A Sampling-Based Relaxation Approach to Contextual Inverse Optimization

Yasunari Hikima[1], **Naoyuki Kamiyama**[2], **Shinsaku Sakaue**[3] and **Taira Tsuchiya**[4]
[1]Fujitsu Limited, Kyushu University
[2]Kyoto University
[3]CyberAgent, National Institute of Informatics, RIKEN Center for Advanced Intelligence Project
[4]The University of Tokyo, RIKEN Center for Advanced Intelligence Project
hikima.yasunari@fujitsu.com, kamiyama@amp.i.kyoto-u.ac.jp,
shinsaku.sakaue@gmail.com, tsuchiya@mist.i.u-tokyo.ac.jp

Abstract

Decision-making pipelines increasingly rely on prediction models whose outputs serve as inputs to downstream optimization problems. Decision-Focused Learning (DFL) has emerged as a promising approach to training such models by directly optimizing decision quality rather than predictive accuracy alone. While most existing DFL methods assume a complete-information setting in which the ground-truth optimization parameters are observed, this paper studies Contextual Inverse Optimization (CIO), an incomplete-information setting in which only the resulting solutions are observed.

Prior work on CIO has proposed learning algorithms based on optimality conditions for linear programs, as well as methods that repeatedly solve inverse optimization problems to handle integer programs, often incurring a substantial computational burden. In this paper, we propose a learning algorithm for general optimization problems with linear objective functions that eliminates the need to solve inverse optimization problems. The proposed method learns prediction models by solving a Relaxed Inverse Optimization Problem (RIOP), constructed based on feasible solutions randomly sampled from the feasible region, thereby reducing the computational overhead associated with existing CIO methods. Numerical experiments demonstrate that our method achieves competitive performance in terms of *regret* compared with existing methods, while offering improved computational efficiency for certain classes of downstream optimization problems.

1 Introduction

In the modern era of data-driven operations management, decision-making processes increasingly rely on contextual data to estimate uncertain parameters included in optimization models [Mišić and Perakis, 2020; Bertsimas and Kallus, 2020; Elmachtoub and Grigas, 2022]. Consider a classical optimization problem where the feasible region is given, but the parameters of the objective function are not known at the time of making decisions. In the field of operations research, various approaches have been studied to address such decision-making problems under uncertainty, including stochastic programming [Kall and Wallace, 1994; Shapiro *et al.*, 2021; Prékopa, 2013; Birge *et al.*, 2022] and robust optimization [Ben-Tal and Nemirovski, 2002; Beyer and Sendhoff, 2007; Delage and Ye, 2010]. In contrast, in modern decision-making problems where diverse data related to the unknown parameters are observed, the "predict-and-optimize" approach is typically employed, where the unknown parameters are predicted from the data, and the decision-maker solves the optimization problem parameterized by the predicted values.

While the conventional two-stage approach is conceptually straightforward, it often leads to suboptimal decisions in practice [Elmachtoub and Grigas, 2022; Mandi *et al.*, 2024]. The primary reason is that standard machine learning models are typically trained to minimize prediction error, such as mean squared error, without accounting for the downstream decision quality. For instance, a small prediction error in objective coefficients may result in a significant degradation in the quality of the resulting decision, particularly when the optimization problem is sensitive to parameter variations. To bridge this gap, approaches such as "Smart Predict-then-Optimize" (SPO) [Elmachtoub and Grigas, 2022] or decision-focused learning (DFL) [Mandi *et al.*, 2024], have attracted much attention in both the operations research and machine learning communities. The former proposes a convex surrogate loss function, referred to as SPO+, that approximates the regret incurred by using predicted parameters instead of true parameters in the optimization problem. The latter integrates the optimization problem into the learning process by differentiating through the optimization solver, allowing for end-to-end training of the prediction model with respect to decision quality [Amos and Kolter, 2017; Wilder *et al.*, 2019; Kotary *et al.*, 2021; Mandi *et al.*, 2022].

Despite the developments of these learning frameworks, a critical limitation of SPO and most existing DFL methods (see [Mandi *et al.*, 2024, Sect. 3] and [Sadana *et al.*, 2024, Sect. 5] for the learning methodologies) is the assumption that the learner has access to ground-truth objective function parameters. In many real-world scenarios, however, such ground-truth parameters are often unobservable or prohibitively expensive to obtain. As an illustrative example,

consider the problem of inferring edge weights for a shortest path problem using the historical trajectories of drivers on an urban network. In such a situation, while we can observe the paths taken by the drivers, obtaining the actual edge weights (i.e., true travel times) may be costly or impractical.

To address the aforementioned limitation, this paper studies the *contextual inverse optimization* problem [Sun *et al.*, 2023; Mishra *et al.*, 2024; Besbes *et al.*, 2025; Hikima and Kamiyama, 2025], where the ground-truth objective function parameters are unavailable, but we have access to observed decision data. Prior works in this setting have primarily focused on linear programming problems to exploit optimality conditions [Sun *et al.*, 2023; Mishra *et al.*, 2024], or relied on algorithms that involve repeatedly solving inverse optimization problems for combinatorial optimization problems [Hikima and Kamiyama, 2025]. In contrast, this paper proposes a learning algorithm for general combinatorial optimization problems that estimates objective function parameters without explicitly solving inverse optimization problems.

The main contributions are summarized as follows:

- We propose a learning algorithm for contextual inverse optimization that estimates objective function parameters without requiring access to ground-truth parameters. In contrast to existing approaches, our algorithm does not rely on optimality conditions (e.g., KKT conditions) nor does it require solving inverse optimization problems as subroutines. The proposed framework extends applicability beyond linear programs to a broad class of optimization problems with linear objective functions, while offering computational efficiency.
- The proposed algorithm is based on a relaxed formulation of the inverse optimization problem constructed from feasible solutions sampled from the feasible region of the downstream optimization problem. A key advantage of the framework is its ability to directly integrate efficient sampling algorithms tailored to specific combinatorial structures, such as knapsack problems or traveling salesman problems. This enables model learning without the computational overhead of solving the forward or inverse optimization at every iteration.
- We demonstrate the effectiveness of the proposed algorithm through numerical experiments on contextual inverse optimization instances that involve estimating objective function parameters. The results indicate that our algorithm achieves competitive performance in terms of expected *regret* compared with the existing methods, including those designed for the complete information setting. Notably, our algorithm converges in fewer epochs and yields faster model training than existing methods. Moreover, we show that competitive performance is maintained even when employing sampling algorithms that entirely bypass solving the forward optimization problem during training.

2 Problem Formulation

This section provides a formal description of the contextual inverse optimization problem. Consider a general class of integer linear programming problems as the forward optimization problem where the feasible region is defined as a subset of $\{0,1\}^n$, where $n \in \mathbb{N}$ is the dimensionality of the decision variable. The forward optimization problem parameterized by the coefficient vector $\boldsymbol{c} \in \mathbb{R}^n$ is defined as follows:

$$\mathrm{FO}(\boldsymbol{c}): \quad \begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X}, \end{aligned} \tag{1}$$

where $\mathcal{X} \subseteq \{0,1\}^n$ is a nonempty feasible region of the forward optimization problem. Although we focus on binary decision variables for simplicity and to match our experimental settings, the proposed framework can be extended to general feasible regions without significant modifications. Given a realized coefficient vector $\boldsymbol{c} \in \mathbb{R}^n$, we denote the optimal solution to the forward problem as $\boldsymbol{x}^\star(\boldsymbol{c}) \in \arg\max_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{c}^\top \boldsymbol{x}$.

