An Exact Symbolic Reduction of Linear Smart Predict+Optimize to Mixed Integer Linear Programming

Jihwan Jeong [1 2] **Parth Jaggi** [1] **Andrew Butler** [1] **Scott Sanner** [1 2]

Abstract

Predictive models are traditionally optimized independently of their use in downstream decision-based optimization. The 'smart, predict then optimize' (SPO) framework addresses this shortcoming by optimizing predictive models in order to *minimize* the final downstream decision loss. To date, several local first-order methods and convex approximations have been proposed. These methods have proven to be effective in practice, however, it remains generally unclear as to how close these local solutions are to global optimality. In this paper, we cast the SPO problem as a bi-level program and apply Symbolic Variable Elimination (SVE) to analytically solve the lower optimization. The resulting program can then be formulated as a mixed-integer linear program (MILP) which is solved to global optimality using standard off-the-shelf solvers. To our knowledge, our framework is the first to provide a globally optimal solution to the linear SPO problem. Experimental results comparing with state-of-the-art local SPO solvers show that the globally optimal solution obtains up to *two orders of magnitude reduction* in decision regret.

1. Introduction

Many problems in engineering and statistics involve both predictive forecasting and decision-based optimization. A prototypical 'predict, then optimize' framework would first fit the predictive models (for example by maximum likelihood or least-squares) and then 'plug-in' those estimates to the corresponding decision-based optimization program. However, an 'objective mismatch' (Lambert et al., 2020) can occur when the prediction and optimization are treated separately such that improved prediction accuracy does not necessarily translate into lower decision error.

Recent work advocates for training prediction models based on their downstream decision loss (Elmachtoub et al., 2020; Donti et al., 2017; Mandi & Guns, 2020; Wilder et al., 2019). In particular, Elmachtoub & Grigas (2017) propose the SPO (Smart "Predict, then Optimize") loss, which measures the decision regret on optimized decisions induced by a particular prediction model. In this paper, we consider an important subset of SPO problems, with a linear decision-based optimization program (LP) and a linear predictive model.

Optimizing the SPO loss function directly is difficult as the SPO loss is non-differentiable and non-convex. Empirical evidence demonstrates the efficacy of surrogate loss functions (Elmachtoub & Grigas, 2017; Elmachtoub et al., 2020; Grigas et al., 2021) and local optimization methods (Mandi & Guns, 2020; Wilder et al., 2019; Tan et al., 2020). However, in general, there does not exist strong theoretical justification for the observed performance of these approximate solutions and, to date, a globally optimal solution and measurement of optimality gap remains unknown. For example, Figure 1 illustrates the surfaces of the true SPO loss and the convex surrogate SPO+ loss as introduced in (Elmachtoub & Grigas, 2017). Note that the surrogate loss fails to capture the complex piecewise structure of the true loss. Notably, a slight perturbation of the optimum of the SPO+ loss can incur a huge increase in the true SPO loss.

In this paper, we present the first global optimization framework for the linear SPO problem, thus providing a benchmark and measurement of optimality gap for the aforementioned approximate techniques. We use the bi-level program formulation of SPO, with a linear upper-level objective and multiple LP $\arg\min$ instances (one per datum) in the lower-level. We apply the *symbolic case calculus* (Boutilier et al., 2001) to analytically solve the lower-level $\arg\min$ problem by reformulating the LP in a generic disjoint bilinear program form (Konno, 1975). We extend the symbolic $\arg\min$ operator (Jeong et al., 2021) to retrieve the solution to the bi-linear program and demonstrate that the general solution is a piecewise linear function of the model parameters. Finally, by substituting the lower-level solution into the upper-level

[1]Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada [2]Vector Institute, Toronto, Canada. Correspondence to: Jihwan Jeong <jhjeong@mie.utoronto.ca>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

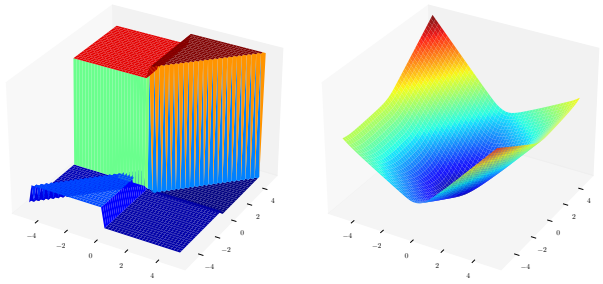

Figure 1. The true SPO loss (left) vs. its convex surrogate SPO+ loss (right) on a 2-D SPO problem from Section 3.1. The piecewise linear structure of true SPO loss hints at a possible MILP reduction.

objective, we reduce the SPO problem to a mixed-integer linear program (MILP), which can be solved to global optimality using standard off-the-shelf solvers.

Our contributions are summarized as follows: (i) firstly, we present a novel symbolic technique that enables optimally solving the SPO problem. In particular, for the class of linear prediction models and MILP with binary variables, we provide the first solution approach to optimally minimizing the SPO loss; (ii) we demonstrate a novel application of the symbolic case calculus to transform one optimization problem (a bi-level form) to another equivalent problem (a MILP); and (iii) lastly, we benchmark existing approximate methods to measure their optimality gap in three domains: noisy shortest path, cost-sensitive classification, and energy cost aware scheduling. Our experimental results show that our global solution obtains *up to two orders of magnitude reduction in decision regret* in comparison to existing approximate solutions.

2. Related Work

Recently there has been a growing body of research on data-driven optimization and the relative merits of decoupled versus integrated predictive decision-making (see for example (Ban & Rudin, 2019; Bertsimas et al., 2019; Bertsimas & Kallus, 2020; Elmachtoub & Grigas, 2017; Elmachtoub et al., 2020; Grigas et al., 2021)). Our methodology focuses on the 'smart predict, then optimize' (SPO) framework, proposed by Elmachtoub & Grigas (2017), which minimizes decision error through the custom SPO loss function (defined later in (2)). The SPO loss is Fisher consistent with least-squares under mild assumptions and can lead to improved overall decision accuracy. To date, optimizing the SPO loss by gradient methods is challenged by non-convexity and the absence of continuity. Instead the authors propose minimizing a convex surrogate loss function, SPO+, which provides an upper bound on the global solution.

Related, Gould et al. (2016) study bi-level optimization problems in a general setting and provide closed-form expressions for the gradient and Hessian under various constraint assumptions. Since then, a plethora of research and methodology has been developed that integrates differentiable optimization layers (DOLs) in an end-to-end trainable neural network (Agrawal et al., 2019; Blondel et al., 2021). Notably the work of Agrawal et al. (2020) and Amos & Kolter (2021) provide general frameworks for learning convex optimization programs and lay the groundwork for backpropagation by implicit differentiation. Indeed, despite the local nature of these solutions, recent applications of DOLs advocate strongly for a fully integrated estimation approach (see for example (Amos et al., 2019; Butler & Kwon, 2021a;b; Donti et al., 2017; Uysal et al., 2021)).

Most relevant to our framework, is the work of Wilder et al. (2019), Mandi & Guns (2020) and Tan et al. (2020) who consider a bi-level formulation for learning LPs and MILPs from optimal decisions. In all cases, however, the proposed solutions are locally optimal and the quality of the local solution with respect to the global solution remains unknown. Butler & Kwon (2021b) consider a bi-level problem with lower-level quadratic programs and provide analytical globally optimal solutions for the unconstrained and equality constrained cases, whereas the inequality constrained case is solved by gradient descent. To our knowledge, our symbolic SPO to MILP reduction framework is the first to provide a globally optimal solution to the linear bi-level SPO program.

3. Problem Description

We consider mixed-integer programs (Wolsey, 1998):

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} && (1)\\ \text{s.t.} \quad & \mathbf{a}_j^\top \mathbf{x} \leq b_j, && \forall j \in J \\ & \mathbf{x} \geq 0;\ x_k \in \{0,1\} && \forall k \in K \end{aligned}$$

where J, K are index sets of constraints and binary variables (respectively), $\mathbf{x} = (x_1, \ldots, x_{n_x})$, $\mathbf{c} \in \mathbb{R}^{n_x}$, and $\mathbf{a}_j \in \mathbb{R}^{n_x}$ $\forall j \in J$. We denote $\mathcal{X}$ as the feasible region of $\mathbf{x}$.

In the SPO framework, some coefficients of $\mathbf{c}$ are unknown (we assume all $\mathbf{c}$ are unknown for ease of exposition), and thus they have to be estimated by a linear prediction model parameterized by $\boldsymbol{\theta} \in \mathbb{R}^{n_x \times p}$ given a dataset $\mathcal{D} = \{(\varphi^{(n)}, \mathbf{c}^{(n)})\}_{n=1}^N$. That is, given a feature $\varphi \in \mathbb{R}^p$, the model outputs the prediction $\bar{\mathbf{c}} = \boldsymbol{\theta}\varphi$, such that we can now solve the above MILP with the cost $\bar{\mathbf{c}}$ rather than $\mathbf{c}$.

Elmachtoub & Grigas (2017) propose the SPO loss that can directly measure how inferior model predictions are — in terms of their induced decision costs — compared to the optimal costs that would be attained had we known the coefficients. More concretely, the SPO loss $l_{\text{spo}}(\bar{\mathbf{c}}, \mathbf{c})$ is

$$l_{\text{spo}}(\bar{\mathbf{c}}, \mathbf{c}) = \mathbf{c}^\top \mathbf{x}^*(\bar{\mathbf{c}}) - \mathbf{c}^\top \mathbf{x}^*(\mathbf{c}) \qquad (2)$$

where $\mathbf{x}^*(\mathbf{c}) \in \arg\min_{\mathbf{x}\in\mathcal{X}} \mathbf{c}^\top\mathbf{x}$ is an optimal solution to (1) with the cost $\mathbf{c}$ (similarly for $\mathbf{x}^*(\bar{\mathbf{c}})$).[1]

With the SPO loss defined as such, we note that the SPO problem can be written in the following bi-level form:

$$\min_{\boldsymbol{\theta}} \quad \frac{1}{N}\sum_{n=1}^{N}\left[\mathbf{c}^{(n)\top}\mathbf{x}^*(\bar{\mathbf{c}}^{(n)}) - \mathbf{c}^{(n)\top}\mathbf{x}^*(\mathbf{c}^{(n)})\right] \tag{3}$$

$$\text{s.t. } \mathbf{x}^*(\bar{\mathbf{c}}^{(n)}) = \arg\min_{\bar{\mathbf{x}}\in\mathcal{X}} \underbrace{(\boldsymbol{\theta}\boldsymbol{\varphi}^{(n)})}_{\bar{\mathbf{c}}^{(n)}}{}^\top\bar{\mathbf{x}}, \qquad \forall n \tag{4}$$

where we want to find the optimal model parameters $\boldsymbol{\theta}^*$ such that $\mathbb{E}_{(\boldsymbol{\varphi}^{(n)},\mathbf{c}^{(n)})\sim\mathcal{D}}[l_{\text{spo}}(\boldsymbol{\theta}\boldsymbol{\varphi}^{(n)},\mathbf{c}^{(n)})]$ is minimized. Crucially, the lower-level problem (4) has a generic form $\mathbf{x}^*(\bar{\mathbf{c}}) = \arg\min_{\bar{\mathbf{x}}\in\mathcal{X}} \bar{\mathbf{c}}^\top\bar{\mathbf{x}}$, which is a disjoint bilinear program (Konno, 1975).[2]

The non-convexity and non-differentiability of (3) make it challenging to obtain optimal model parameters $\boldsymbol{\theta}^*$ (Figure 1). That said, we provide the first exact optimal solution approach by analytically solving the lower optimization to reduce (3) to a single-level problem. We show that the lower-level solution is a piecewise linear function of $\boldsymbol{\theta}$; hence, substituting the solutions to the upper-level reduces the overall problem to a MILP as hinted at by Figure 1 (left).

