulo Research Foundation (FAPESP grant 2019/07665-4) and by the IBM Corporation. We thank the Instituto de Ciência e Tecnologia Itaú (ICTi), for providing key funding for this work through the Centro de Ciência de Dados (C2D) at Universidade de São Paulo. D.D.M. was partially supported by CNPq grant 305136/2022-4 and FAPESP grant 2022/02937-9. F.G.C. was partially supported by CNPq grants 312180/2018-7 and 305753/2022-3. The authors also thank support by CAPES - Finance Code 001.

References

A. Balke and J. Pearl. Counterfactual probabilities: Computational methods, bounds and applications. In *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence*, pages 46–54, 1994.

A. Balke and J. Pearl. Bounds on treatment effects from studies with imperfect compliance. *Journal of the American Statistical Association*, 92:1171–1176, 1997.

Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

Fabio G. Cozman. Credal networks. *Artificial Intelligence*, 120(2):199–233, 2000.

Fabio G. Cozman and Lucas Fargoni di Ianni. Probabilistic satisfiability and coherence checking through integer programming. *International Journal Approximate Reasoning*, 58(C):57–70, 2015. ISSN 0888-613X.

Guilherme Duarte, Noam Finkelstein, Dean Knox, Jonathan Mummolo, and Ilya Shpitser. An automated approach to causal inference in discrete settings. *Journal of the American Statistical Association*, 119(547):1778–1793, 2024.

David Galles and Judea Pearl. An axiomatic characterization of causal counterfactuals. *Foundations of Science*, 3(1): 151–182, 1998.

Joseph Y. Halpern. Axiomatizing causal reasoning. *Journal of Artificial Intelligence Research*, 12:317–337, 2000.

Daphner Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Judea Pearl. *Causality*. Cambridge University Press, 2009.

Michael C. Sachs, Gustav Jonzon, Arvid Sjölander, and Erin E. Gabriel. A general method for deriving tight symbolic bounds on causal effects. *Journal of Computational and Graphical Statistics*, 32(2):567–576, 2023.

Madhumitha Shridharan and Garud Iyengar. Scalable computation of causal bounds. *Journal of Machine Learning Research*, 24(237):1–35, 2023a.

Madhumitha Shridharan and Garud Iyengar. Causal bounds in quasi-Markovian graphs. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31675–31692. PMLR, 2023b.

J. Tian. *Studies in Causal Reasoning and Learning*. PhD thesis, UCLA, 2002.

J. Tian and J. Pearl. Probabilities of causation: Bounds and identification. *Annals of Mathematics and Artificial Intelligence*, 28:287–313, 2000.

M. Zaffalon, A. Antonucci, R. Cabañas, D. Huber, and D. Azzimonti. Efficient computation of counterfactual bounds. *International Journal of Approximate Reasoning*, pages 1–24, 2024.

Marco Zaffalon, Alessandro Antonucci, and Rafael Cabañas. Structural causal models are (solvable by) credal networks. In *International Conference on Probabilistic Graphical Models*, pages 581–592. PMLR, 2020.

Junzhe Zhang, Jin Tian, and Elias Bareinboim. Partial counterfactual identification from observational and experimental data. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 26548–26558. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/zhang22ab.html`.

Multilinear and Linear Programs for Partially Identifiable Queries in quasi-Markovian Structural Causal Models (Supplementary Material)

João P. Arroyo[1] **João G. Rodrigues**[1] **Daniel Lawand**[1] **Denis D. Mauá**[1] **Junkyu Lee**[2] **Radu Marinescu**[2]
Alex Gray[3] **Eduardo R. Laurentino**[4] **Fabio G. Cozman**[1]

[1] Universidade de São Paulo, São Paulo, Brazil
[2]IBM Research – J. L.: Yorktown Heights, USA; R. M.: Ireland
[3]Centaur Institute, USA
[4]Instituto de Ciência e Tecnologia Itaú, São Paulo, Brazil

7 ADDITIONAL DETAILS ABOUT EXAMPLE 5

Here we offer details about Example 5. We start by repeating the relevant graph (left) and presenting the corresponding intervened semi-marginal graph for $\mathrm{do}(X = x)$ (right):

We take all conventions described in Example 4. That is, variables in $\mathbf{W}^*$ are ordered so that V_1 is W, V_2 is X, V_3 is Z; values of U_1 are written in binary notation as $b_0^1 b_1^1 b_2^1 b_3^1 b_0^2$; and we code mechanisms $f_W(X, Z, U_1)$ and $f_X(U_1)$ as follows:

$$f_W(0,0,u) = b_0^1, \quad f_W(0,1,u) = b_1^1, \quad f_W(1,0,u) = b_2^1, \quad f_W(1,1,u) = b_3^1, \quad f_X(u) = b_0^2.$$