In the contextual inverse optimization problem, we assume that we have access to contextual information $\boldsymbol{z} \in \mathbb{R}^d$ that is correlated with the unknown coefficient vector $\boldsymbol{c} \in \mathbb{R}^n$. Here, $d \in \mathbb{N}$ is the dimensionality of the context vector. In contrast to the SPO [Elmachtoub and Grigas, 2022] or DFL [Mandi *et al.*, 2024; Sadana *et al.*, 2024] settings, we do not assume that the ground-truth objective coefficient parameters are observable during the training phase. When the ground-truth parameters are observable, the problem is said to be in the *complete-information* setting; otherwise, it is referred to as the *incomplete-information* setting [Berden *et al.*, 2025]. Note that the latter is more challenging, as the learner must infer the underlying objective parameters solely from observed decisions and contextual information.

The goal of the contextual inverse optimization problem is to learn a prediction model $h\colon \mathbb{R}^d \to \mathbb{R}^n$ that maps a context vector $\boldsymbol{z} \in \mathbb{R}^d$ to a coefficient vector $\boldsymbol{c} \in \mathbb{R}^n$ such that the optimal solution to the forward optimization problem $\mathrm{FO}(h(\boldsymbol{z}))$ closely approximates the optimal solution to $\mathrm{FO}(\boldsymbol{c})$ for the underlying true coefficient parameter. In other words, given a dataset consisting of context–solution pairs $\mathcal{D} = \{(\boldsymbol{z}_i, \boldsymbol{x}_i^\star)\}_{i=1}^N$, we aim to learn a prediction model h such that the optimal solution to the forward problem $\mathrm{FO}(h(\boldsymbol{z}))$ attains a near-optimal objective value with respect to the true coefficient $\boldsymbol{c}$. Throughout this paper, we say that an observed solution $\boldsymbol{x}$ is *rationalized* by an estimated coefficient vector $\hat{\boldsymbol{c}}\,(\neq \boldsymbol{0})$ if $\langle \hat{\boldsymbol{c}}, \boldsymbol{x}\rangle \approx \langle \hat{\boldsymbol{c}}, \boldsymbol{x}^\star(\hat{\boldsymbol{c}})\rangle$ holds. This means that the observed solution $\boldsymbol{x}$ is nearly optimal for the forward optimization problem parameterized by $\hat{\boldsymbol{c}}$.

Let $\mathcal{H}$ be a hypothesis class consisting of prediction models that map context vectors to coefficient vectors. Then, the performance of a hypothesis $h \in \mathcal{H}$ is evaluated in terms of the following expected *regret* on the distribution of test data ρ,

$$\ell_{\text{regret}}(h; \boldsymbol{z}, \boldsymbol{c}) \coloneqq \boldsymbol{c}^\top \boldsymbol{x}^\star(\boldsymbol{c}) - \boldsymbol{c}^\top \boldsymbol{x}^\star(h(\boldsymbol{z})),$$
$$R_{\text{regret}}(h) \coloneqq \mathbb{E}_{(\boldsymbol{z},\boldsymbol{c})\sim\rho}\left[\ell_{\text{regret}}(h; \boldsymbol{z}, \boldsymbol{c})\right],$$

which measures the expected difference in objective value between the optimal solution under the true cost vector $\boldsymbol{c}$ and the optimal solution under the predicted coefficient vector $h(\boldsymbol{z})$.

3 Proposed Algorithm

In this section, we present an algorithm for learning a prediction model to estimate the coefficient parameters of the

objective function in the incomplete-information setting. To estimate the optimization parameter from a given optimal solution, optimality conditions or inverse optimization have been typically employed in the literature [Sun *et al.*, 2023; Mishra *et al.*, 2024; Hikima and Kamiyama, 2025]. In contrast, the proposed algorithm solves a Relaxed Inverse Optimization Problem (RIOP) formulated based on feasible solutions randomly sampled from the feasible region. In the first subsection (Sect. 3.1), we outline the framework of our proposed algorithm based on solving the forward optimization to obtain feasible solutions. The subsequent subsection (Sect. 3.2) introduces cases where sampling feasible solutions can be performed efficiently for specific optimization problems.

3.1 Algorithmic Framework

First, we present an algorithmic framework for learning a prediction model that maps contexts to coefficient vectors of the objective function in the incomplete-information setting. Let $T \in \mathbb{N}$ denote a maximum number of iterations, and $\boldsymbol{\Theta}^{(0)}$ be an initial model parameter of the prediction model $h(\cdot;\boldsymbol{\Theta})$ to be learned. We are given a training dataset $\mathcal{D} = \{(\boldsymbol{z}_i, \boldsymbol{x}_i^\star)\}_{i=1}^N$ consisting of context–solution pairs drawn i.i.d. from an unknown distribution ρ. Here, each $\boldsymbol{x}_i^\star$ is assumed to be an optimal solution to the forward optimization problem $\mathrm{FO}(\boldsymbol{c}_i)$ for some unknown coefficient vector $\boldsymbol{c}_i$ that is correlated with the context $\boldsymbol{z}_i$.

At each iteration $0 \le t < T$, we first predict the coefficient vector for each data point as $\hat{\boldsymbol{c}}_i = h(\boldsymbol{z}_i; \boldsymbol{\Theta}^{(t)})$ using the current prediction model with parameter $\boldsymbol{\Theta}^{(t)}$. Given this prediction, we then construct a candidate coefficient vector under which the observed solution $\boldsymbol{x}_i^\star$ is encouraged to be nearly optimal. More precisely, we aim to find a coefficient vector $\boldsymbol{c}$ such that $\boldsymbol{x}_i^\star$ can be rationalized by $\boldsymbol{c}$, namely $\langle \boldsymbol{c}, \boldsymbol{x}_i^\star \rangle \approx \langle \boldsymbol{c}, \boldsymbol{x}^\star(\boldsymbol{c}) \rangle$. Instead of directly imposing the rationalization condition, we formulate a relaxed inverse optimization problem that promotes the observed solution $\boldsymbol{x}_i^\star$ to have a higher objective value than other feasible solutions sampled from the feasible region $\mathcal{X}$. The resulting coefficient vectors are used as regression targets to update the model parameter $\boldsymbol{\Theta}$, and the procedure is repeated until the number of iterations reaches T.

To construct the finite set of feasible solutions used in the formulation of the relaxed inverse optimization problem, we first perturb the predicted coefficient vector $\hat{\boldsymbol{c}}_i$ with random noise drawn from the multivariate normal distribution, that is, $\widetilde{\boldsymbol{c}}_{i,k} = \hat{\boldsymbol{c}}_i + \boldsymbol{\eta}_{i,k}$ with $\boldsymbol{\eta}_{i,k} \sim \mathcal{N}(\mathbf{0}_n, \sigma \mathbf{I}_n)$, where $\sigma > 0$ is a constant noise level, $\mathbf{0}_n$ and $\mathbf{I}_n$ denote the zero vector and the identity matrix of size n. We then sample $K \in \mathbb{N}$ feasible solutions $\{\widetilde{\boldsymbol{x}}_{i,k}\}_{k=1}^K$ by solving the forward optimization problem $\mathrm{FO}(\widetilde{\boldsymbol{c}}_{i,k})$. Namely,

$$\widetilde{\boldsymbol{x}}_{i,k} \in \arg\max_{\boldsymbol{x}} \left\{ (\widetilde{\boldsymbol{c}}_{i,k})^\top \boldsymbol{x} \;\middle|\; \boldsymbol{x} \in \mathcal{X} \right\} \text{ for } k \in [K],$$

where $[K]$ denotes the finite set $\{1, 2, \ldots, K\}$.