3.1. A Worked Example

To foreshadow the general methodology that we explore in this paper, we first demonstrate a fully detailed worked example of reducing the SPO problem to a MILP. This will serve as guidance once we proceed to derive the more general algorithmic solution in subsequent sections.

Example 3.1. Consider lower-level LP, $\min_{\bar{\mathbf{x}}\in\mathcal{X}} \bar{\mathbf{c}}^\top\bar{\mathbf{x}}$ with feasible region $\mathcal{X}=\{(\bar{x}_1,\bar{x}_2): \bar{x}_1+\bar{x}_2 \le 2, \bar{x}_2 \le 1, \bar{\mathbf{x}} \ge 0\}$ and estimated cost $\bar{\mathbf{c}} = (\bar{c}_1,\bar{c}_2)^\top = \boldsymbol{\theta}\boldsymbol{\varphi}$.

We seek to analytically solve the lower-level problem (4) by applying symbolic variable elimination (SVE) (Sanner & Abbasnejad, 2012). Without loss of generality, we 'min-out' $\bar{x}_1$ first. Observe that when $\bar{x}_1$ is minimized, $\bar{\mathbf{c}}$ and $\bar{x}_2$ are considered free variables, allowing us to treat the bilinear objective as linear in $\bar{x}_1$. The minimum, therefore, must occur at a boundary value of $\bar{x}_1$, obtained from the constraints specified in $\mathcal{X}$.

Concretely, we compare the objective values corresponding to $\bar{x}_1 = \bar{x}_1^{lb}$ and $\bar{x}_1 = \bar{x}_1^{ub}$ to determine the minimum w.r.t. $\bar{x}_1$, where $\bar{x}_1^{lb} = 0$ and $\bar{x}_1^{ub} = 2 - \bar{x}_2$ are obtained from $\mathcal{X}$. Since $\bar{x}_2 \le 1$ then $(2-\bar{x}_2) > 0$ and we get the following:

$$\min_{\bar{x}_1\in\mathcal{X}} \bar{c}_1\bar{x}_1 + \bar{c}_2\bar{x}_2 = \min\left(\bar{c}_2\bar{x}_2,\ \bar{c}_1(2-\bar{x}_2) + \bar{c}_2\bar{x}_2\right) \tag{5}$$

$$= \begin{cases} (Case1)\ \bar{c}_1 \le 0: & \bar{c}_1(2-\bar{x}_2) + \bar{c}_2\bar{x}_2,\ (x_1' = \bar{x}_1^{ub}) \\ (Case2)\ \bar{c}_1 > 0: & \bar{c}_2\bar{x}_2,\ (x_1' = \bar{x}_1^{lb}) \end{cases}$$

Note we have *annotated* the solution x_1' associated with each case. Observe that when we min-out $\bar{x}_1$ then equation (5) reduces to a conditional constrained optimization program.

We now proceed to min-out $\bar{x}_2$ from (5), noting that we can treat each case as an LP over $\bar{x}_2$. Since the case conditionals ($\bar{c}_1 \le 0$ and $\bar{c}_1 > 0$) in (5) do not constrain $\bar{x}_2$, we get $0 \le \bar{x}_2 \le 1$ directly from the domain bounds over $\bar{x}_2$ in $\mathcal{X}$.

For $Case1$, the objective value evaluates to $2\bar{c}_1$ if $\bar{x}_2 = 0$ and $\bar{c}_1 + \bar{c}_2$ if $\bar{x}_2 = 1$. Therefore, the minimum is obtained by conditionally comparing the two objectives:

$$\min_{0\le\bar{x}_2\le 1} \bar{c}_1(2-\bar{x}_2) + \bar{c}_2\bar{x}_2 = \begin{cases} \bar{c}_1 \le \bar{c}_2: & 2\bar{c}_1,\ (x_2^* = 0) \\ \bar{c}_1 > \bar{c}_2: & \bar{c}_1 + \bar{c}_2,\ (x_2^* = 1) \end{cases}$$

Conversely, for $Case2$, the objective is $\bar{c}_2$ and 0 when $\bar{x}_2 = 1$ and $\bar{x}_2 = 0$, respectively. Hence, we obtain:

$$\min_{0\le\bar{x}_2\le 1} \bar{c}_2\bar{x}_2 = \begin{cases} \bar{c}_2 \le 0: & \bar{c}_2,\ (x_2^* = 1) \\ \bar{c}_2 > 0: & 0,\ (x_2^* = 0) \end{cases}$$

Here, note that we are able to determine the optimal solution x_2^* for each case as annotations. Combining the results with the conditions in (5), we get:

$$x_2^*(\bar{\mathbf{c}}) = \begin{cases} (\bar{c}_1 \le \bar{c}_2): & 0 \\ (\bar{c}_1 > \bar{c}_2) \wedge (\bar{c}_2 \le 0): & 1 \\ (\bar{c}_1 > \bar{c}_2) \wedge (\bar{c}_2 > 0): & 0 \end{cases} \tag{6}$$

For x_1' in (5), however, notice that one solution is represented as a function of $\bar{x}_2$ ($x_1' = 2 - \bar{x}_2$). Substituting the optimal $x_2^*(\bar{\mathbf{c}})$ values in (6) into x_1' yields the optimal $x_1^*(\bar{c})$:

$$x_1^*(\bar{\mathbf{c}}) = \begin{cases} (\bar{c}_1 \le 0) \wedge (\bar{c}_1 \le \bar{c}_2 - 2): & 2 \\ (\bar{c}_1 \le 0) \wedge (\bar{c}_1 > \bar{c}_2 - 2): & 1 \\ (\bar{c}_1 > 0): & 0 \end{cases} \tag{7}$$

Finally, observe that we have obtained an analytical solution $\mathbf{x}^*(\bar{\mathbf{c}})$ as two piecewise linear functions of $\bar{c}_1$ and $\bar{c}_2$.

Therefore, for each training example in dataset $\mathcal{D}$, the cases in (6) and (7) reduce the lower-level program to a set of linear inequality constraints on $\boldsymbol{\theta}$; one for each value of $\boldsymbol{\varphi}^{(n)}$. It follows then that the overall bi-level program (3) reduces to a MILP over model parameters $\boldsymbol{\theta}$. □

[1] Note there exists a pathological solution $\boldsymbol{\theta} = \mathbf{0}$ because then any $\mathbf{x} \in \mathcal{X}$ makes the SPO loss zero. As such, the original SPO work introduced an unambiguous version to handle this issue and the non-uniqueness of $\mathbf{x}^*(\bar{\mathbf{c}})$ in general. However, other gradient-based works (Mandi & Guns, 2020; Wilder et al., 2019) often directly use the l_{spo} as in (2). See Section 5.2 for our approach.

[2] We write $\bar{\mathbf{c}}$ whenever we want to denote the generic cost coefficients of the lower optimization, whereas the predicted coefficients for a specific datum is denoted as $\bar{\mathbf{c}}^{(n)}$.

4. Symbolic Case Calculus

We now cover the symbolic case representation and calculus (Boutilier et al., 2001; Sanner et al., 2011) underpinning our exact solution to the linear SPO problem that extends the example of Section 3.1 to the general case in Section 5.

4.1. Case Representation

We represent symbolic functions in *case form*:

$$f = \begin{cases} \phi_1 : & f_1 \\ \vdots & \vdots \\ \phi_k : & f_k \end{cases} \tag{8}$$

Here, ϕ_i (called a *partition*) are logical formulae, which can include arbitrary logical $(\wedge, \vee, \neg)$ combinations of linear inequalities $(\geq, >, \leq, <)$ over continuous variables (called a *conditional*). We assume that the set of partitions $\{\phi_1, \ldots, \phi_k\}$ disjointly and exhaustively partition the domain of the variables such that f is well-defined. We restrict f_i (a *function value*) to be linear or bilinear. When a function value is bilinear in $\bar{\mathbf{x}}$ and $\bar{\mathbf{c}}$, it is of the form of $\mathbf{p}^\top \bar{\mathbf{c}} + \bar{\mathbf{c}}^\top Q \bar{\mathbf{x}} + \mathbf{r}^\top \bar{\mathbf{x}} + s$ for some coefficients $\mathbf{p}, Q, \mathbf{r}$ and s. ϕ_i is dubbed "disjointly linear" if each and every conditional of it consists only of either $\bar{\mathbf{c}}$ or $\bar{\mathbf{x}}$. Further, we restrict ϕ_i to be disjointly linear if f has bilinear f_i.

We refer to functions with linear ϕ_i and f_i as linear piecewise linear (LPWL). Functions with disjointly linear ϕ_i and bilinear f_i are dubbed disjointly linear piecewise bilinear (LPWB). These specifications and restrictions are critical in analyzing which case functions are closed under which operations.

4.2. Case Operators

Unary case operations on f in (8) such as scalar multiplication $\alpha \cdot f$ $(\alpha \in \mathbb{R})$ or negation $-f$ are simply applied to the function value f_i for every partition ϕ_i. We can also define *binary operations* between two case functions by taking the cross-product of the logical partitions from the two case statements and performing the operation on the resulting paired partitions, e.g., the "cross-sum" $\oplus$ of two cases is:

$$\begin{cases} \phi_1 : & f_1 \\ \phi_2 : & f_2 \end{cases} \oplus \begin{cases} \psi_1 : & g_1 \\ \psi_2 : & g_2 \end{cases} = \begin{cases} \phi_1 \wedge \psi_1 : & f_1 + g_1 \\ \phi_1 \wedge \psi_2 : & f_1 + g_2 \\ \phi_2 \wedge \psi_1 : & f_2 + g_1 \\ \phi_2 \wedge \psi_2 : & f_2 + g_2 \end{cases}$$

Likewise, we perform $\ominus$ by subtracting function values per each pair of partitions. Observe that LPWL and LPWB functions are closed under $\oplus$ and $\ominus$.