As noted in Example 5, a generic column $\mathbf{a}_u$ has a entry per configuration $\mathbf{w} = (wxz)$ of $\mathbf{W}^*$,

$$\mathbf{a}_{u,\mathbf{w}} = (b^1_{2x+z} \leftrightarrow w) \wedge (b_0^2 \leftrightarrow x).$$

The following table explicitly shows the entries of $\mathbf{a}_u$:

$w = (wxz)$	$\mathbf{a}_u$, for $u = b_0^1 b_1^1 b_2^1 b_3^1 b_0^2$
000	$(b_0^1 \leftrightarrow 0) \wedge (b_0^2 \leftrightarrow 0)$
001	$(b_1^1 \leftrightarrow 0) \wedge (b_0^2 \leftrightarrow 0)$
010	$(b_2^1 \leftrightarrow 0) \wedge (b_0^2 \leftrightarrow 1)$
011	$(b_3^1 \leftrightarrow 0) \wedge (b_0^2 \leftrightarrow 1)$
100	$(b_0^1 \leftrightarrow 1) \wedge (b_0^2 \leftrightarrow 0)$
101	$(b_1^1 \leftrightarrow 1) \wedge (b_0^2 \leftrightarrow 0)$
110	$(b_2^1 \leftrightarrow 1) \wedge (b_0^2 \leftrightarrow 1)$
111	$(b_3^1 \leftrightarrow 1) \wedge (b_0^2 \leftrightarrow 1)$

For binary variable V and any bit b, $(v \leftrightarrow b)$ is just b when $v = 1$, and is just $(1 - b)$ when $v = 0$. Moreover, with binary variables we can reproduce conjunction using product. For instance if $wxz = 000$, we have $\mathbf{a}_{u,000} = (1 - b_0^1)(1 - b_0^2)$. That is, each element of the column $\mathbf{a}_{u,\mathbf{w}}$ is a product of bits (or negated bits) of u. Hence we have:

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

$w = (wxz)$	$\mathbf{a}_u$, for $u = b_0^1 b_1^1 b_2^1 b_3^1 b_0^2$
000	$(1 - b_0^1)(1 - b_0^2)$
001	$(1 - b_1^1)(1 - b_0^2)$
010	$(1 - b_2^1)b_0^2$
011	$(1 - b_3^1)b_0^2$
100	$b_0^1(1 - b_0^2)$
101	$b_1^1(1 - b_0^2)$
110	$b_2^1 b_0^2$
111	$b_3^1 b_0^2$

Note that all bits b_j^i are integer variables in $\{0, 1\}$. Then each product of bits can be turned into linear integer constraints; here are a few instances:

- $\mathbf{a}_{u,000}$ (that is, $wxz = 000$):

$$0 \leq \mathbf{a}_{u,000} \leq (1 - b_0^1), \quad 0 \leq \mathbf{a}_{u,000} \leq (1 - b_0^2), \quad 1 - b_0^1 - b_0^2 \leq \mathbf{a}_{u,000} \leq 1.$$

- $\mathbf{a}_{u,011}$ (that is, $wxz = 011$):

$$0 \leq \mathbf{a}_{u,011} \leq (1 - b_3^1), \quad 0 \leq \mathbf{a}_{u,011} \leq b_0^2, \quad b_0^2 - b_3^1 \leq \mathbf{a}_{u,011} \leq 1.$$

- $\mathbf{a}_{u,111}$ (that is, $wxz = 111$):

$$0 \leq \mathbf{a}_{u,111} \leq b_3^1, \quad 0 \leq \mathbf{a}_{u,111} \leq b_0^2, \quad b_3^1 + b_0^2 - 1 \leq \mathbf{a}_{u,111} \leq 1.$$

Hence the term $\mathbf{d} \cdot \mathbf{a}_u$ in the reduced cost (Expression (8)) is a summation with 8 terms, one per configuration of (wxz), subject to 24 linear constraints and the fact that all $b_j^i \in \{0, 1\}$.

As described in Example 6, we can use the same sort of encoding via bits for the objective function; in this particular example the objective function is itself a linear function of bits. A more complex example may exhibit products of bits (or negated) bits in the objective function.