With the sampled feasible solutions, we can formulate a relaxed inverse optimization problem whose optimal solution corresponds to a candidate for the coefficient vector, denoted as $\boldsymbol{c}_i^{(t)}$, for rationalizing the observed optimal solution $\boldsymbol{x}_i^\star$. For a given vector $\boldsymbol{r} \in \mathbb{R}^n$, an observed solution $\boldsymbol{x}_i^\star \in \mathcal{X}$, and sampled feasible solutions $\{\widetilde{\boldsymbol{x}}_{i,k}\}_{k=1}^K$, we define the relaxed inverse optimization problem (RIOP) as follows:

$$\mathrm{RIOP}(\boldsymbol{r}, \{\widetilde{\boldsymbol{x}}_{i,k}\}_{k=1}^K, \mathrm{dist}(\cdot,\cdot)):$$

$$\underset{\boldsymbol{c}\in\mathbb{R}^n,\, \boldsymbol{\xi}\in\mathbb{R}^K}{\text{minimize}} \quad \mathrm{dist}(\boldsymbol{r}, \boldsymbol{c}) - \frac{1}{K}\sum_{k=1}^K \xi_k \tag{2a}$$

$$\text{subject to} \quad \langle \boldsymbol{c}, \boldsymbol{x}_i^\star - \widetilde{\boldsymbol{x}}_{i,k} \rangle \ge \xi_k \;\; (\forall k \in [K]), \tag{2b}$$

$$\xi_k \ge 0 \;\; (\forall k \in [K]). \tag{2c}$$

Here, $\mathrm{dist}\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\ge 0}$ is a proximity measure on $\mathbb{R}^n$, $\boldsymbol{r} \in \mathbb{R}^n$ is a so-called reference point, and we set it as an output of a current prediction model, i.e., $\boldsymbol{r} = \hat{\boldsymbol{c}}_i = h(\boldsymbol{z}_i; \boldsymbol{\Theta}^{(t)})$.

The formulation of RIOP can be interpreted as finding a coefficient vector $\boldsymbol{c}$ that balances two objectives: proximity to a reference point $\boldsymbol{r} = \hat{\boldsymbol{c}}_i$, and consistency with the observed optimal solution $\boldsymbol{x}_i^\star$ in terms of objective value. The latter is enforced by the constraints (2b), which require that the observed optimal solution $\boldsymbol{x}_i^\star$ has an objective value at least ξ_k greater than that of each sampled feasible solution $\{\widetilde{\boldsymbol{x}}_{i,k}\}_{k=1}^K$ under the cost vector $\boldsymbol{c}$. The margin variable ξ_k quantifies the extent to which $\boldsymbol{x}_i^\star$ is preferred over $\widetilde{\boldsymbol{x}}_{i,k}$ under $\boldsymbol{c}$. Consequently, the optimal solution $\boldsymbol{c}_i^{(t)}$ to RIOP provides a candidate for the coefficient vector under which $\boldsymbol{x}_i^\star$ is nearly optimal among the sampled feasible solutions. In this sense,

Algorithm 1: Algorithm for learning a model for contextual inverse optimization problem (Full batch case)

Init: Initial model parameter $\boldsymbol{\Theta}^{(0)}$ of hypothesis h, number of samples $K \in \mathbb{N}$ to draw from the feasible region $\mathcal{X} \subset \mathbb{R}^n$ in each iteration, maximum number of iterations $T > 0$, noise level parameter $\sigma > 0$, regularization parameter $\lambda \ge 0$, and distance function $\mathrm{dist}(\cdot,\cdot)$ on $\mathbb{R}^n$
Input : Context–solution pairs $\{(\boldsymbol{z}_i, \boldsymbol{x}_i^\star)\}_{i=1}^N$
Output: Model parameter $\boldsymbol{\Theta}^{(T)}$

Set $\boldsymbol{\Theta}^{(t)} \leftarrow \boldsymbol{\Theta}^{(0)}$ and $t \leftarrow 0$
while $t < T$ **do**
  Predict $\hat{\boldsymbol{c}}_i \leftarrow h(\boldsymbol{z}_i; \boldsymbol{\Theta}^{(t)})$ for each $i \in [N]$
  for $k = 1$ **to** K **do**
    Draw $\boldsymbol{\eta}_{i,k}^{(t)} \sim \mathcal{N}(\mathbf{0}_n, \sigma \mathbf{I}_n)$ for each $i \in [N]$
    Perturb $\widetilde{\boldsymbol{c}}_{i,k} \leftarrow \hat{\boldsymbol{c}}_i + \boldsymbol{\eta}_{i,k}^{(t)}$ for each $i \in [N]$
    Solve $\mathrm{FO}(\widetilde{\boldsymbol{c}}_{i,k})$ and obtain $\widetilde{\boldsymbol{x}}_{i,k} \in \mathcal{X}$ for each $i \in [N]$
  end
  Solve $\mathrm{RIOP}(\hat{\boldsymbol{c}}_i, \{\widetilde{\boldsymbol{x}}_{i,k}\}_{k=1}^K, \mathrm{dist}(\cdot,\cdot))$ in Eq. (2) and obtain $\boldsymbol{c}_i^{(t)}$ for each $i \in [N]$
  Set $\boldsymbol{\Theta}^{(t+1)}$ by solving Eq. (3)
  Set $t \leftarrow t + 1$
end
return $\boldsymbol{\Theta}^{(T)}$

RIOP can be viewed as a relaxation of the inverse optimization and is used to obtain a coefficient vector that approximately rationalizes the observed solution $\boldsymbol{x}_i^\star$.

Note that the choice of $\mathrm{dist}(\cdot,\cdot)$ determines the class of the optimization problem in Eq. (2). When $\mathrm{dist}(\cdot,\cdot)$ is the ℓ_1-norm on $\mathbb{R}^n$, i.e., $\mathrm{dist}(\boldsymbol{r},\boldsymbol{c}) = \|\boldsymbol{r}-\boldsymbol{c}\|_1$, RIOP can be reduced to a linear program by introducing auxiliary variables for the absolute values, which is computationally efficient to solve even if K is relatively large. Moreover, although the squared ℓ_2-norm is not a distance metric in the strict sense, it can be used here as a proximity measure, and RIOP becomes a convex quadratic program with linear constraints, which can also be solved efficiently by standard optimization solvers.

After obtaining the candidate estimate of the coefficient parameter $\boldsymbol{c}_i^{(t)}$ for each data point, we update the model parameter $\boldsymbol{\Theta}$ by the following regularized empirical risk minimization problem:

$$\boldsymbol{\Theta}^{(t+1)} \in \arg\min_{\boldsymbol{\Theta}} \left\{ \sum_{i=1}^{N} \left\| h(\boldsymbol{z}_i;\boldsymbol{\Theta}) - \boldsymbol{c}_i^{(t)} \right\|_2^2 + \lambda \|\boldsymbol{\Theta}\|_{\mathsf{F}}^2 \right\}, \quad (3)$$

where $\lambda \geq 0$ is a regularization parameter for preventing overfitting and $\|\cdot\|_{\mathsf{F}}$ denotes the Frobenius norm. For a linear model, $h(\boldsymbol{z};\boldsymbol{\Theta}) = \boldsymbol{\Theta}\boldsymbol{z}$, the above update rule can be expressed in closed-form as $\boldsymbol{\Theta}^{(t+1)} = \left(\sum_i \boldsymbol{c}_i^{(t)} \boldsymbol{z}_i^\top\right)\left(\sum_i \boldsymbol{z}_i \boldsymbol{z}_i^\top + \lambda \mathbf{I}\right)^{-1}$, or equivalently $\boldsymbol{\Theta}^{(t+1)} = \left(\mathbf{C}^{(t)}\mathbf{Z}^\top\right)\left(\mathbf{Z}\mathbf{Z}^\top + \lambda\mathbf{I}\right)^{-1}$, where $\mathbf{C}^{(t)} = [\boldsymbol{c}_1^{(t)}, \boldsymbol{c}_2^{(t)}, \ldots, \boldsymbol{c}_N^{(t)}]$ and $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N]$. The overall procedure for the full batch case is summarized in Algorithm 1.