Next, we define symbolic *case min(max)* as:

$$\text{casemin}\left(\begin{cases} \phi_1 : f_1 \\ \phi_2 : f_2 \end{cases}, \begin{cases} \psi_1 : g_1 \\ \psi_2 : g_2 \end{cases} \right) = \begin{cases} \phi_1 \wedge \psi_1 \wedge \boldsymbol{f_1 > g_1} : g_1 \\ \phi_1 \wedge \psi_1 \wedge \boldsymbol{f_1 \leq g_1} : f_1 \\ \phi_1 \wedge \psi_2 \wedge \boldsymbol{f_1 > g_2} : g_2 \\ \phi_1 \wedge \psi_2 \wedge \boldsymbol{f_1 \leq g_2} : f_1 \\ \vdots \qquad\qquad\qquad\quad \vdots \end{cases}$$

wherein the resulting partitions also include the comparison of the associated function values f_i and g_j to determine $\min(f_i, g_j)$ (highlighted in bold). The casemin of more than two case functions is straightforward since the operator is associative. Crucially, LPWL functions are closed under casemin (max), but LPWB functions are not because $f_i \leq g_j$ can introduce bilinear or jointly linear conditionals.

Another important operation is *symbolic substitution*. This operation takes a set σ of variables and their substitutions, e.g., $\sigma = \{y/x_1, z/(x_1 - x_2)\}$ where the LHS of '/' represents the substitution variable and the RHS of '/' is the expression being substituted in. Then, we write the substitution operation on f_i with σ as $f_i\sigma$. All substitutions in this paper will remain closed-form since we substitute linear expressions of $\{\bar{x}_j\}_{j \neq i}$ variables into $\bar{x}_i$, which clearly preserves the LPWL and LPWB properties. Infeasible partitions resulting from case operators may be removed.

4.3. The Lower-level Problem in Case Form

We now show the case perspective of the reduction of the lower-level SPO problem to a set of LPWL functions corresponding to an $\arg\min$ solution $\mathbf{x}^*(\bar{\mathbf{c}}^{(n)})$. In fact, this procedure can be *performed independently of the dataset* such that we need only solve the lower-level problem once to get a generic $\arg\min$ solution denoted by $\mathbf{x}^*(\bar{\mathbf{c}})$.

First, we need to represent the lower optimization in case form. We define an LPWB function $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ as below,

$$f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}) = \begin{cases} \bar{\mathbf{x}} \in \mathcal{X} : & \bar{\mathbf{c}}^\top \bar{\mathbf{x}} \\ \bar{\mathbf{x}} \notin \mathcal{X} : & \infty \end{cases} \tag{9}$$

Then, the $\min$ problem in the lower-level (4) is equivalent to $\min_{\bar{\mathbf{x}}} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ up to a well-defined symbolic $\min$ operator, which we discuss in detail in the following section. Note that the non-empty feasible set $\mathcal{X}$ is encoded as one partition, whose value corresponds to the objective; we use ∞ to indicate infeasibility because all feasible partitions should have finite values as we are minimizing f_{low}.

5. Exact MILP Reduction of the Linear SPO

5.1. The Exact Lower-level $\arg\min$ Solution

Similar to the procedure described in Example 3.1, the lower-level symbolic $\arg\min$ solution can be obtained from the *annotations* of the variables, which are generated during

the symbolic min operation on $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$. Although previous work has introduced the symbolic min and arg min operator for LPWL functions (Zamani et al., 2012; Jeong et al., 2021), their results do not extend to an LPWB function in (9). We fill this gap by generalizing min and arg min to work with LPWB functions. Specifically, we show that an LPWB function remains closed-form under the min operator, allowing us to retain both annotations and the final arg min solution in an LPWL form.

As mentioned in Section 3.1, we view the symbolic min operation from the perspective of SVE, where we 'min-out' one variable at a time. That is, $\min_{\bar{\mathbf{x}}} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ is equivalent to $\min_{\bar{x}_{n_x}} \cdots \min_{\bar{x}_1} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$. Thus, we treat $\bar{\mathbf{c}}$ and $\bar{\mathbf{x}}_{2:n_x}$ (i.e., $\{\bar{\mathbf{x}} \backslash \bar{x}_1\}$) as symbolic free variables when $\bar{x}_1$ is minimized out, making $\bar{\mathbf{c}}^\top \bar{\mathbf{x}}$ linear in $\bar{x}_1$.

Now, $\min_{\bar{x}_1} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ proceeds by observing the following: (i) the minimum occurs at lower or upper bounds of $\bar{x}_1$ due to the linearity. (ii) The partition $\bar{\mathbf{x}} \in \mathcal{X}$ defines the bounds of $\bar{x}_1$, which are LPWL functions of $\bar{\mathbf{x}}_{2:n_x}$. For instance, if $[\bar{x}_1 + \bar{x}_2 \leq 2] \wedge [0 \leq \bar{x}_1] \wedge [\bar{x}_1 \leq 2]$, then $\bar{x}_1^{lb} = 0$ and $\bar{x}_1^{ub} = \text{casemin}(2 - \bar{x}_2, 2) = \begin{cases} \bar{x}_2 \leq 0\text{: } 2 \\ \bar{x}_2 > 0\text{: } 2 - \bar{x}_2 \end{cases}$. (iii) To determine at which bound $\bar{\mathbf{c}}^\top \bar{\mathbf{x}}$ attains the minimum, we substitute the bounds ($\bar{x}_1^{ub}$ and $\bar{x}_1^{lb}$) to $\bar{\mathbf{c}}^\top \bar{\mathbf{x}}$ and compare the resulting values. We formally summarize this process by

$$\min_{\bar{x}_1} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}) = \text{casemin}(\bar{\mathbf{c}}^\top \bar{\mathbf{x}}\ \sigma_1^{ub},\ \bar{\mathbf{c}}^\top \bar{\mathbf{x}}\ \sigma_1^{lb}), \quad (10)$$

where $\sigma_1^{ub} = \{\bar{x}_1 / \bar{x}_1^{ub}\}$ and $\sigma_1^{lb} = \{\bar{x}_1 / \bar{x}_1^{lb}\}$ define the corresponding substitution operations.

Since $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ is an LPWB function, the casemin operation in (10) requires comparing bilinear function values, which can lead to jointly linear or bilinear partitions. Fortunately, Lemma 5.1 shows that only disjointly linear conditionals are added after the casemin operation (all proofs are deferred to Appendix A).

Lemma 5.1. *Consider an LPWB function $f(\mathbf{y}, \mathbf{z})$ with $\mathbf{y} \in \mathbb{R}^{n_y}$ and $\mathbf{z} \in \mathbb{R}^{n_z}$, which has a single feasible partition:*

$$f(\mathbf{y}, \mathbf{z}) = \begin{cases} \phi(\mathbf{y}) \wedge \psi(\mathbf{z}) : & \mathbf{p}^\top \mathbf{y} + \mathbf{y}^\top Q \mathbf{z} + \mathbf{r}^\top \mathbf{z} + s \\ \textit{otherwise} : & \infty \end{cases}$$

where $\mathbf{p} \in \mathbb{R}^{n_y}$, $\mathbf{r} \in \mathbb{R}^{n_z}$, $Q \in \mathbb{R}^{n_y \times n_z}$, and $s \in \mathbb{R}$ are some constant coefficients. Then, $\min_{y_i} f$ results in an LPWB function of $\{y_j\}_{j \neq i}$ and $\mathbf{z}$ for any $i \in \{1, \ldots, n_y\}$.

As per Lemma 5.1, we obtain an LPWB function with n_2 partitions denoted as $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x})$ once we *eliminate* $\bar{x}_1$ from $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$,

$$f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x}) = \begin{cases} \phi_2^1(\bar{\mathbf{x}}_{2:n_x}) \wedge \psi_2^1(\bar{\mathbf{c}}) : & f_2^1(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x}),\ a_1^1 \\ \vdots & \vdots \\ \phi_2^{n_2}(\bar{\mathbf{x}}_{2:n_x}) \wedge \psi_2^{n_2}(\bar{\mathbf{c}}) : & f_2^{n_2}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x}),\ a_1^{n_2} \end{cases}$$

Importantly for our purpose, once we compute $\text{casemin}(\bar{\mathbf{c}}^\top \bar{\mathbf{x}}\ \sigma_1^{ub}, \bar{\mathbf{c}}^\top \bar{\mathbf{x}}\ \sigma_1^{lb})$, we can determine which of $\bar{x}_1^{ub}$ or $\bar{x}_1^{lb}$ has produced the min for partition j_2 ($j_2 \in [1, n_2]$). We then *annotate* this partition with the corresponding $\bar{x}_1$ value, denoted as $a_1^{j_2}$ above.[3]

After annotating each resulting partition with the associated $\bar{x}_1$, we can compute the arg min of $\bar{x}_1$ by noting $\arg\min_{\bar{x}_1} f_{low} = \arg(\min_{\bar{x}_1} f_{low})$. That is, we apply the arg operator to $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x})$ to retrieve the annotations of $\bar{x}_1$ noted during the min operation. Specifically, the arg operator *replaces* the function values with their annotations in all partitions. In other words,

$$x_1' = \arg\min_{\bar{x}_1} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}) = \arg_{\bar{x}_1} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x}) \quad (11)$$
$$= \begin{cases} \phi_2^1(\bar{\mathbf{x}}_{2:n_x}) \wedge \psi_2^1(\bar{\mathbf{c}}) : & a_1^1 \\ \vdots & \vdots \\ \phi_2^{n_2}(\bar{\mathbf{x}}_{2:n_x}) \wedge \psi_2^{n_2}(\bar{\mathbf{c}}) : & a_1^{n_2} \end{cases}$$

which is an LPWL function. Furthermore, $a_1^{j_2}$ is an LPWL case function of $\bar{\mathbf{x}}_{2:n_x}$ but not of $\bar{\mathbf{c}}$, since it has been derived from $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ with disjointly linear partitions in $\bar{\mathbf{c}}$ and $\bar{\mathbf{x}}$.

It has been straightforward to eliminate $\bar{x}_1$ and obtain its optimal solution because $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ has a single feasible partition, unlike $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x})$. We now need to solve

$$\min_{\bar{x}_2} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x}) \quad (12)$$
$$= \min_{\bar{x}_2} \underset{j_2=1,\ldots,n_2}{\text{casemin}} \begin{cases} \phi_2^{j_2}(\bar{\mathbf{x}}_{2:n_x}) \wedge \psi_2^{j_2}(\bar{\mathbf{c}}) : & f_2^{j_2}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x}) \\ \text{otherwise} : & \infty \end{cases}$$
$$= \underbrace{\underset{j_2=\{1,\ldots,n_2\}}{\text{casemin}}}_{(12a)} \underbrace{\min_{\bar{x}_2} \begin{cases} \phi_2^{j_2}(\bar{\mathbf{x}}_{2:n_x}) \wedge \psi_2^{j_2}(\bar{\mathbf{c}}) : & f_2^{j_2}(\bar{\mathbf{x}}_{2:n_x}, \bar{\mathbf{c}}) \\ \neg\phi_2^{j_2}(\bar{\mathbf{x}}_{2:n_x}) \vee \psi_2^{j_2}(\bar{\mathbf{c}}) : & \infty \end{cases}}_{(12b)}$$

where we have used the disjointness of partitions and the commutative property of min and casemin to get to (12a,b). Equation (12) shows that we can min-out $\bar{x}_2$ from each partition (12b), followed by casemin of the results (12a).