To illustrate the latter possibility, consider the more complex graph in Figure 4 (left). We focus on the expression for the objective function, where the goal is to bound $\Pr(y|\mathrm{do}(X = x))$. Hence we have $\mathbf{C}^* = \{T, W, X\}$ and $\mathbf{W}^* = \{T, W, X, Z\}$. We adopt this lexicographic order of variables. We now write a value u of U_1 as a sequence of bits $b_0^1 b_1^1 b_0^2 b_1^2 b_2^2 b_3^2 b_0^3$, where the bits are associated with mechanisms $f_T(X, U_1)$, $f_W(X, Z, U_1)$ and $f_X(U_1)$ are as follows:

$$f_T(0, u) = b_0^1,\ f_T(1, u) = b_1^1,\ f_W(0, 0, u) = b_0^2,\ f_W(0, 1, u) = b_1^2,\ f_W(1, 0, u) = b_2^2,\ f_W(1, 1, u) = b_3^2,\ f_X(u) = b_0^3.$$

The algorithm in Section 4 produces

$$\Pr(Y = y|\mathrm{do}(X = 1)) = \sum_u \gamma_u \Pr(U_1 = u),$$

with (introducing a few auxiliary variables $\gamma_{u,\cdot}$):

$$\begin{aligned}
\gamma_u \quad = \quad & \widehat{\Pr}(y|W = 0, T = 0)\gamma_{u,1} + \widehat{\Pr}(y|W = 0, T = 1)\gamma_{u,2} + \\
& \widehat{\Pr}(y|W = 1, T = 0)\gamma_{u,3} + \widehat{\Pr}(y|W = 1, T = 1)\gamma_{u,4},
\end{aligned}$$

$$\gamma_{u,1} = \widehat{\Pr}(Z = 0|X = 1)(1 - b_1^1)(1 - b_2^2) + \widehat{\Pr}(Z = 1|X = 1)(1 - b_1^1)(1 - b_3^2),$$

$$\gamma_{u,2} = \widehat{\Pr}(Z = 0|X = 1)b_1^1(1 - b_2^2) + \widehat{\Pr}(Z = 1|X = 1)b_1^1(1 - b_3^2),$$

$$\gamma_{u,3} = \widehat{\Pr}(Z = 0|X = 1)(1 - b_1^1)b_2^2 + \widehat{\Pr}(Z = 1|X = 1)(1 - b_1^1)b_3^2,$$

$$\gamma_{u,4} = \widehat{\Pr}(Z = 0|X = 1)b_1^1 b_2^2 + \widehat{\Pr}(Z = 1|X = 1)b_1^1 b_3^2.$$

We can then collect all these expressions into a single one (as done in previous examples), or feed them separately to the appropriate linear solver. In any case, the key point here is that we have to deal with terms such as $b'b''$ or $b'(1 - b'')$ or $(1 - b')(1 - b'')$, where b' and b'' are bits that appear in the program as integer variables in $[0, 1]$. There are three cases:

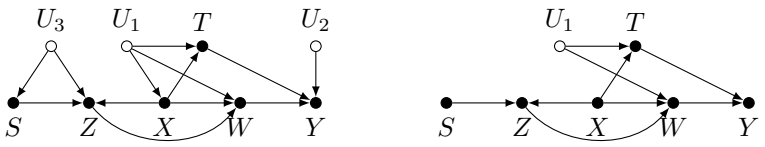

Figure 4: Left: a quasi-Markovian model. Right: the intervened semi-marginal graph for $\mathrm{do}(X = x)$.

- To handle $b'b''$, introduce a fresh optimization variable β and constraints

$$0 \leq \beta \leq b', \quad 0 \leq \beta \leq b'', \quad b' + b'' - 1 \leq \beta \leq 1.$$

- To handle $b'(1 - b'')$, introduce a fresh optimization variable β and constraints

$$0 \leq \beta \leq b', \quad 0 \leq \beta \leq 1 - b'', \quad b' - b'' \leq \beta \leq 1.$$

- To handle $(1 - b')(1 - b'')$, introduce a fresh optimization variable β and constraints

$$0 \leq \beta \leq 1 - b', \quad 0 \leq \beta \leq 1 - b'', \quad 1 - b' - b'' \leq \beta \leq 1.$$

8 ADDITIONAL DETAILS ABOUT EXPERIMENTS

We start by providing more details on the characteristics of the SCMs depicted in Figure 3. We wish to compute bounds on $\Pr(y|\mathrm{do}(X = x))$. Note that there are on the order of 2^{N2^M} optimization variables in the corresponding linear program containing values of $\Pr(U_1)$.

Write U_1 as $b^1_0...b^1_{2^{M+1}-1}...b^N_{2^{M+1}-1}b^{N+1}_0$, where each block $b^i_0...b^i_{2^{M+1}-1}$ corresponds to a variable W_i and b^{N+1}_0 corresponds to X. The subindex corresponds to the binary number associated with the vector $(W_{i-1}, Z_1, ..., Z_M)$, where Z_M is the least significant bit.