3.2 Making the Proposed Algorithm Efficient

In the proposed algorithm (Algorithm 1), it is necessary to repeatedly solve a forward optimization problem to obtain feasible solutions at every iteration (lines 4–8 in Algorithm 1), which can be computationally expensive. Fortunately, there exists a substantial body of research on methods for sampling or enumerating feasible solutions from the feasible region, exploiting the specific structure of the combinatorial optimization problem. In cases where efficient sampling algorithms can be applied to the forward optimization problem, lines 4–8 of Algorithm 1 can be replaced by the sampling algorithm. This modification significantly reduces the computational overhead by eliminating the need to repeatedly solve the forward optimization problem to generate a set of feasible solutions at each iteration.

Below, to illustrate the flexibility of our framework, we introduce sampling methods based on a dynamic programming approach for the $\{0,1\}$-knapsack problem [Bossek *et al.*, 2021], a randomized algorithm for the max-cut problem, and briefly discuss how to generate feasible solutions for the traveling salesman problem.

Knapsack Problem. Consider the $\{0,1\}$-knapsack problem with $n \in \mathbb{N}$ items. Each item $i \in \{1,2,\ldots,n\}$ has its profit $p_i \in \mathbb{N}$ and weight $w_i \in \mathbb{N}$, and the maximum capacity is given as $W \in \mathbb{N}$. The objective of the $\{0,1\}$-knapsack problem is to find the subset of items $S \subseteq \{1,2,\ldots,n\}$ that maximizes the total profit $\sum_{i\in S} p_i$ without exceeding the maximum capacity of the knapsack.

Algorithm 2: Sampling algorithm for the $\{0,1\}$-Knapsack Problem (cf., [Bossek *et al.*, 2021])

Input : Weights $\boldsymbol{w} \in \mathbb{N}^n$ and capacity $W \in \mathbb{N}$
Output: Feasible solution $\boldsymbol{x} \in \{0,1\}^n$
1 Calculate $\{F(j,k)\}_{j,k}$ according to Eq. (4)
2 Set $\boldsymbol{x} \leftarrow \mathbf{0}$, $j \leftarrow n$, and $k \leftarrow W$
3 **while** $j > 0$ **do**
4 With probability $F(j-1,k-w_j)/F(j,k)$, set $x_j \leftarrow 1$ and $k \leftarrow k - w_j$
5 Set $j \leftarrow j-1$
6 **end**
7 **return** $\boldsymbol{x}$

Let $F(j,k)$ be the number of sets $T \subseteq [j]$ satisfying $\sum_{i\in T} w_i \leq k$. The values $F(j,k)$ for $j = 0,1,\ldots,n$ and $k = 0,1,\ldots,W$ can be computed by the dynamic programming (DP) recursion process expressed as follows:

$$F(j,k) = F(j-1,k) + F(j-1,k-w_j) \quad \text{if } j \geq 1,$$
$$F(0,k) = \begin{cases} 1 & \text{if } k \geq 0, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

The DP table $\{F(j,k)\}_{j,k}$ can be constructed in $O(nW)$ time. Using this table, we can sample a feasible solution uniformly at random by backtracking the recursion from state (n,W), including item j at state (j,k) with probability $F(j-1,k-w_j)/F(j,k)$, and then moving to the subproblem with the first $j-1$ items, updating the remaining capacity if item j is selected. This procedure runs in $O(n)$ time, as summarized in Algorithm 2.

Max-Cut Problem. Consider the Max-Cut problem defined on an undirected graph. Given an undirected graph $G = (V,E)$ with vertex set V and edge set E where each edge $e \in E$ has its weight $w(e) \in \mathbb{R}$, the Max-Cut problem asks to find a set $S \subseteq V$ such that the total weight of edges crossing S and $V \setminus S$ is maximized.

To sample a feasible solution for the Max-Cut problem, one can employ simulated annealing-based algorithms [Haribara *et al.*, 2016]. As another approach, approximation algorithms based on semidefinite programming relaxation can be used, such as the Goemans–Williamson algorithm [Goemans and Williamson, 1995]. Alternatively, we can generate a feasible solution by randomly assigning each vertex to one of the two sides with equal probability, which runs in $O(|V|)$ time.

Traveling Salesman Problem. Consider the traveling salesman problem (TSP) defined on an undirected graph. Given an undirected graph $G = (V,E)$ with vertex set V and edge set E where each edge $e \in E$ has its weight $w(e) \in \mathbb{R}$, the (symmetric) TSP is a classic combinatorial optimization where the goal is to find the shortest possible route that visits all cities and returns to the origin city.

To generate a feasible solution for the TSP, we can use heuristic algorithms such as Lin–Kernighan neighborhood search [Lin and Kernighan, 1973] or the Christofides–Serdyukov algorithm [Christofides, 2022]. In addition, we can employ metaheuristic approaches like genetic algorithms [Larranaga *et al.*, 1999; Chatterjee *et al.*, 1996] or

simulated annealing [Malek *et al.*, 1989; Zhan *et al.*, 2016] to sample feasible solutions from the solution space. Alternatively, if G is a complete graph, we can generate a feasible solution by randomly permuting the list of cities, which runs in $O(|V|)$ times using the Fisher–Yates shuffle algorithm [Fisher and Yates, 1953]. To obtain a higher quality solution, it is also possible to apply the nearest neighbor algorithm [Rosenkrantz *et al.*, 1977], which sequentially visits the nearest unvisited city from the current city.

4 Numerical Experiments

In this section, we present numerical experiments to evaluate the performance of the proposed algorithm for learning contextual inverse optimization problems[1].

4.1 Problem Statements

We apply the proposed algorithm to three integer programming problems: a $\{0,1\}$-knapsack problem (**Knapsack**), a diverse bipartite matching problem (**Matching**), and a traveling salesman problem (**TSP**). For $\{0,1\}$-knapsack problem, refer to Sect. 3.2 for the problem statement.

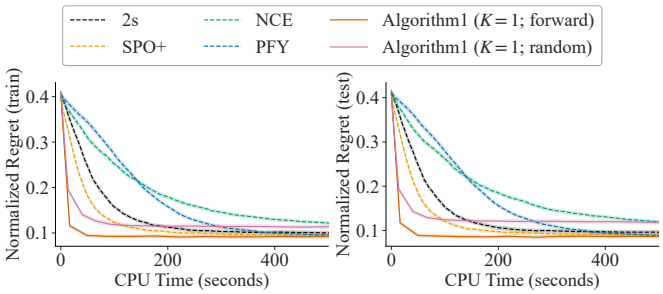
As stated in Sect. 3.2, the **TSP** is to find the shortest possible route that visits each city exactly once and returns to the origin city. Given $n \in \mathbb{N}$ cities, denoted as $V \coloneqq \{1, \dots, n\}$, where each city $i \in V$ has its coordinates $\boldsymbol{s}_i \in \mathbb{R}^2$. The distance matrix $\boldsymbol{c} = (c_{ij})_{ij} \in \mathbb{R}^{n \times n}$ is defined as $c_{ij} = \|\boldsymbol{s}_i - \boldsymbol{s}_j\|_2$ for $i, j \in V$. The (symmetric) TSP is known to be NP-complete [Schrijver, 2003, Theorem 58.1], and it can be formulated as the integer linear program [Dantzig *et al.*, 1954; Applegate *et al.*, 2011] with an exponential number of subtour elimination constraints.