Applying Lemma 5.1 allows us to symbolically compute the min over one partition in (12b), as was done for $\bar{x}_1$, thus producing an LPWB function. However, it is unclear whether the result will remain closed-form after the casemin step in (12a). To answer this question, we first construct the result that we get an LPWL function of $\bar{\mathbf{c}}$ once we eliminate *all* $\bar{\mathbf{x}}$ variables from $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ in Proposition 5.2. From this, we deduce Corollary 5.3 that is useful for our purpose.

Proposition 5.2. *The* min *problem, $\min_{\bar{\mathbf{x}}} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$, can be exactly reduced to an LPWL function of $\bar{\mathbf{c}}$ in closed-form.*

[3] Note that annotations themselves can be case functions. However for simplicity, we treat them as function values in (11), obtained after merging the partitions with those of $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x})$.

Corollary 5.3. *Consider the application of the symbolic* min *operator to* $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ *with the variable ordering* $\bar{x}_1, \ldots, \bar{x}_{n_x}$. *Define* $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{i+1:n_x}) := \min_{\bar{x}_i} \cdots \min_{\bar{x}_1} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ *for* $i = 1, \ldots, n_x - 1$. *Then,* $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{i+1:n_x})$ *is an LPWB function of* $\bar{\mathbf{c}}$ *and* $\mathbf{x}_{i+1:n_x}$.

Hence, the casemin operation in (12a) leaves an LPWB form unchanged. Furthermore, the same procedure can be applied to all remaining variables $\bar{x}_3, \ldots, \bar{x}_{n_x}$. In other words, an LPWB function is closed under the SVE, which implies that we can get the solution of $\bar{x}_i \ \forall i$ in LPWL form via annotations and their retrieval with the arg operator. Note that during the casemin operation, annotations should simply follow their associated function values.

Completing the symbolic arg min procedure requires one final step, the *backward substitution*. To see this, observe (11) where the solution x'_1 depends on $\bar{\mathbf{x}}_{2:n_x}$ (as is the case in Example 3.1). Likewise, the solutions x'_i for $i \in [1, n_x)$ are functions of $\bar{\mathbf{x}}_{i+1:n_x}$ and $\bar{\mathbf{c}}$.[4] As such, we need to be rid of the remaining $\bar{\mathbf{x}}_{i+1:n_x}$ from x'_i in order to obtain the optimal solution $x^*_i(\bar{\mathbf{c}})$ that is a function of $\bar{\mathbf{c}}$ only. To this end, we first substitute the solution $x^*_{n_x}(\bar{\mathbf{c}})$ into x'_{n_x-1} for all occasions of $\bar{x}_{n_x}$: i.e., $x^*_{n_x-1}(\bar{\mathbf{c}}) = x'_{n_x-1}\sigma_{n_x}$ with $\sigma_{n_x} = \{\bar{x}_{n_x}/x^*_{n_x}(\bar{\mathbf{c}})\}$. We then proceed to substitute $x^*_{n_x}(\bar{\mathbf{c}})$ and $x^*_{n_x-1}(\bar{\mathbf{c}})$ into x'_{n_x-2}, and we repeat this process all the way back to the solution x'_1. We call the proposed overall symbolic arg min solution algorithm *SymArgMin* (summarized in Algorithm 2 in the Appendix).

Proposition 5.4 provides important properties regarding the symbolic solution we obtain, which leads to the exact closed-form MILP reduction of the SPO problem (Corollary 5.5).

Proposition 5.4. *Let* $\mathcal{X} = \{\bar{\mathbf{x}} : \mathbf{a}_j^\top \bar{\mathbf{x}} \leq b_j, \ j = 1, \ldots\}$ *denote a non-empty feasible set over* $\bar{\mathbf{x}}$. *Then,* SymArgMin *(Algorithm 2) outputs* $\mathbf{x}^*(\bar{\mathbf{c}}) \in \arg\min_{\bar{\mathbf{x}} \in \mathcal{X}} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ *that is a set of LPWL functions of* $\bar{\mathbf{c}}$. *Assume (i)* $\bar{\mathbf{c}} \neq \mathbf{0}$ *and (ii) there does not exist* $k \neq 0$ *such that* $\bar{\mathbf{c}} = k\mathbf{a}_j \ \forall j$. *Then, the solution is the unique solution of the lower-level problem.*

Corollary 5.5. *The SPO problem in* (3) *reduces to a MILP over variable* $\boldsymbol{\theta}$.

Throughout the derivation, we have used the generic coefficients $\bar{\mathbf{c}}$ for a model prediction, which allows us to solve the lower-level only once. Now given a dataset $\mathcal{D}$, we *instantiate* the predictions from $\bar{\mathbf{c}}$ by substituting in the data in its place. For example, we get $\bar{\mathbf{c}}^{(n)} = \boldsymbol{\theta}\varphi^{(n)}$ for a feature $\varphi^{(n)}$; thus $\mathbf{x}^*(\bar{\mathbf{c}}^{(n)}) = \mathbf{x}^*(\bar{\mathbf{c}})\sigma^{(n)}$ with $\sigma^{(n)} = \{\bar{\mathbf{c}}/\boldsymbol{\theta}\varphi^{(n)}\}$. Since each solution for a sample is essentially a piecewise linear function of $\boldsymbol{\theta}$, we see the SPO problem is a MILP (Corollary 5.5), which can be solved using off-the-shelf modern MILP solvers such as Gurobi (Gurobi Optimization, LLC, 2021).

[4] Naturally, $x'_{n_x}(\bar{\mathbf{c}})$ only contains $\bar{\mathbf{c}}$ since all $\bar{x}$ variables have already been eliminated at this point, so we say $x^*_{n_x}(\bar{\mathbf{c}}) = x'_{n_x}(\bar{\mathbf{c}})$.

Furthermore, when the mild assumptions of Proposition 5.4 hold, we can guarantee that we can optimally solve the SPO problem. As discussed in Section 3, $\boldsymbol{\theta} = \mathbf{0}$ (hence, $\bar{\mathbf{c}} = \mathbf{0}$) is a pathological solution which trivially optimizes the SPO loss. Elmachtoub & Grigas (2017) get around this issue by proposing a worst-case based SPO loss, $l_{\text{spo}}(\bar{\mathbf{c}}, \mathbf{c}) := \max_{\mathbf{x}^* \in \mathcal{X}^*(\bar{\mathbf{c}})} \mathbf{c}^\top \mathbf{x}^*(\bar{\mathbf{c}}) - \mathbf{c}^\top \mathbf{x}^*(\mathbf{c})$ where $\mathcal{X}^*(\bar{\mathbf{c}})$ is the set of optimal solutions to the lower-level problem. Other gradient-based works (Mandi & Guns, 2020; Wilder et al., 2019) have simply ignored this issue, since they are only guaranteed to find a local optimum, and it is unlikely that the final $\boldsymbol{\theta}$ they obtain falls exactly at $\mathbf{0}$ given random initialization.

In our case, we first note that the optimal solution $\mathbf{x}^*(\mathbf{c}^{(n)})$ of the lower-level LP is invariant to positive scaling ($\alpha > 0$) of $\mathbf{c}^{(n)}$. Hence, regardless of how we scale $\mathbf{c}^{(n)}$, we get the same set of solutions.[5] Thus, we can enforce a constraint of the form $|\sum_{i,k} \boldsymbol{\theta}_{ik}| = \alpha$ to effectively remove the pathological solution from the feasible set of $\boldsymbol{\theta}$. Potential complications of this approach are (a) that there can still exist a non-trivial optimal solution that satisfies $\sum_{i,k} \boldsymbol{\theta}_{ik} = 0$ with $\boldsymbol{\theta}_{ik} \neq \mathbf{0}$ and (b) that $\bar{\mathbf{c}}^{(n)} = \mathbf{0}$ can happen despite $\boldsymbol{\theta} \neq \mathbf{0}$. However, it is very unlikely that a non-zero $\boldsymbol{\theta}$ would be able to satisfy $\bar{\mathbf{c}}^{(n)} = \mathbf{0}$ for all n. Alternatively, we can enforce the bias values of $\boldsymbol{\theta}$ to be set fixed at some constants, e.g., at the average of a given dataset, which can be a reasonable option especially if we normalize the dataset to have zero mean. In our experiments, we find both approaches work well and result in small decision regret. Future work can examine the impacts of these additional constraints more carefully.

Finally, notice that $\bar{\mathbf{c}}$ appears only in the conditionals of $\mathbf{x}^*(\bar{\mathbf{c}})$ but not in its function values. Since the annotations and the solution are originated from bound analysis over a series of LPWB functions $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{i:n_x})$, it is impossible for $\bar{\mathbf{c}}$ to end up in a bound expression of $\bar{x}_i \ \forall i$. From a structural point of view, since the lower-level problem is a MILP, we need to have $\mathbf{x}^*(\bar{\mathbf{c}}^{(n)}) \in V(\text{Conv}(\mathcal{X}))$ where $V(\text{Conv}(\mathcal{X}))$ is the set of vertices of the convex hull of $\mathcal{X}$. In other words, the generic arg min solution $\mathbf{x}^*(\bar{\mathbf{c}})$ encodes a subset of vertices of $V(\text{Conv}(\mathcal{X}))$; upon instantiation of the coefficient $\bar{\mathbf{c}}^{(n)} = \boldsymbol{\theta}\varphi^{(n)}$, the vertex that is located the furthest from the direction of $\bar{\mathbf{c}}^{(n)}$ is then selected for $\mathbf{x}^*(\bar{\mathbf{c}}^{(n)})$. As $\text{Conv}(\mathcal{X})$ does not depend on $\bar{\mathbf{c}}$, neither do the function values of $\mathbf{x}^*(\bar{\mathbf{c}})$ and $\mathbf{x}^*(\bar{\mathbf{c}}^{(n)})$.

5.2. Building the MILP Model for Optimally Solving the SPO Problem

In this section, we discuss how we can efficiently substitute data into the closed-form solution $\mathbf{x}^*(\bar{\mathbf{c}})$ and subsequently build a MILP model over $\boldsymbol{\theta}$.

[5] The second assumption of Proposition 5.4 guarantees that we get a unique solution.

To this end, first note that in practice, maintaining case representations with explicit partitions can be prohibitively expensive. Hence, we use a more compact representation known as Extended Algebraic Decision Diagrams (XADD) (Sanner et al., 2011). Due to space constraints, we defer the detailed discussion to Appendix B. However, we remark that an XADD is a directed acyclic graph (DAG), which can be exponentially more compact than an equivalent case or tree data structure. When it comes to representing an LPWL function with an XADD, it suffices to restrict decision nodes to have linear inequalities and leaf values to linear expressions.