8.1 MASTER LINEAR PROGRAM

8.1.1 Objective function

Using the algorithm in Section 4, we get the objective function:

$$P(y|\mathrm{do}(X = x)) = \sum_{W_i, Z_j, U} P(y|W_N)P(Z_M, ..., Z_1|x) \prod_{i=1}^{N} P(W_i|W_{i-1}, Z_M, ..., Z_1, U)P(U). \tag{10}$$

8.1.2 Constraints

We have:

$$\sum_{u \mapsto (W_1, W_2, ..., W_N, X, Z_1, ..., Z_M)} P(u) = P(W_N, W_{N-1}, ..., W_1|X, Z_1, ..., Z_M)P(x). \tag{11}$$

8.2 AUXILIARY LINEAR INTEGER PROGRAM

8.2.1 The Costs

Note that in Equation 10, each term of the form $P(W_i|W_{i-1}, Z_1, ..., Z_M, U)$ can be expressed as a function of bits from U_1, more specifically:

$$P(W_i|W_{i-1}, Z_M, ..., Z_1, U) = b^{i*}_{Z_M + 2Z_{M-1} + ... 2^{M-1}Z_1 + 2^M W_{i-1}}. \tag{12}$$

Let us call $Z_M + 2Z_{M-1} + ...2^{M-1}Z_1 + 2^M W_{i-1} = w_i$, in which w_i is the binary number associated with the vector $(W_{i-1}, Z_1, ..., Z_M)$, i.e. the realization for parents of $W_i, i \in \{1, 2, ...N\}$ (adopt $W_0 = X$). Furthermore, define:

$$b^{i*}_{w_i} = \begin{cases} b^i_{w_i}, W_i = 1, \\ 1 - b^i_{w_i}, W_i = 0. \end{cases} \tag{13}$$

Therefore the cost can be written as:

$$\gamma_u = \sum_{Z_j, W_i} P(y|Z_N)P(Z_M, ..., Z_1|x) \prod_i b^{i*}_{w_i}. \tag{14}$$

Now we define $\prod_i b^{i*}_{w_i} = \beta_{x,w}$, where w is the binary number associated with $(W_1, ..., W_N, X, Z_1, ..., Z_M)$, i. e. the c-component and it's tail realization (note that the subindex x is redundant, however we leave it for emphasis). The reason for this will be evident in the next section.

8.2.2 The Columns

Each column a_u can be indexed by w for a specific realization and it's value can be expressed as: $a_{u,w} = b_0^{N+1*}\beta_{x,w}$, which gives:

$$a_{u,w} = \begin{cases} b_0^{N+1}\beta_{1,w}, x = 1, \\ (1 - b_0^{N+1})\beta_{0,w}, x = 0. \end{cases} \tag{15}$$

8.2.3 The Linear Program

We write a linear program in optimizing variables $\beta_{0,w}, \beta_{1,w}, a_{u,w}$, $w \in \{0, 1, ..., 2^{M+N+1} - 1\}$, a set with $\mathcal{O}(2^{M+N})$ optimizing variables. Restrictions:

- $\beta_{x,w}$:

$$\begin{aligned} &0 \leq \beta_{x,w} \leq b^{i*}_{w_i}, \forall i \in \{1, 2..., N\}, \\ &1 - N + \sum_i b^{i*}_{w_i} \leq \beta_{x,w} \leq 1. \end{aligned} \tag{16}$$

- $a_{u,w}$:

$$\begin{aligned} &0 \leq a_{u,w} \leq b_0^{N+1*}, \\ &0 \leq a_{u,w} \leq \beta_{x,w}, \\ &-1 + b_0^{N+1*} + \beta_{x,w} + \leq a_{u,w} \leq 1. \end{aligned} \tag{17}$$

Note that there are $\mathcal{O}(2^{M+N})$ constraints.

The objective function is:

$$\begin{aligned} &\sum_{w, X=1} P(y|W_N)P(Z_M, ..., Z_1|x)x_0\beta_{1,w} + \\ &\sum_{w, X=0} P(y|W_N)P(Z_M, ..., Z_1|x)(1 - x_0)\beta_{0,w} - \\ &\sum_w d_w a_{u,w}. \end{aligned} \tag{18}$$

This ensures that, for the $\beta_{x,w}$ variables, only $\beta_{x_0,w}$ has non-zero coefficient in the objective function.

8.2.4 Initialization

To obtain the first dual cost vector, we need an initial value base so we can start the column generation procedure. In our implementation we used the bigM method Bertsimas and Tsitsiklis [1997], with $M = 10^4$, with an initial basis $I_{2^{M+N+1}+1}$, i. e., the identity matrix in the size of the restrictions.