The diverse bipartite matching problem [Ferber *et al.*, 2020] aims to find a matching between two disjoint sets of nodes that maximizes the total weight of the selected edges while satisfying diversity conditions. In this setting, each node is associated with a group attribute (e.g., a field of study), and diversity is enforced by ensuring that the matching contains a certain proportion of pairs connecting nodes from distinct groups as well as a certain proportion of pairs connecting nodes from the same group. Specifically, given constants $p, q \in [0,1]$, at least a fraction p of the selected pairs should connect nodes from the same group, and at least a fraction q of the selected pairs should connect nodes from different groups. With a reward matrix denoted by $\boldsymbol{c} = (c_{ij})_{ij}$ and a feasible region $\mathcal{X}(\boldsymbol{m}, p, q)$ encoding these diversity conditions, the problem can be formulated as the following integer linear program:

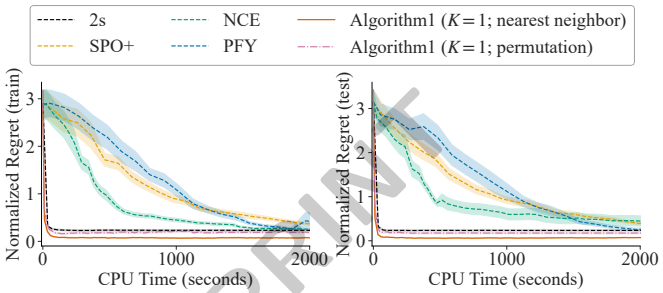
$$\max_{\boldsymbol{x} \in \{0,1\}^{N_1 \times N_2}} \sum\nolimits_{i=1}^{N_1} \sum\nolimits_{j=1}^{N_2} c_{ij} x_{ij} \quad \text{s.t.} \quad \boldsymbol{x} \in \mathcal{X}(\boldsymbol{m}, p, q),$$

where $\boldsymbol{m} = \{m_{ij}\}_{ij} \in \{0,1\}^{N_1 \times N_2}$ indicate whether the two endpoints of edge (i,j) belong to a specific group or not. The specific definition of the feasible set $\mathcal{X}(\boldsymbol{m}, p, q)$ can be found in [Mandi *et al.*, 2024, Sect. 5.1.6].

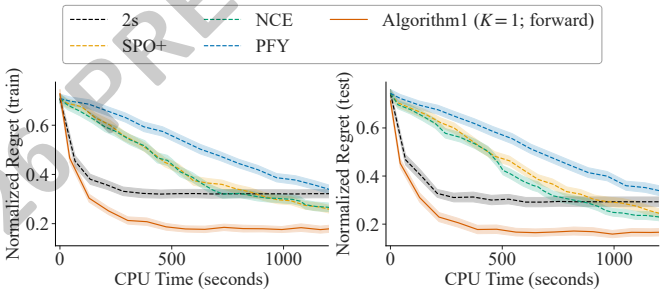
[1]The source code of the proposed algorithm is available at https://github.com/hikimay/sbr-cio.



(a) **Knapsack** (deg $= 4$)



(b) **TSP** (deg $= 6$)



(c) **Matching** (deg $= 8$)

Figure 1: The learning curve of three contextual inverse optimization instances. The vertical axis represents the expected normalized regret on the training dataset (left) and the test dataset (right). The horizontal axis represents the execution time in seconds.

4.2 Synthetic Data Generation

The context vectors $\boldsymbol{z} \in \mathbb{R}^d$ are generated from the normal distribution $\mathcal{N}(\boldsymbol{0}_d, \mathbf{I}_d)$, and we set $d = 5$ for all the experiments. For **Knapsack** instance, each entry of the objective coefficient vectors $(c_i)_i$ is generated according to a polynomial mapping from the context vectors as $c_i = \frac{5}{3.5^{\deg}} \left[\left([\mathbf{B}^* \boldsymbol{z}]_i + 3 \right)^{\deg} + 1 \right] \cdot \varepsilon_i$, where $\deg > 0$ is a predetermined polynomial degree, $\mathbf{B}^* \in \mathbb{R}^{n \times d}$ is a latent true coefficient matrix whose each entry is generated from the Bernoulli distribution with mean 0.5, and ε_i is a multiplicative noise term generated from the uniform distribution on $[1-\bar{\varepsilon}, 1+\bar{\varepsilon}]$ for some $\bar{\varepsilon} > 0$. We denote the polynomial model as $\mathrm{poly}(\boldsymbol{z}; \mathbf{B}^*, \deg, \bar{\varepsilon})$. The weights of items, $\boldsymbol{w} = (w_i)_i$, are sampled from $\{3, 4, \dots, 8\}$ uniformly at random. Note that the data generation process based on the polynomial model is first employed in [Elmachtoub and Grigas, 2022] for the shortest path problem, and it has been used for other instances

Instance	2s		SPO+		NCE		PFY		Algorithm1($K=1$)	
(deg)	N-Reg	CPU[s/ep]	N-Reg	CPU[s/ep]	N-Reg	CPU[s/ep]	N-Reg	CPU[s/ep]	N-Reg	CPU[s/ep]
Knapsack	0.092	0.06	**0.085**	0.97	0.096	1.37	0.096	1.27	**0.085**	21.78
($\deg = 4$)	(0.028)	(0.00)	(0.026)	(0.13)	(0.028)	(0.63)	(0.026)	(0.10)	(0.026)	(1.26)
TSP	0.224	0.08	0.214	51.05	0.268	26.65	0.084	52.52	**0.065**	11.18
($\deg = 6$)	(0.123)	(0.01)	(0.147)	(1.28)	(0.342)	(13.38)	(0.088)	(1.36)	(0.060)	(1.08)
Matching	0.325	0.07	0.186	38.11	0.209	21.99	0.179	42.70	**0.178**	43.05
($\deg = 8$)	(0.156)	(0.02)	(0.121)	(6.00)	(0.131)	(11.82)	(0.120)	(1.87)	(0.121)	(1.01)

Table 1: Normalized regret on test dataset (average and standard deviation in parentheses) based on the obtained final model, and execution time per epoch in seconds [s/ep] (mean and standard deviation in parentheses) of existing methods and proposed algorithm with $K = 1$. The normalized regret results of the proposed algorithm for Knapsack and Matching instances are based on the sampling via forward optimization, and those for TSP instance are based on the nearest neighbor algorithm.

in subsequent research.

For **TSP** instance, we generate cost matrix $\boldsymbol{c} = (c_{ij})_{ij}$ according to the same procedure as [Tang and Khalil, 2024, Sect. 5.1.3], which is based on the polynomial model defined by $c_k = \frac{1}{3^{\deg-1}}\left[\frac{1}{\sqrt{d}}[\mathbf{D}^*\boldsymbol{z}]_k + 3\right]^{\deg} \cdot \zeta_k$. Here, each element of the latent matrix $\mathbf{D}^*$ is generated from the multiplication of Bernoulli distributions with mean 0.5 and uniform distribution on $[-2, 2]$. The noise term ζ_k is generated from the uniform distribution on $[1 - \overline{\zeta}, 1 + \overline{\zeta}]$ for some $\overline{\zeta} > 0$. We denote this polynomial model as $\mathrm{poly}_{\mathrm{TSP}}(\boldsymbol{z}; \mathbf{D}^*, \deg, \overline{\zeta})$.

For **Matching** instance, we generate a reward matrix according to the same polynomial model $\mathrm{poly}(\boldsymbol{z}; \mathbf{B}^*, \deg, \overline{\varepsilon})$ as the knapsack instance. The binary matrix $\boldsymbol{m} = (m_{ij})_{ij}$ is generated from Bernoulli distribution with mean 0.5.

The specific settings of parameters for the data generation process for each instance are described in the next subsection.

4.3 Experimental Setup

Baselines. For baselines, we compare the performance of the proposed algorithm with ridge regression as a standard two-stage method (2s) and SPO+ loss (SPO+) [Elmachtoub and Grigas, 2022] as existing methods for the complete-information setting. We also employ the noise contrastive estimation method (NCE) [Mulamba *et al.*, 2021] and perturbed Fenchel–Young loss (PFY) [Berthet *et al.*, 2020] as existing methods for the incomplete-information setting. Note that the inverse optimization-based method proposed in [Hikima and Kamiyama, 2025] could, in principle, be applied under incomplete information; however, it requires solving inverse optimization at every iteration exactly. Preliminary experiments showed that applying this method to the problems considered in this section resulted in prohibitively high computational costs, so we exclude it from the comparison.