To help understand the model formulation procedure, recall the $\arg\min$ solution (7) of $x_1^*(\bar{\mathbf{c}})$ we have obtained in Example 3.1. We need to substitute the solution into the term $c_1 x_1^*(\bar{\mathbf{c}}^{(n)})$ in the upper-level objective. Here for simplicity, assume $\varphi^{(n)} = 1$ and hence $\bar{\mathbf{c}} = (\theta_1, \theta_2)$. Then, we need two binary variables β_1, β_2 which encode the logical relationships. For example, $\beta_1 = 0$ (or 1) $\Rightarrow \bar{c}_1 \leq 0$ (or $\bar{c}_1 > 0$) and $\beta_2 = 0$ (or 1) $\Rightarrow \bar{c}_1 \leq \bar{c}_2 - 2$ (or $\bar{c}_1 > \bar{c}_2 - 2$). Similarly, the function values are modeled as following

$$\beta_1 = 0 \Rightarrow x_1^* = y_1,\ \beta_1 = 1 \Rightarrow x_1^* = 0$$
$$\beta_2 = 0 \Rightarrow y_1 = 2,\ \beta_2 = 1 \Rightarrow y_1 = 1$$

where we have defined y_1 which is associated with the conditional $\bar{c}_1 \leq \bar{c}_2 - 2$. These constraints (called indicator constraints) can be effectively handled by modern MILP solvers like Gurobi (Gurobi Optimization, LLC, 2021).

Algorithm 1 summarizes our approach to optimally solving the linear SPO problem. We first get the generic analytic solution of the lower-level problem of SPO as a set of LPWL functions of $\bar{\mathbf{c}}$ (SymArgMin in Algorithm 2, Appendix). This step involves representing the optimization problem as a case function, followed by SVE of the decision variables while annotating the solutions along the way. Then, we apply the $\arg$ operator to retrieve these annotated solutions. Finally, we perform the backward substitution to eliminate all $\bar{\mathbf{x}}_{i+1:n_x}$ variables existing in the solution x_i', giving us $\mathbf{x}^*$, a set of LPWL functions of $\bar{\mathbf{c}}$. Subsequently, we substitute the data into the symbolic solution and build the corresponding MILP model. In the end, we have a single MILP model over $\boldsymbol{\theta}$ which is equivalent to the SPO problem, which we solve to obtain the optimal model parameters $\boldsymbol{\theta}^*$.

Algorithm 1 EMSPO: Exact MILP Reduction of SPO

Input: $\mathcal{X}, \mathcal{D} = \{(\varphi^{(n)}, \mathbf{c}^{(n)})\}_{n=1}^N$
Output: The optimal $\boldsymbol{\theta}^*$
Initialize symbolic variables $\bar{\mathbf{x}}, \bar{\mathbf{c}}$
$x_1^*(\bar{\mathbf{c}}), \ldots, x_{n_x}^*(\bar{\mathbf{c}}) \leftarrow$ SymArgMin$(\mathcal{X}, \bar{\mathbf{x}}, \bar{\mathbf{c}})$
for $i = 1$ **to** n_x **do**
  $\{x_i^{(n)*}\}_{n=1}^N \leftarrow$BuildMILPfromCase$(x_i^*(\bar{\mathbf{c}}), N)$
end for
for $n = 1$ **to** N **do**
  Add a constraint $\bar{c}_i(n) = (\boldsymbol{\theta}\varphi^{(n)})_i$
end for
Set objective as $\frac{1}{N}\sum_{n=1}^N \sum_{i=1}^{n_x} x_i^{(n)*} \cdot c_i^{(n)}$
Optimize the model to get $\boldsymbol{\theta}^*$

6. Empirical Evaluation

Given our novel globally optimal **EMSPO** (**E**xact **M**ILP Reduction of **SPO**) solution, we can now use it to evaluate the quality of existing SOTA approximate SPO solvers on some predict-then-optimize problems to address two key research questions: (RQ1) How does the level of nonlinearity in the data (i.e., model misspecification w.r.t. a linear predictive model) affect the decision quality of approximate solvers? (RQ2) How does decision quality of approximate solvers vary with the size of a dataset? The baseline models we compare with are the **TwoStage** model (i.e., simple least squares), **SPO+** (Elmachtoub & Grigas, 2017), **IntOpt** (Mandi & Guns, 2020), and **QPTL** (Wilder et al., 2019). The last two methods were developed for use within an end-to-end deep learning framework, but they can be easily ported to the linear setting. We followed the original implementations given in Mandi & Guns (2020). Our code is available at `https://github.com/jihwan-jeong/xaddpy`.

In the evaluation, we compare the true decision loss computed during training based on (2). For baselines, we use the grid search to tune hyperparameters such as the learning rate (Appendix C). All experiments are done for 5 random seeds. For EMSPO, we show the costs of MILP solutions obtained by Gurobi with a time limit of 420 minutes to reach an optimality gap of 5%. Below, we describe the domains we have used for evaluation, with more detailed descriptions provided in Appendix C.

Noisy shortest path Similar to Elmachtoub et al. (2020), we consider a shortest path problem on a 3×3 grid network consisting of 12 edges directing towards either north or east. The travel times of the edges are unknown, and they are estimated using 5-dimensional features. We follow the same data generation procedure given in Elmachtoub et al. (2020), in which we can control the level of noise ($\bar{\epsilon}$) and nonlinearity (deg) of the generated datasets. We consider the datasets of sizes 50, 100, and 200; the polynomial nonlinearity of 1, 2, 5, and 10; and the noise level of 0 and 0.25.

Energy cost aware scheduling We consider a reduced version of the resource-constrained day-ahead job scheduling problem presented in Mandi & Guns (2020). In this problem, a machine has a resource capacity constraint, and there are multiple tasks, each of which is specified with time and resource constraints. We must schedule the tasks with-

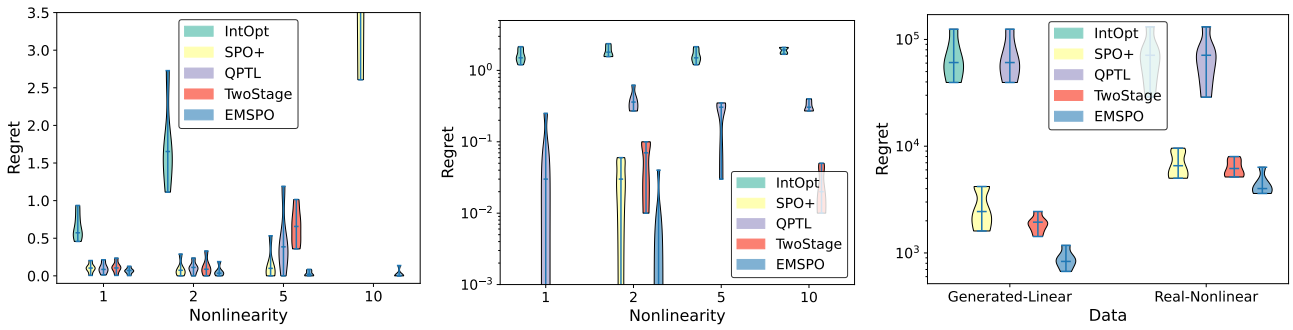

Figure 2. The decision regret subject to different levels of (non)linearity: Shortest path (left); Classification (middle); Energy scheduling (Right). The middle and the right figures are shown in log scale.

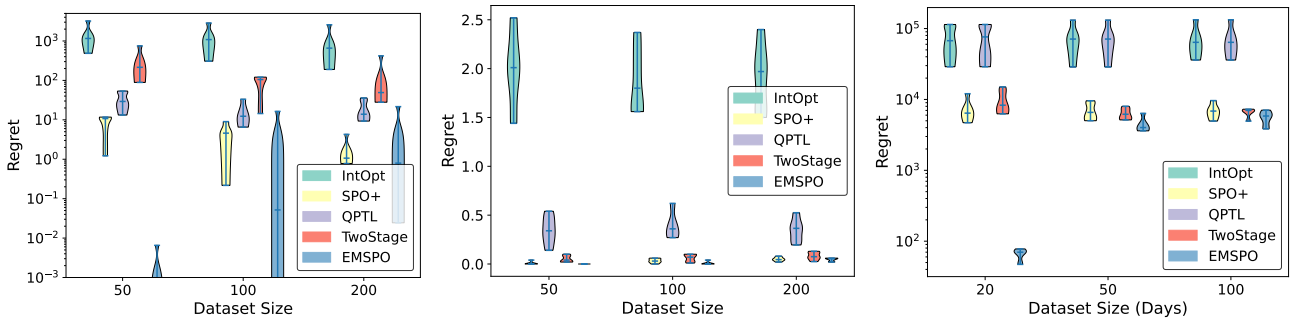

Figure 3. The decision regret subject to different dataset sizes on three domains: Shortest path (left); Classification (middle); Energy scheduling (Right). Notice the log scale in the plots on the both sides.

out knowing day-ahead energy prices to minimize energy consumption cost; prices are predicted by a linear model.

Cost-sensitive multi-class classification The cost-sensitive classification problem is introduced in Liu & Grigas (2021), wherein the feasible region is the unit simplex $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^5 : \mathbf{x} \geq 0, \sum_{i=1}^{5} x_i = 1\}$. Unlike a typical classification task where incorrect prediction incurs the same error loss regardless of what the estimated label is, different costs are incurred by differently estimated labels. Specifically, a predicted label i incurs the cost $c_i = |i - \text{lab}|$ for $i = 1, \ldots, 5$ when lab is the true label. This cost function is unknown and has to be estimated based on features. We follow Liu & Grigas (2021) for data generation to control the noise ($\bar{\epsilon}$) and nonlinearity (deg) of the dataset. We use datasets of size 50, 100, and 200.

RQ1: Does increased nonlinearity in the dataset affect decision performance and optimality gap? Figure 2 shows the decision regret compared to our optimal EMSPO with varying data nonlinearity. We controlled the nonlinearity parameter deg in (left, middle). We generated a synthetic dataset using a linear model for the energy scheduling problem (right); the real dataset is expected to have higher nonlinearity. In the shortest path problem, we clearly see that as the nonlinearity increases, the approximate methods have increasingly higher and more variable regret values. When the nonlinearity $deg = 10$, we observe large optimality gaps from the baselines compared to our optimal solution. Although SPO+ achieved the best regret, it is far from optimal. In energy scheduling, we observe that approximate solvers tend to achieve much better regret with the synthetic linear data than with the original data. However, the optimality gap has increased because now EMSPO can attain even lower regret value. This pattern is not as evident in the classification example. However, unlike other approximate approaches, we observe that our method constantly achieves near-zero regrets that are possible in the linear case (recall that our Gurobi MILP solver settings allow up to a bounded 5% optimality gap).

RQ2: How does dataset size impact performance? Figure 3 shows the impact of the dataset size on the SPO loss. In the shortest path problem, we observe that all approximate methods suffer from large optimality gaps for a sparse dataset of size 50. As more data become available, the performances of approximate solvers tend to increase across the domains. The best approximate solver is SPO+, which works well across different dataset sizes and approaches optimality as the sizes increase. Regardless of a dataset size, TwoStage appears to have solved the classification problem with small regrets, potentially suggesting the simplicity of the underlying decision problem. As more data become available, the optimal cost that can be achieved with a linear

model tends to increase, narrowing the gap between the best approximate and optimal solutions.

Noticeably, reformulating the linear SPO problem to a MILP provides another significant advantage; that is, we can obtain the lower bound of the SPO loss by a MILP solver, which gives some sense of how good the current incumbent solution is. Further, we remark that it is also possible to combine neural network based approaches such as IntOpt or QPTL with EMSPO. That is, we can pretrain a neural network, after which its internal parameters will be set fixed. Then, we can optimize for only the parameters in the last layer using EMSPO. This may help reduce the decision loss when the dataset contains a high level of nonlinearity, which we leave as future work.

In Figure 2 and Figure 3, we see that more recent approaches (QPTL and IntOpt) consistently perform worse than the convex surrogate approach (SPO+). This suggests that SPO+ works robustly across the board. For SPO+, the decision regrets decrease for larger datasets, which is conforming with the Fisher consistent property of the method. On the other hand, the poor performance of QPTL and IntOpt may be attributed to the small sizes of the datasets we have used as well as the low complexity of these datasets.

Overall, our experimental results show that our global solution obtains *up to two orders of magnitude reduction in decision regret* in comparison to existing approximate solutions and yields best performance in the sparse data regime.

7. Conclusion

We propose the first globally optimal solution approach to the linear SPO problem by symbolically eliminating the lower level of its bi-level optimization formulation using novel closed-form bilinear generalizations of Symbolic Variable Elimination. This analytical solution yields a piecewise linear case structure that can then be reduced to a MILP and solved to (bounded) global optimality with off-the-shelf solvers. We evaluated on three predict-then-optimize problems and showed that our globally optimal solution obtains up to *two orders of magnitude* reduction in decision regret compared to existing approximate solutions when there is sparse data or high model misspecification due to nonlinearity in the underlying data. We observe that SPO+ routinely performs among the best approximate methods when benchmarked according to our global MILP-based solutions, but the large optimality gap in some cases suggests the need for continued research on tractable SPO approximations. Finally, we stress that our work can lead to interesting future analysis on the generalization of SPO solvers since now we can obtain the model parameters that exactly minimize the SPO loss in the train set.

Acknowledgements

This project was supported by a Natural Sciences and Engineering Research Council (NSERC) Discovery Grant.

References

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, pp. 9562–9574. Curran Associates, Inc., 2019.

Agrawal, A., Barratt, S., Boyd, S., Busseti, E., and Moursi, W. M. Differentiating through a cone program, 2020. URL http://arxiv.org/abs/1703.00443.

Amos, B. and Kolter, J. Z. Optnet: Differentiable optimization as a layer in neural networks, 2021. URL http://arxiv.org/abs/1703.00443.

Amos, B., Rodriguez, I. D. J., Sacks, J., Boots, B., and Kolter, J. Z. Differentiable mpc for end-to-end planning and control, 2019. URL http://arxiv.org/abs/1810.13400.

Bahar, R. I., Frohm, E. A., Gaona, C. M., Hachtel, G. D., Macii, E., Pardo, A., and Somenzi, F. Algebraic decision diagrams and their applications. *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, pp. 188–191, 1993.

Ban, G.-Y. and Rudin, C. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67(1):90–108, 2019.

Bertsimas, D. and Kallus, N. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020.

Bertsimas, D., Dunn, J., and Mundru, N. Optimal prescriptive trees. *INFORMS Journal on Optimization*, 1(2):164 – 183, 2019.

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-Lopez, F., Pedregosa, F., and Vert, J.-P. Efficient and modular implicit differentiation, 2021. URL http://arxiv.org/abs/2105.15183.

Boutilier, C., Reiter, R., and Price, B. Symbolic dynamic programming for first-order MDPs. In *IJCAI-01*, pp. 690–697, Seattle, 2001.

Butler, A. and Kwon, R. Covariance estimation for risk-based portfolio optimization: an integrated approach. *Journal of Risk*, December 2021a.

Butler, A. and Kwon, R. H. Integrating prediction in mean-variance portfolio optimization, 2021b. URL http://arxiv.org/abs/2102.09287.

Donti, P., Amos, B., and Kolter, J. Z. Task-based end-to-end model learning in stochastic optimization. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 5484 – 5494. Curran Associates, Inc., 2017.

Elmachtoub, A. and Grigas, P. Smart "predict, then optimize". *Management Science*, 10 2017. doi: 10.1287/mnsc.2020.3922.

Elmachtoub, A. N., Liang, J. C. N., and McNellis, R. Decision trees for decision-making under the predict-then-optimize framework, 2020. URL http://arxiv.org/abs/2003.00360.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization, 2016. URL http://arxiv.org/abs/1607.05447.

Grigas, P., Qi, M., Zuo-Jun, and Shen. Integrated conditional estimation-optimization, 2021. URL http://arxiv.org/abs/2110.12351.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL https://www.gurobi.com.

Ifrim, G., O'Sullivan, B., and Simonis, H. Properties of energy-price forecasts for scheduling. In *International Conference on Principles and Practice of Constraint Programming*, pp. 957–972. Springer, 2012.

Jeong, J., Jaggi, P., and Sanner, S. Symbolic dynamic programming for continuous state mdps with linear program transitions. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI-21)*, Online, 2021.

Konno, H. A Bilinear Programming: Part II. Applications of Bilinear Programming. *Technical Report*, 1975.

Lambert, N., Amos, B., Yadan, O., and Calandra, R. Objective mismatch in model-based reinforcement learning. volume 120 of *Proceedings of Machine Learning Research*, pp. 761–770. PMLR, 10–11 Jun 2020.

Liu, H. and Grigas, P. Risk bounds and calibration for a smart predict-then-optimize method. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=pSitk34qYit.

Mandi, J. and Guns, T. Interior point solving for lp-based prediction+optimisation, 2020. URL http://arxiv.org/abs/2010.13943.

Sanner, S. and Abbasnejad, E. Symbolic variable elimination for discrete and continuous graphical models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):1954–1960, Sep. 2012.

Sanner, S., Delgado, K. V., and de Barros, L. N. Symbolic dynamic programming for discrete and continuous state mdps. In *Proceedings of the 27th Conference on Uncertainty in AI (UAI-2011)*, Barcelona, 2011.

Simonis, H., O'Sullivan, B., Mehta, D., Hurley, B., and Cauwer, M. D. CSPLib problem 059: Energy-cost aware scheduling. http://www.csplib.org/Problems/prob059.

Tan, Y., Terekhov, D., and Delong, A. Learning linear programs from optimal decisions, 2020. URL http://arxiv.org/abs/2006.08923.

Uysal, A. S., Li, X., and Mulvey, J. M. End-to-end risk budgeting portfolio optimization with neural networks, 2021. URL http://arxiv.org/abs/2107.04636.

Wilder, B., Dilkina, B., and Tambe, M. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1658–1665, July 2019. doi: 10.1609/aaai.v33i01.33011658.

Wolsey, L. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998. ISBN 9780471283669. URL https://books.google.ca/books?id=x7RvQgAACAAJ.

Zamani, Z., Sanner, S., and Fang, C. Symbolic dynamic programming for continuous state and action mdps. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI-12)*, Toronto, Canada, 2012.

Appendix

A. Proofs

In this section, we prove the lemmas and propositions provided in the main text.

Firstly, we restate Lemma 5.1 below for convenience.

Lemma A.1. *Consider an LPWB function $f(\mathbf{y}, \mathbf{z})$ with $\mathbf{y} \in \mathbb{R}^{n_y}$ and $\mathbf{z} \in \mathbb{R}^{n_z}$, which has a single feasible partition:*

$$f(\mathbf{y}, \mathbf{z}) = \begin{cases} \phi(\mathbf{y}) \wedge \psi(\mathbf{z}): & \mathbf{p}^\top \mathbf{y} + \mathbf{y}^\top Q \mathbf{z} + \mathbf{r}^\top \mathbf{z} + s \\ \textit{otherwise}: & \infty \end{cases}$$

where $\mathbf{p} \in \mathbb{R}^{n_y}$, $\mathbf{r} \in \mathbb{R}^{n_z}$, $Q \in \mathbb{R}^{n_y \times n_z}$, and $s \in \mathbb{R}$ are some constant coefficients. Then, $\min_{y_i} f(\mathbf{y}, \mathbf{z})$ results in an LPWB function of $\{y_j\}_{j \neq i}$ and $\mathbf{z}$ for any $i \in \{1, \ldots, n_y\}$.

Proof of Lemma A.1. Without loss of generality, we prove the case for $i = 1$. As stated in equation (10), $\min_{y_1} f(\mathbf{y}, \mathbf{z})$ can be computed by $\operatorname{casemin}(f(\mathbf{y}, \mathbf{z})\sigma^{lb}, f(\mathbf{y}, \mathbf{z})\sigma^{ub})$, where $\sigma^{lb} = \{y_1 / y_1^{lb}\}$ and $\sigma^{ub} = \{y_1 / y_1^{ub}\}$ define the operations that substitute the lower and upper bound (respectively) of y_1 into y_1 in the function value $\mathbf{p}^\top \mathbf{y} + \mathbf{y}^\top Q \mathbf{z} + \mathbf{r}^\top \mathbf{z} + s$. The bound values are computed from the partition $\phi(\mathbf{y})$, and let us denote the lower bound as $l(\mathbf{y}_{2:n_y})$ and the upper bound as $u(\mathbf{y}_{2:n_y})$. Observe that these bounds are LPWL functions of $y_2, \ldots, y_{n_y}$ variables.

To compute the casemin, define a case function $h : \mathbb{R}^{(n_y - 1) \times n_z} \mapsto \mathbb{R}$ as $h(\mathbf{y}_{2:n_y}, \mathbf{z}) := f(\mathbf{y}, \mathbf{z})\sigma^{ub} - f(\mathbf{y}, \mathbf{z})\sigma^{lb}$. Then, we note that for cases when $h \geq 0$, the value from $f(\mathbf{y}, \mathbf{z})\sigma^{lb}$ is selected, whereas the upper bound substitution is chosen when $h \leq 0$. In other words, the casemin can be written as follows:

$$\operatorname{casemin}(f(\mathbf{y}, \mathbf{z})\sigma^{lb}, f(\mathbf{y}, \mathbf{z})\sigma^{ub}) = \begin{cases} h(\mathbf{y}_{2:n_y}, \mathbf{z}) \geq 0: & f(\mathbf{y}, \mathbf{z})\sigma^{lb} \\ h(\mathbf{y}_{2:n_y}, \mathbf{z}) < 0: & f(\mathbf{y}, \mathbf{z})\sigma^{ub} \end{cases} \tag{13}$$

Let J, K be the index sets associated with the partitions of $l(\mathbf{y}_{2:n_y})$ and $u(\mathbf{y}_{2:n_y})$, respectively. We now consider any pair of indices from the two sets: (j, k) $\forall j \in J, \forall k \in K$. The function values of $h(\mathbf{y}_{2:n_y}, \mathbf{z})$ can then be seen as $f(\mathbf{y}, \mathbf{z})\sigma_k - f(\mathbf{y}, \mathbf{z})\sigma_j$ with $\sigma_k = \{y_1 / u_k\}$ and $\sigma_j = \{y_1 / l_j\}$. We then note that the substitution operation does not affect terms that do not include y_1 such that they cancel out from the function values when we subtract $f(\mathbf{y}, \mathbf{z})\sigma_j$ from $f(\mathbf{y}, \mathbf{z})\sigma_k$. Hence, we can factorize the function value of $h(\mathbf{y}_{2:n_y}, \mathbf{z})$ derived from l_j and u_k as follows:

$$h_{j,k}(\mathbf{y}_{2:n_y}, \mathbf{z}) = \big(u_k - l_j\big) \left[\mathbf{p}_1 + \sum_{r=1}^{n_z} z_r Q_{r,1} \right] \geq 0 \tag{14}$$

Here, the second factor is derived from the terms multiplied to y_1 in the original function value of $f(\mathbf{y}, \mathbf{z})$. Then, we can write the bilinear conditional $[h(\mathbf{y}_{2:n_y}, \mathbf{z}) \geq 0]$ as $[\mathbf{p}_1 + \sum_{r=1}^{n_z} z_r Q_{r,1} \geq 0]$ given that the upper bound u_k of y_1 should be greater than or equal to its lower bound l_j $\forall j, k$. Then, we can see that for this (j, k) pair, we can express the casemin with disjointly linear conditions and bilinear function values, or an LPWB function. As any pairs of $j \in J$ and $k \in K$ give the same result, we can conclude that we get an LPWB function from $\min_{y_1} f(\mathbf{y}, \mathbf{z})$. □

Now, we prove Proposition 5.2 which is restated below for convenience.