Implementation details. For all the experiments, we generate $N_{\mathrm{tr}} = 1024$ and $N_{\mathrm{te}} = 1024$ context–solution pairs for the training and test datasets, respectively. We use a linear model, $h(\boldsymbol{z}; \boldsymbol{\Theta}, \boldsymbol{\theta}_0) = \boldsymbol{\Theta}\boldsymbol{z} + \boldsymbol{\theta}_0$, and train all the models by using stochastic gradient descent method [Amari, 1993] with a constant learning rate $\eta = 5 \cdot 10^{-3}$ and a mini-batch size of $N_{\mathrm{tr}}/4$. We set the maximum number of epochs as $T = 3 \cdot 10^4$ for the two-stage method and $T = 100$ for other methods.

Settings for the existing methods. For NCE method, we set the oracle-call ratio for solving the forward optimization to 0.5, and for PFY method, we set the number of perturbations as one and the perturbation level as 0.5. All the existing methods are implemented based on the Python-based PyEPO package [Tang and Khalil, 2024], which is an open-source software that supports modeling and solving optimization problems. Except for the two-stage approach, all the optimization problems are solved by using the GUROBI optimizer (version 12.0.3) [Gurobi Optimization, LLC, 2024].

Settings for the proposed method. Regarding the proposed method (Algorithm 1), we set $K = 1$, $\sigma = 0.5$, $\lambda = 0$, and $T = 50$. For the distance function in Eq. (2), we use the squared ℓ_2-norm for **Knapsack** and **TSP** instances, and the ℓ_1-norm for **Matching** instance. We report the ablation study on the number of samples K in Sect. 4.6. For sampling feasible solutions, we employed the forward sampling method described in Algorithm 1 and Algorithm 2 for **Knapsack** instance. For the **TSP**, we applied the nearest neighbor algorithm and the permutation-based sampling method as simple and practical choices. For the **Matching**, since no efficient sampling algorithm exists, we used the forward sampling method by solving the forward optimization problem with perturbed cost vectors. Similar to the existing methods, we model and solve the RIOP by using the CVXPY package and GUROBI optimizer (version 12.0.3).

Parameter settings for data generation For **Knapsack** instance, we set $n = 256$, $W = 512$, and the polynomial model is set to be $\mathrm{poly}(\boldsymbol{z}; \mathbf{B}_1^*, 4, 0.75)$. For **TSP** instance, we set the number of cities as $n = 16$, and the polynomial model is set to be $\mathrm{poly}_{\mathrm{TSP}}(\boldsymbol{z}; \mathbf{D}^*, 6, 0.5)$. For **Matching** instance, we set the number of nodes on each side as $N_1 = N_2 = 10$, and the polynomial model is set to be $\mathrm{poly}(\boldsymbol{z}; \mathbf{B}_2^*, 8, 1.0)$. Here, the latent matrix of each instance $\mathbf{B}_i^*$ for $i \in \{1, 2\}$ is generated from the Bernoulli distribution with mean 0.5, and $\mathbf{D}^*$ is generated from the mixture of the Bernoulli distribution with mean 0.5 and the uniform distribution on $[-2, 2]$.

4.4 Evaluation Metric

For one sample of the test data $(\boldsymbol{z}, \boldsymbol{c}) \in \mathcal{D}_{\mathrm{te}}$, we evaluate the performance of the existing methods and the proposed algorithm by the normalized regret, which is defined for a

maximization problem as follows:

$$\text{N-Reg}(h; \boldsymbol{z}, \boldsymbol{c}) \coloneqq \frac{\boldsymbol{c}^\top \boldsymbol{x}^\star(\boldsymbol{c}) - \boldsymbol{c}^\top \boldsymbol{x}^\star(h(\boldsymbol{z}))}{\boldsymbol{c}^\top \boldsymbol{x}^\star(\boldsymbol{c})},$$

where h is a prediction model obtained by each algorithm. Using the test dataset $\mathcal{D}_{\text{te}}$, we report the average and standard deviation of the normalized regret over all samples in $\mathcal{D}_{\text{te}}$.

4.5 Comparison with Existing Methods

Figure 1 shows the learning curves of the proposed algorithm and existing methods on each instance. The vertical axis represents the normalized regret on the training data (left), and test data (right), and the horizontal axis represents the execution time in seconds. The error bands represent an interval of one standard deviation above and below the mean. From the figure, we can see that the proposed algorithm reaches competitive regret values in a shorter wall-clock time than existing methods for all instances.

Table 1 shows the average and standard deviation of normalized regret on the test dataset based on the final model after training, and the average and standard deviation of execution time per epoch in seconds for each method. From the results, we can see that the proposed algorithm achieves comparable or better performance in terms of normalized regret compared to existing methods for all instances. For the **Knapsack** instance, our algorithm takes longer execution time per epoch compared to existing methods. This is likely because the knapsack problem is relatively easy to solve, allowing for faster computation than our algorithm, which requires solving the RIOP in Eq. (2) at each iteration.

In contrast, for the **TSP** instance, we can see that the proposed algorithm is more computationally efficient than existing methods. This is because, while other methods must solve the TSP exactly at each iteration, our algorithm solves the TSP approximately, thereby reducing the computational cost. Notice that the proposed algorithm repeatedly solves the RIOP to obtain the candidate coefficient parameters, but the RIOP with squared ℓ_2-norm as the distance function can be formulated as a quadratic program, which can be solved efficiently with standard optimization solvers.

For the **Matching** instance, the computational time is comparable to existing methods because the proposed algorithm also solves a forward optimization problem to obtain feasible solutions. However, as can be confirmed in Figure 1, the proposed algorithm can achieve faster model training, and Table 1 shows comparable or better performance in terms of normalized regret.

Overall, these results (Figure 1 and Table 1) indicate that the proposed algorithm can effectively learn a prediction model for the contextual inverse optimization problems while maintaining computational efficiency.

4.6 Ablation Study on the Number of Samples K

To investigate the impact of the number of samples K on the performance of Algorithm 1, we conduct an ablation study by varying K. Table 2 reports the normalized regret on the test dataset and execution time per epoch for different $K \in \{1, 4, 16, 64\}$ on the **Knapsack** and **TSP** instances. As sampling strategies in the proposed algorithm, we use Algorithm 2 and forward sampling method for the **Knapsack** instance, and the permutation-based sampling method and nearest neighbor algorithm for the **TSP** instance. The results indicate that even with $K = 1$, the proposed algorithm achieves performance comparable to that obtained with larger values of K. Moreover, the results suggest that higher-quality solutions generated through sampling lead to improved model performance, as reflected by lower normalized regret. However, a theoretical characterization of the relationship between the quality of sampled solutions and the performance of the proposed algorithm remains an open question.

4.7 Computational Complexity

The computational cost of existing methods is mainly governed by the number of times the forward optimization problem is solved. These methods typically require solving the forward optimization problem exactly at each iteration, either to evaluate the loss function or to obtain a gradient estimate for updating the model parameters. In contrast, when an efficient sampling method for generating feasible solutions is available, the proposed algorithm does not require solving the forward optimization exactly. In this case, the computational cost is mainly determined by the number of times the RIOP in Eq. (2) is solved. Thus, the main computational bottleneck shifts from repeatedly solving the forward optimization problem to repeatedly solving the RIOP. When no such sampling method is available, the proposed algorithm must also solve the forward optimization problem to generate feasible solutions, incurring additional computational cost compared with the case in which efficient sampling is available.

5 Related Work

This section reviews related work on CIO in which the available data are limited to context–solution pairs. For prior work on the complete-information setting, readers are referred to the survey papers [Sadana *et al.*, 2024; Mandi *et al.*, 2024].

5.1 Optimality Condition-Based Algorithm

[Sun *et al.*, 2023] and [Mishra *et al.*, 2024] proposed a method for learning the objective coefficients based on the optimality conditions for linear programs defined by $\min_{\boldsymbol{x}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} \;\middle|\; \mathbf{A}\boldsymbol{x} = \boldsymbol{b},\, \boldsymbol{x} \geq \mathbf{0} \right\}$, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. For an arbitrary point $\boldsymbol{q} \in \mathbb{R}^n$ and given optimal solution $\boldsymbol{x}^\star$, the algorithm in [Mishra *et al.*, 2024] solves the quadratic program formulated as

$$\min_{\boldsymbol{c}, \boldsymbol{\nu}, \boldsymbol{\lambda}} \|\boldsymbol{c} - \boldsymbol{q}\|_2^2 \;\text{ s.t. }\; \boldsymbol{\nu}^\top \mathbf{A} - \boldsymbol{c} + \boldsymbol{\lambda} = \mathbf{0},\, \boldsymbol{\lambda}^\top \boldsymbol{x}^\star = 0,\, \boldsymbol{\lambda} \geq \mathbf{0},$$

where $\boldsymbol{\nu} \in \mathbb{R}^m$ and $\boldsymbol{\lambda} \in \mathbb{R}^n_{\geq 0}$ are the Lagrange multipliers for the forward linear program, and the constraints are defined based on the KKT conditions of the linear program, which ensures that the given optimal solution $\boldsymbol{x}^\star$ is optimal with respect to the optimal cost vector $\boldsymbol{c}$ for the above problem. Then, update the model parameters by minimizing the distance between the predicted cost vector and the optimal solution to the above quadratic program.

Both of the aforementioned methods rely on the optimality conditions of linear programs to update the prediction model,

		$K=1$		$K=4$		$K=16$		$K=64$	
instance	sampler	N-Reg	CPU[s/ep]	N-Reg	CPU[s/ep]	N-Reg	CPU[s/ep]	N-Reg	CPU[s/ep]
Knapsack ($\deg=4$)	**Algorithm 2**	0.115 (0.031)	17.15 (1.12)	0.115 (0.032)	19.11 (1.12)	0.115 (0.032)	37.16 (1.93)	0.115 (0.032)	52.14 (2.17)
	forward	0.085 (0.026)	21.78 (1.26)	0.085 (0.026)	39.46 (3.14)	0.085 (0.026)	102.44 (16.86)	0.084 (0.026)	323.33 (47.09)
TSP ($\deg=6$)	**permutation**	0.181 (0.156)	9.22 (0.46)	0.183 (0.162)	10.83 (0.45)	0.184 (0.162)	12.38 (0.34)	0.181 (0.158)	17.18 (0.44)
	nearest neighbor	0.065 (0.062)	10.56 (0.37)	0.062 (0.058)	11.28 (0.32)	0.063 (0.058)	13.84 (0.31)	0.063 (0.060)	22.73 (0.49)

Table 2: Normalized regret (average and standard deviation) and execution time per epoch in seconds (mean and standard deviation).

and their applicability is therefore restricted to settings in which such optimality conditions provide a tractable characterization of optimal solutions. In contrast, [Hikima and Kamiyama, 2025] proposed a more general framework that can be applied to a broad class of optimization with linear objective functions. Their method updates the prediction model by repeatedly solving an inverse optimization associated with the downstream optimization problem at each iteration.

5.2 Perturbed Optimizer-Based Algorithm

[Berthet *et al.*, 2020] proposed a learning algorithm for contextual inverse optimization problems based on the Fenchel–Young loss framework [Blondel *et al.*, 2020]. Given context-solution pairs $\{(\boldsymbol{z}_i, \boldsymbol{x}_i)\}_{i=1}^N \subset \mathbb{R}^d \times \mathbb{R}^n$, one can learn the prediction model h such that $\boldsymbol{x}^\star_\varepsilon(h(\boldsymbol{z}_i)) \approx \boldsymbol{x}_i$, where $\boldsymbol{x}^\star_\varepsilon(\boldsymbol{c})$ is a perturbed optimal solution defined as $\boldsymbol{x}^\star_\varepsilon = \mathbb{E}_{p_{\boldsymbol{c}}(\boldsymbol{x})}[X]$ where $p_{\boldsymbol{c}}(\boldsymbol{x}) = P(\boldsymbol{x}^\star(\boldsymbol{c}+\varepsilon\boldsymbol{\Xi}) = \boldsymbol{x})$, and $\varepsilon > 0$ is a temperature parameter. Here $\boldsymbol{\Xi} \in \mathbb{R}^n$ is a random variable with a positive and differentiable density $\mathrm{d}\mu(\xi) \propto \exp(-\nu(\xi))\mathrm{d}\xi$. Unlike our approach, which constructs explicit regression targets by solving RIOP, the perturbed optimizer-based approach learns through a smoothed approximation of the optimizer induced by random perturbations. As observed in the experiments in Section 4, this difference leads to slower empirical convergence compared with our method.

5.3 Algorithm Based on Enumerating Adjacent Solutions

[Berden *et al.*, 2025] proposed a loss function called *Loss via Adjacent Vertex Alignment* which is defined for a hypothesis $h \in \mathcal{H}$, and a data consisting of context $\boldsymbol{z}$ and an observed optimal solution $\boldsymbol{x}^\star(\boldsymbol{c}) \in \mathcal{X}$ as

$$\ell_{\mathrm{AVA}}(h; \boldsymbol{z}, \boldsymbol{x}^\star(\boldsymbol{c})) \coloneqq \sum_{\boldsymbol{x}_{\mathsf{adj}} \in \mathcal{X}_{\mathsf{adj}}(\boldsymbol{x}^\star(\boldsymbol{c}))} \max\left\{h(\boldsymbol{z})^\top \boldsymbol{x}^\star(\boldsymbol{c}) - h(\boldsymbol{z})^\top \boldsymbol{x}_{\mathsf{adj}}, -\varepsilon\right\},$$

where $\mathcal{X}_{\mathsf{adj}}(\boldsymbol{x}) \subset \mathcal{X}$ represents the set of vertices adjacent to $\boldsymbol{x}$ in the feasible region $\mathcal{X}$, and $\varepsilon \geq 0$ is a margin parameter. To train a prediction model based on the above loss function, it is required to enumerate all adjacent solutions for each observed solution. Therefore, [Berden *et al.*, 2025] focus on linear programs where efficient methods based on performing pivots for enumerating adjacent vertices are known [Berthet *et al.*, 2020; Gal and Geue, 1992]. On the other hand, for combinatorial optimization problems, it is generally difficult to enumerate all adjacent solutions efficiently. In particular, even for specific combinatorial optimization problems such as the $\{0,1\}$-knapsack problem, there is no known efficient method for enumerating all adjacent feasible solutions. Furthermore, it is known that even determining whether two given vertices are adjacent is NP-complete [Matsui, 1995; Matsui and Tamura, 1995].

6 Conclusion

In conclusion, we have studied the contextual inverse optimization problem, which aims to learn a prediction model that estimates the unknown parameters of an underlying optimization problem from observed context–solution pairs. We have proposed an algorithm that iteratively solves a relaxed inverse optimization problem constructed based on feasible solutions sampled from the feasible region of the downstream optimization problem. The proposed algorithm can naturally incorporate a variety of sampling strategies for generating feasible solutions, making it applicable to a broad range of optimization problems. Numerical experiments demonstrate that our approach learns prediction models more efficiently than existing methods while achieving comparable or superior performance in terms of *regret*.