Proposition A.2. *The* min *problem, $\min_{\bar{\mathbf{x}}} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$, can be exactly reduced to an LPWL function of $\bar{\mathbf{c}}$ in closed-form.*

Proof of Proposition 5.2. The proof relies on inductive reasoning as we show how each $\bar{x}_i$ can be eliminated in turn from $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$, yielding an LPWB closed-form (intermidiately) and a final LPWL form (ultimately) once all $\bar{\mathbf{x}}$ have been eliminated.

Firstly, remember we have already established that we get an LPWB function $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x})$ after minimizing out $\bar{x}_1$. Thus, we can start analysis from (12), which is restated below:

$$\min_{\bar{\mathbf{x}}} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}) = \min_{\bar{x}_{n_x}} \cdots \min_{\bar{x}_2} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{2:n_x}) = \min_{\bar{x}_{n_x}} \cdots \left[\operatorname*{casemin}_{j_2 = \{1, \ldots, n_2\}} \underbrace{\min_{\bar{x}_2} \begin{cases} \phi_2^{j_2}(\bar{\mathbf{x}}_{2:n_x}) \wedge \psi_2^{j_2}(\bar{\mathbf{c}}): & f_2^{j_2}(\bar{\mathbf{x}}_{2:n_x}, \bar{\mathbf{c}}) \\ \neg \phi_2^{j_2}(\bar{\mathbf{x}}_{2:n_x}) \vee \psi_2^{j_2}(\bar{\mathbf{c}}): & \infty \end{cases}}_{(12b)} \right] \tag{15}$$

where we have noted in the brackets that we can eliminate $\bar{x}_2$ from a partition j_2 and combine the results from all partitions ($j_2 = 1, \dots, n_2$) via casemin. Lemma A.1 shows that the $\min_{\bar{x}_2}$ part will output an LPWB function. Denote this resulting LPWB function from partition j_2 as $g_{j_2}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{3:n_x})$. Then, we again use the commutative property of min and casemin operators to obtain the following:

$$\min_{\bar{\mathbf{x}}} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}) = \operatorname*{casemin}_{j_2=\{1,\dots,n_2\}} \left[\min_{\bar{x}_{n_x},\dots,\bar{x}_4} \left(\min_{\bar{x}_3} g_{j_2}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{3:n_x}) \right) \right] \tag{16}$$

At this point, we can observe the repeating pattern: that is, we eliminate a variable ($\bar{x}_3$ in this case) from an LPWB function (g_{j_2}), which is decomposed into $\min_{\bar{x}_3}$ and the casemin. At each step of the min operation, Lemma A.1 assures that we get an LPWB function as an output. We then change the order of the min and casemin operators and proceed to $\bar{x}_4$.

Finally, at the last iteration when we eliminate $\bar{x}_{n_x}$, we note that the bounds of $\bar{x}_{n_x}$ can be determined as a case function whose function values are all scalar. This is because (i) we have maintained the disjointness between $\bar{\mathbf{c}}$ and $\bar{\mathbf{x}}$ variables in the partitions and (ii) none other $\bar{x}$ variables are left at this stage. This leaves us to compute the expanded casemin operations stacked from $\bar{x}_2, \dots,$. As all operands of the casemin operation are LPWL functions of $\bar{\mathbf{c}}$, we get an LPWL function of $\bar{\mathbf{c}}$ as the final outcome (note that the closedness of the casemin operator with respect to LPWL functions has been exploited elsewhere, e.g., in Sanner et al. (2011)). □

Based on Proposition 5.2, we can deduce Corollary 5.3. Note that Proposition 5.2 uses the commutative property of the min and casemin operators and Lemma A.1 to conclude that the *final* output of SVE on $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ is an LPWL function. On the other hand, Corollary 5.3 asserts that all the intermediate case functions retain the LPWB form. As discussed in the main text, this allows us to keep track of annotations of each symbolic variable, being eliminated at each iteration, since the LPWB operand of the min operator guarantees that the annotations are LPWL functions.

Corollary A.3 (Corollary 5.3 restated)**.** *Consider the application of the symbolic* min *operator to* $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ *with the variable ordering* $\bar{x}_1, \dots, \bar{x}_{n_x}$*. Define* $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{i+1:n_x}) := \min_{\bar{x}_i} \cdots \min_{\bar{x}_1} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ *for* $i = 1, \dots, n_x - 1$*. Then,* $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}}_{i+1:n_x})$ *is an LPWB function of* $\bar{\mathbf{c}}$ *and* $\mathbf{x}_{i+1:n_x}$*.*

Proof sketch of Corollary 5.3. The proof is by contradiction. We first assume that there exists an intermediate case function which does not have the LPWB form. This means that the case function has conditionals that are jointly linear, bilinear, or generally nonlinear, or it has function values other than linear or bilinear form. Then, as we proceed to compute $\min_{\bar{x}_3}, \dots,$ more and more nonlinear expressions will appear in the intermediate results and in the final reduced function of $\bar{\mathbf{c}}$. We can then immediately see that this violates Proposition 5.2 which states that we have to get an LPWL function once we eliminate all $\bar{\mathbf{x}}$ variables from $f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$.

□

For clarity, we include Proposition 5.4 here.

Proposition A.4. *Let* $\mathcal{X} = \{\bar{\mathbf{x}} : \mathbf{a}_j^\top \bar{\mathbf{x}} \leq b_j,\ j = 1, \dots\}$ *denote a non-empty feasible set over* $\bar{\mathbf{x}}$*. Then,* SymArgMin *(Algorithm 2) outputs* $\mathbf{x}^*(\bar{\mathbf{c}}) \in \arg\min_{\bar{\mathbf{x}} \in \mathcal{X}} f_{low}(\bar{\mathbf{c}}, \bar{\mathbf{x}})$ *that is a set of LPWL functions of* $\bar{\mathbf{c}}$*. Assume (i)* $\bar{\mathbf{c}} \neq \mathbf{0}$ *and (ii) there does not exist* $k \neq 0$ *such that* $\bar{\mathbf{c}} = k\mathbf{a}_j\ \forall j$*. Then, the solution is the unique solution of the lower-level problem.*

Proof of Proposition 5.4. The first part is evident as Algorithm 2 surely returns one symbolic solution of $\bar{\mathbf{x}}$ given non-empty $\mathcal{X}$.

To see how the additional assumptions provide the uniqueness of the solution, we first consider an LP formulated as $\min_{\mathbf{x} \in \mathcal{X}} \mathbf{c}^\top \mathbf{x}$ with $\mathcal{X} = \{\mathbf{x} :\ \mathbf{a}_j^\top \mathbf{x} \leq b_j,\ 1 \leq j \leq m\}$, $\mathbf{a}_j \in \mathbb{R}^{n_x}$, and $b_j \in \mathbb{R}$. Then, we know that there exists one and only solution if none of the coefficient vectors $\mathbf{a}_j$ is parallel to the cost vector $\mathbf{c}$ for $\mathbf{c} \neq \mathbf{0}$ (n.b. the reverse does not have to hold). That is, if there is no $k \neq 0$ such that $\mathbf{a}_j = k\ \mathbf{c}\ \forall j$, then we have a unique solution to this problem. Moving on to our problem setting, the cost vector of the lower-level problem is $\bar{\mathbf{c}} = \boldsymbol{\theta}\varphi^{(n)}$ for the nth sample. Hence for $k \neq 0$, if $\boldsymbol{\theta}\varphi^{(n)} \neq k\mathbf{a}_j\ \forall j, n$, then there is only one solution to the problem. Therefore, the one we obtain by Algorithm 2 should be the unique solution. □

Algorithm 2 SymArgMin: Symbolic $\arg\min$

Input: non-empty feasible set $\mathcal{X}$, symbolic $\bar{\mathbf{x}}$ and $\bar{\mathbf{c}}$
Output: symbolic $\arg\min$ solution $\mathbf{x}^*(\bar{\mathbf{c}})$
Decide the variable ordering $i = 1, \ldots, n_x$
$f_{low}^{(1:n_x)} \leftarrow$ initialize the case function as per (9)
$n_1 \leftarrow 1$
for $i = 1$ **to** n_x **do**
  $f_{low}^{(i+1:n_x)} \leftarrow \infty$
  for $j = 1$ **to** n_i **do**
    $f_j^{(i+1:n_x)} \leftarrow \min_{\bar{x}_i} f_{low,j}^{(i:n_x)}$ (12b)
    $f_{low}^{(i+1:n_x)} \leftarrow \text{casemin}(f_{low}^{(i+1:n_x)}, f_j^{(i+1:n_x)})$ (12a)
  end for
  $x'_i \leftarrow \arg f_{low}^{(i+1:n_x)}$ as per (11)
  $n_{i+1} \leftarrow$ # of partitions in $f_{low}^{(i+1:n_x)}$
end for
$x^*_{n_x} \leftarrow x'_{n_x}$
for $i = 1$ **to** $n_x - 1$ **do**
  $x^*_{n_x-i} \leftarrow x'_{n_x-i}$
  for $j = 0$ **to** $i - 1$ **do**
    $x^*_{n_x-i} \leftarrow x^*_{n_x-i}\sigma_j$ with $\sigma_j = \{\bar{x}_{n_x-j}/x^*_{n_x-j}\}$
  end for
end for

B. Symbolic Argmin Computation via Compact Decision Diagram Representation of Case Functions

As stated in Section 5.2, maintaining case representations with explicit partitions can be prohibitively expensive in practice. Hence, we use a more compact representation known as Extended Algebraic Decision Diagrams (XADD) (Sanner et al., 2011).

An XADD is similar to an algebraic decision diagram (ADD) (Bahar et al., 1993), except that (a) decision nodes can have arbitrary inequalities (one per node) and (b) leaf nodes can represent arbitrary functions (Sanner & Abbasnejad, 2012). Every leaf of an XADD corresponds to a function value of a case function, and a path from the root to this leaf is uniquely associated with one partition. Note that there can be multiple paths from the root to a leaf when different partitions share the same function value. Notably, an XADD is a directed acyclic graph (DAG), which can be exponentially more compact than an equivalent case or tree data structure (Sanner & Abbasnejad, 2012). When it comes to representing an LPWL function with an XADD, it suffices to restrict decision nodes to have linear inequalities and leaf values to linear expressions. For evaluation, we ported the original XADD implementation in Java[6] to our own implementation in Python.