A limitation of the proposed framework is that its computational advantage relies on the availability of efficient methods for sampling feasible solutions. When such methods are unavailable, the forward optimization problem must be solved repeatedly to generate feasible solutions, which may reduce the computational benefits of the proposed approach. Another limitation is that the role of the sampled feasible solutions in the learning process is not yet theoretically characterized. The algorithm requires specifying the number of sampled feasible solutions, and the final performance of the model may depend not only on this number but also on the quality of the sampled solutions. Although our ablation study indicates that even one sampled solution can be sufficient in the tested instances and that higher-quality samples tend to yield better performance, clarifying how the number and quality of sampled feasible solutions affect the learned model remains an important direction for future research.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This research was partially conducted by the Fujitsu Small Research Lab "Division of Fujitsu Mathematical Modeling for Decision Making," a joint research center between Fujitsu Limited and Kyushu University. In addition, this work was supported in part by JST ACT-X, Japan, Grant Number JPMJAX25CK, JST BOOST Program Grant Number JPMJBY24D1, and JSPS KAKENHI Grant Number JP26K21297.

References

[Amari, 1993] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[Amos and Kolter, 2017] Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 136–145. PMLR, 2017.

[Applegate *et al.*, 2011] David L Applegate, Robert E Bixby, Vašek Chvátal, and William J Cook. The traveling salesman problem: a computational study. In *The Traveling Salesman Problem*. Princeton university press, 2011.

[Ben-Tal and Nemirovski, 2002] Aharon Ben-Tal and Arkadi Nemirovski. Robust optimization–methodology and applications. *Mathematical programming*, 92(3):453–480, 2002.

[Berden *et al.*, 2025] Senne Berden, Ali İrfan Mahmutoğulları, Dimos Tsouros, and Tias Guns. Solver-free decision-focused learning for linear optimization problems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[Berthet *et al.*, 2020] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable perturbed optimizers. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9508–9519. Curran Associates, Inc., 2020.

[Bertsimas and Kallus, 2020] Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, March 2020.

[Besbes *et al.*, 2025] Omar Besbes, Yuri Fonseca, and Ilan Lobel. Contextual inverse optimization: Offline and online learning. *Operations Research*, 73(1):424–443, 2025.

[Beyer and Sendhoff, 2007] Hans-Georg Beyer and Bernhard Sendhoff. Robust optimization–a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33-34):3190–3218, 2007.

[Birge *et al.*, 2022] John R Birge, Xiaocheng Li, and Chunlin Sun. Stochastic inverse optimization. Technical report, Working paper, 2022.

[Blondel *et al.*, 2020] Mathieu Blondel, André FT Martins, and Vlad Niculae. Learning with Fenchel-Young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020.

[Bossek *et al.*, 2021] Jakob Bossek, Aneta Neumann, and Frank Neumann. Exact counting and sampling of optima for the knapsack problem. In *International Conference on Learning and Intelligent Optimization*, pages 40–54. Springer, 2021.

[Chatterjee *et al.*, 1996] Sangit Chatterjee, Cecilia Carrera, and Lucy A Lynch. Genetic algorithms and traveling salesman problems. *European Journal of Operational Research*, 93(3):490–510, 1996.

[Christofides, 2022] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. In *Operations Research Forum*, volume 3, page 20. Springer, 2022.

[Dantzig *et al.*, 1954] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, November 1954.

[Delage and Ye, 2010] Erick Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58(3):595–612, 2010.

[Elmachtoub and Grigas, 2022] Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68(1):9–26, January 2022.

[Ferber *et al.*, 2020] Aaron Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. MIPaaL: Mixed integer program as a layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):1504–1511, April 2020.

[Fisher and Yates, 1953] Ronald Aylmer Fisher and Frank Yates. *Statistical tables for biological, agricultural and medical research*. Hafner Publishing Company, 1953.

[Gal and Geue, 1992] Tomas Gal and Ferdinand Geue. A new pivoting rule for solving various degeneracy problems. *Operations Research Letters*, 11(1):23–32, 1992.

[Goemans and Williamson, 1995] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[Gurobi Optimization, LLC, 2024] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[Haribara *et al.*, 2016] Yoshitaka Haribara, Shoko Utsunomiya, and Yoshihisa Yamamoto. A coherent Ising machine for MAX-CUT problems: performance evaluation against semidefinite programming and simulated annealing. *Principles and Methods of Quantum Information Technologies*, 911:251–262, 2016.

[Hikima and Kamiyama, 2025] Yasunari Hikima and Naoyuki Kamiyama. An inverse optimization approach to contextual inverse optimization. In James Kwok, editor,

Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25, pages 5354–5362. International Joint Conferences on Artificial Intelligence Organization, 2025. Main Track.

[Kall and Wallace, 1994] Peter Kall and Stein W Wallace. *Stochastic Programming*, volume 5. Springer, 1994.

[Kotary *et al.*, 2021] James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4475–4482. International Joint Conferences on Artificial Intelligence Organization, 2021. Survey Track.

[Larranaga *et al.*, 1999] Pedro Larranaga, Cindy M. H. Kuijpers, Roberto H. Murga, Inaki Inza, and Sejla Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

[Lin and Kernighan, 1973] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[Malek *et al.*, 1989] Miroslaw Malek, Mohan Guruswamy, Mihir Pandya, and Howard Owens. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Annals of Operations Research*, 21(1):59–84, 1989.

[Mandi *et al.*, 2022] Jayanta Mandi, Víctor Bucarey, Maxime Mulamba Ke Tchomba, and Tias Guns. Decision-focused learning: Through the lens of learning to rank. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 14935–14947. PMLR, 2022.

[Mandi *et al.*, 2024] Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 2024.

[Matsui and Tamura, 1995] Tomomi Matsui and Sunao Tamura. Adjacency on combinatorial polyhedra. *Discrete Applied Mathematics*, 56(2-3):311–321, 1995.

[Matsui, 1995] Tomomi Matsui. NP-completeness of non-adjacency relations on some 0-1-polytopes. *Lecture Notes in Operations Research*, 1:249–258, 1995.

[Mishra *et al.*, 2024] Saurabh Mishra, Anant Raj, and Sharan Vaswani. From inverse optimization to feasibility to ERM. In *Proceedings of the Forty-first International Conference on Machine Learning*, Vienna, Austria, 2024.

[Mišić and Perakis, 2020] Velibor V Mišić and Georgia Perakis. Data analytics in operations management: A review. *Manufacturing & Service Operations Management*, 22(1):158–169, 2020.

[Mulamba *et al.*, 2021] Maxime Mulamba, Jayanta Mandi, Michelangelo Diligenti, Michele Lombardi, Victor Bucarey, and Tias Guns. Contrastive losses and solution caching for predict-and-optimize. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2833–2840. International Joint Conferences on Artificial Intelligence Organization, 2021. Main Track.

[Prékopa, 2013] András Prékopa. *Stochastic Programming*, volume 324. Springer Science & Business Media, 2013.

[Rosenkrantz *et al.*, 1977] Daniel J Rosenkrantz, Richard E Stearns, and Philip M Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977.

[Sadana *et al.*, 2024] Utsav Sadana, Abhilash Chenreddy, Erick Delage, Alexandre Forel, Emma Frejinger, and Thibaut Vidal. A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, 2024.

[Schrijver, 2003] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

[Shapiro *et al.*, 2021] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM, 2021.

[Sun *et al.*, 2023] Chunlin Sun, Shang Liu, and Xiaocheng Li. Maximum optimality margin: A unified approach for contextual linear programming and inverse linear programming. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 32886–32912. PMLR, 2023.

[Tang and Khalil, 2024] Bo Tang and Elias B. Khalil. PyEPO: a Pytorch-based end-to-end predict-then-optimize library for linear and integer programming. *Mathematical Programming Computation*, July 2024.

[Wilder *et al.*, 2019] Bryan Wilder, Bistra Dilkina, and Milind Tambe. 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.

[Zhan *et al.*, 2016] Shi-hua Zhan, Juan Lin, Ze-jun Zhang, and Yi-wen Zhong. List-based simulated annealing algorithm for traveling salesman problem. *Computational Intelligence and Neuroscience*, 2016(1):1712630, 2016.