Given this representation, Algorithm 2 summarizes our symbolic $\arg\min$ workflow. We first define a case function that corresponds to a given MILP. Then, we iterate through each and every decision variable for variable elimination (i.e., symbolic minimization). This can be done by eliminating a variable from each partition and taking the casemin for these results. We annotate the solution during this process and retrieve the solution once each variable is eliminated. Once all variables are eliminated, we are left with $x^*_{n_x}$ which is a case function of $\bar{\mathbf{c}}$. We start the backward substitution step to restrict the values of x_{i+1} variables occurring in the expressions of x'_i.

C. Details of Empirical Evaluation

Here, we describe the domains we have used in the evaluation in more detail. Then, we discuss the hyperparameter tuning for the baselines, followed by an additional set of experiments varying the noise level in data generating processes.

[6] https://github.com/ssanner/xadd-inference

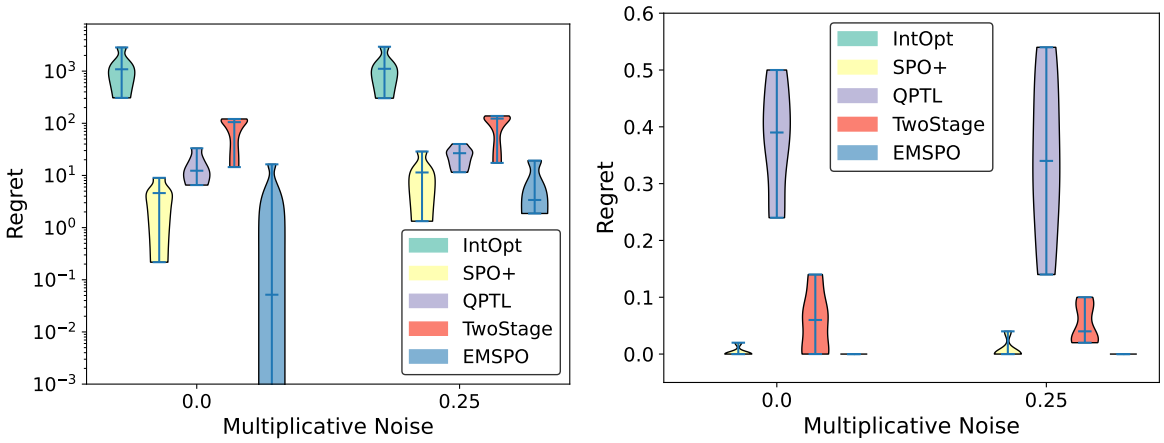

Figure 4. The decision regret subject to different levels of noise (Shortest path - Classification). Notice the log scale on the left figure.

Noisy shortest path Similar to Elmachtoub & Grigas (2017), we consider a shortest path problem on a 3×3 grid network consisting of 12 edges directing towards either north or east. The travel times of the edges (edge costs) are unknown, and they are estimated using p-dimensional features with $p = 5$. The cost vector of the shortest path problem has the dimensionality of $n_x = 12$, since each edge of the network has an edge cost.

We first generate the parameters of the true model using a random matrix $\Theta^* \in \mathbb{R}^{n_x \times p}$, sampled from the Bernoulli distribution with probability 0.5. Next for the ith sample, we generate the feature vector $\varphi_i \in \mathbb{R}^p$ using a multivariate Gaussian distribution with i.i.d. standard normal entries, i.e., $\varphi_i \sim N(0, I_p)$. With these features, we then generate the associated cost vector c_i according to $\left[\left(\frac{1}{\sqrt{p}}(\Theta^*\varphi_i)_j + 3\right)^{\text{deg}} + 1\right] \cdot \epsilon_i^j$ for $j = 1, \ldots, n_x$, where c_{ij} denotes the jth component of c_i and $(\Theta^*\varphi_i)_j$ is the jth component of $\Theta^*\varphi_i$. Here, deg is a fixed positive integer parameter that controls the level of nonlinearity in the generated dataset. ϵ_i^j is a multiplicative noise term which we sample independently at random from the uniform distribution on $[1 - \bar{\epsilon}, 1 + \bar{\epsilon}]$ for some parameter $\bar{\epsilon} \geq 0$.

Energy-cost aware scheduling The energy cost-aware scheduling problem was proposed in Simonis et al., where the energy price data were obtained from the Irish Single Electricity Market Operator (SEMO) (Ifrim et al., 2012). The price dataset consists of historical energy price data at 30-minute intervals. To reduce the problem size in our evaluation, we instead used 2-hour intervals. This brings down the discretized timeslots within the day from 48 to 12. The price dataset is available for 789 days, while we used a fifth of it for each instance of training. See Simonis et al. for more detailed description.

Cost-sensitive multi-class classification The synthetic dataset for the cost-sensitive multi-class classification problem was generated following Liu & Grigas (2021). In this experiment, the number of classes n_x is 5 and the feature dimension p is 5. First, we generate a weight vector $b \in \mathbb{R}^p$, whose jth entry is sampled from the Bernoulli distribution with the probability $\mathbb{P}(b_j = 1) = \frac{1}{2}$. Then, we proceed to generate the training dataset $\{(\varphi^{(n)}, \mathbf{c}^{(n)})\}_{n=1}^N$ by sequentially going through the following steps: (a) From the standard multivariate normal distribution, we generate a feature vector $\varphi^{(n)} \in \mathbb{R}^p$. (b) Using the feature vector we generate the score $s^{(n)} = \sigma((b^T\varphi^{(n)})^{\text{deg}} \cdot \text{sign}(b^T\varphi^{(n)}) \cdot \epsilon)$, where $\sigma(\cdot)$ is the logistic function and therefore $s^{(n)} \in (0, 1)$. Here, the multiplicative noise term ϵ is sampled from a uniform distribution $[1 - \bar{\epsilon}, 1 + \bar{\epsilon}]$ with $\bar{\epsilon} \geq 0$. (c) Ultimately, we generate the true class label $\text{lab}^{(n)} = \lceil 5s^{(n)} \rceil \in \{1, \ldots, 5\}$, which in turn is utilized to generate the true cost vector $\mathbf{c}^{(n)} = (c_1^{(n)}, \ldots, c_5^{(n)})$ using $c_j^{(n)} = |j - \text{lab}^{(n)}|$ for $j = 1, \ldots, 5$.

C.1. Hyperparameter Selection of Baselines

We experimented with 4 baselines in the main paper: TwoStage, SPO+, QPTL, and IntOpt. Each method was tuned for its hyperparameters for the decision regret value in the train set. We ran each method for 300 epochs and recorded the best result obtained for each experiment configuration. For details about the hyperparameters other than the learning rate, we refer readers to the original papers (Mandi & Guns, 2020; Wilder et al., 2019).

Problem	Value	TwoStage	SPO+	IntOpt	QPTL
Nonlinearity	1	lr (0.1)	lr (0.001)	lr (0.001); d (10^{-6}); λ (0.1)	lr (0.01); τ (100)
	2	lr (0.01)	lr (0.01)	lr (0.001); d (10^{-6}); λ (0.1)	lr (0.001); τ (10)
	5	lr (0.1)	lr (0.1)	lr (0.001); d (10^{-6}); λ (0.1)	lr (0.01) τ (100)
	10	lr (0.01)	lr (0.01)	lr (0.001); d (10^{-6}); λ (0.001)	lr (0.001); τ (100)
Data size	50	lr (0.01)	lr (0.1)	lr (0.1); d (10^{-6}); λ (0.001)	lr (0.001); τ (10)
	100	lr (0.01)	lr (0.01)	lr (0.001); d (10^{-6}); λ (0.1)	lr (0.001); τ (10)
	200	lr (0.01)	lr (0.01)	lr (0.1); d (10^{-6}); λ (0.001)	lr (0.1); τ (10)
Noise	0.0	lr (0.01)	lr (0.1)	lr (0.1); d (0.001); λ (0.01)	lr (0.001) τ (10)
	0.25	lr (0.01)	lr (0.1)	lr (0.1); d (10^{-6}); λ (0.001)	lr (0.001); τ (10)

Table 1. Hyperparameters selected for baseline algorithms for the classification problem.

Problem	Value	TwoStage	SPO+	IntOpt	QPTL
Nonlinearity	1	lr (0.01)	lr (0.001)	lr (0.01); d (0.001); λ (0.1)	lr (0.01); τ (10)
	2	lr (0.1)	lr (0.01)	lr (0.001); d (10^{-6}); λ (0.1)	lr (0.01); τ (10)
	5	lr (0.1)	lr (0.1)	lr (0.01); d (10^{-6}); λ (0.1)	lr (0.01) τ (10)
	10	lr (0.1)	lr (0.1)	lr (0.01); d (10^{-6}); λ (0.01)	lr (0.001); τ (10)
Data size	50	lr (0.1)	lr (0.1)	lr (0.01); d (10^{-6}); λ (0.01)	lr (0.01); τ (10)
	100	lr (0.1)	lr (0.1)	lr (0.001); d (0.0001); λ (0.1)	lr (0.001); τ (100)
	200	lr (0.1)	lr (0.1)	lr (0.001); d (0.01); λ (0.001)	lr (0.001); τ (10)
Noise	0.0	lr (0.1)	lr (0.1)	lr (0.001); d (0.0001); λ (0.1)	lr (0.001) τ (100)
	0.25	lr (0.1)	lr (0.1)	lr (0.001); d (0.0001); λ (0.1)	lr (0.001); τ (100)

Table 2. Hyperparameters selected for baseline algorithms for the shortest path problem.

Problem	Value	TwoStage	SPO+	IntOpt	QPTL
Size (Days)	20	lr (0.001)	lr (0.01)	lr (0.001); d (0.001); λ (0.001)	lr (0.01); τ (100000)
	50	lr (0.001)	lr (0.01)	lr (0.1); d (0.001); λ (0.001)	lr (0.1); τ (10000)
	100	lr (0.001)	lr (0.1)	lr (0.1); d (0.001); λ (0.001)	lr (0.1) τ (100000)
Data Type	Linear	lr (0.001)	lr (0.001)	lr (0.1); d (0.001); λ (0.001)	lr (0.1); τ (10000)
	Nonlinear	lr (0.001)	lr (0.001)	lr (0.1); d (0.001); λ (0.001)	lr (0.1); τ (10000)

Table 3. Hyperparameters selected for baseline algorithms for the energy-cost aware scheduling problem.

C.2. Additional Empirical Evaluations

In this part, we present some additional experimental results. Specifically, Figure 4 shows the decision regret and optimality gap subject to two different noise levels in the datasets. For this, we used the shortest path and the cost-sensitive classification datasets, which allows us to control the noise level by varying the noise parameter $\bar{\epsilon}$. Hyperparameters were selected based on grid-search and were evaluated with the decision regret on the training data after 300 epochs of training. The best-recorded hyperparameters are shown in Table 1.