# An Inverse Optimization Approach to Contextual Inverse Optimization

**Yasunari Hikima**[1,2] , **Naoyuki Kamiyama**[1]
[1]Kyushu University, [2]Fujitsu Limited
hikima.yasunari@fujitsu.com, kamiyama@imi.kyushu-u.ac.jp

## Abstract

Contextual Inverse Optimization (CIO) is a generalized framework of the predict-then-optimize approach, also referred to as decision-focused learning or contextual optimization, aiming to learn a model that predicts the unknown parameters of a nominal optimization problem using related covariates without compromising the solution quality. Unlike the predict-then-optimize approach, which assumes access to datasets containing realized unknown parameters, CIO considers a setting where only historical optimal solutions are available. Previous work has primarily focused on CIO under linear programming problems and proposed methods based on optimality conditions. In this study, we propose a general algorithm based on inverse optimization as a more general approach that does not require optimality conditions. To validate its effectiveness, we apply the proposed method to multiple CIO problems and demonstrate that it performs comparably to or better than existing predict-then-optimize methods, even without ground-truth unknown parameters.

## 1 Introduction

In many decision-making problems modeled as optimization problems, a decision-maker frequently suffers from the fact that some parameters of the optimization model contain uncertainty. For instance, a road planner predicts traffic demand based on information about weather conditions or time of day and presents an optimal route to drivers based on the forecast results. The predict-then-optimize paradigm [El Balghiti *et al.*, 2019; Elmachtoub and Grigas, 2022] studies a setting where the decision-maker has access to a history of covariates which is correlated with the unknown parameter of the optimization model and the realization of the unknown parameters in the past, and the objective is to learn a model that predicts unknown parameters which leads to a better quality of decisions.

One might think that it is sufficient to build a prediction model by using existing statistical or machine learning methods, such as ridge regression, and then subsequently perform an optimization process with the output of the trained model.

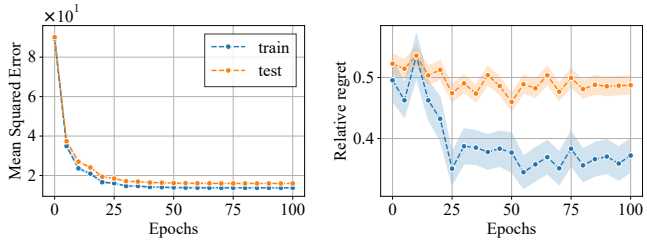



Figure 1: The history of a mean squared loss (left) and a relative *regret* (right) for a shortest path problem trained by ridge regression. The blue (orange) line is the result of training (test) data, and the error bands represent $\pm 1$ of the standard deviation from the average. These results imply that the naive prediction method does not contribute to the quality of the decision of the nominal optimization problem.

Figure 1 shows that such a typical two-stage framework fails to produce a high-quality decision. On the left-hand side of Figure 1, the history of the mean squared loss is shown for predicting the coefficient vector of the shortest path problem. The right-hand side of Figure 1 shows a history of relative *regrets*, which represents the difference between the true optimal value and the objective value calculated by the solution of an optimization problem whose unknown parameters are predicted by a trained model. It can be observed that a model trained using the mean squared error does not significantly contribute to improving the solution quality. Indeed, the regret on the test data (orange line) shows little improvement from the initial parameters.

Besides a predict-then-optimize paradigm, methodologies to support high-quality decision-making in the presence of uncertainty by integrating machine learning and optimization methods have been extensively studied in the name of contextual optimization [Hu *et al.*, 2022; Sadana *et al.*, 2024], decision-focused learning [Amos and Kolter, 2017; Wilder *et al.*, 2019; Mandi *et al.*, 2022] and prescriptive analysis [Bertsimas and Kallus, 2020].

The paradigms mentioned above consider a situation in which the available data contains the realized unknown optimization parameters to be predicted. In contrast, research on what is called *contextual inverse optimization*, which considers a more general situation in which the realized optimal solutions to the nominal optimization problem of interest

are available, has attracted much attention [Sun *et al.*, 2023; Besbes *et al.*, 2023; Mishra *et al.*, 2024].

Prior work on contextual inverse optimization [Sun *et al.*, 2023; Mishra *et al.*, 2024] has basically focused on linear programming problems and proposed learning methods based on the optimality conditions. These methods are fundamentally limited to linear programming problems to which optimality conditions can be applicable, and cannot be directly applied to more general problems, such as integer programming problems, where the feasible region is defined as a subspace of discrete space.

In this paper, to address the limitation of the optimization problem that can be handled in the existing methods for contextual inverse optimization, we propose a general method for solving contextual inverse optimization problems involving integer programming problems that do not explicitly require the optimality conditions.

### 1.1 Contributions

The contributions of this study are summarized as follows:

1. We provide a method for solving contextual inverse optimization problems. Unlike existing work for contextual inverse optimization [Sun *et al.*, 2023; Mishra *et al.*, 2024], the proposed method is not limited to linear programs and can be applied to any linear objective optimization problem, including integer programs in which inverse optimization can be applicable. Furthermore, unlike existing predict-then-optimize methods, our method does not require hyperparameter tuning for the learning rate. As a result, the model can be trained in fewer epochs compared to traditional methods.
2. The proposed method employs an inverse optimization algorithm as an oracle. This study demonstrates that inverse optimization can be effectively utilized as an algorithm for contextual inverse optimization problems. It is known that inverse optimization can be efficiently solved for specific problems like the knapsack problem. As algorithms for inverse optimization continue to advance, it is expected that the proposed method will also improve.
3. We empirically examine the effectiveness of the proposed method for several contextual *integer* optimization problems. Although our method does not need the ground-truth unknown parameters of the nominal optimization problem, the performance is competitive with existing methods such as smart, predict-then-optimize [El Balghiti *et al.*, 2019] that require the unknown parameters and perturbed-based method [Berthet *et al.*, 2020]. Additionally, while the proposed method requires more computation time per iteration than existing methods, we found that the overall computational time is comparable with that of existing methods because it converges after a small number of iterations.

### 1.2 Notation

Let $n$ be a positive integer. Define $\mathbb{N}$ as a set of positive integers. Let $\mathbb{R}^n$ and $\mathbb{Z}^n$ be the set of real-valued vectors of dimension $n$ and the set of integer vectors of dimension $n$, respectively. Additionally, let $\mathbb{R}^n_+$ and $\mathbb{Z}^n_+$ be a set of non-negative real-valued vectors of dimension $n$ and a set of non-negative integer vectors of dimension $n$, respectively. Let $[n]$ be a finite set $\{1, 2, \dots, n\}$. We use lowercase bold letters for vectors and uppercase bold letters for matrices. Let $\boldsymbol{I}_n$ be an identity matrix of size $n$ and $\boldsymbol{O}$ be a zero matrix of appropriate size. For a vector $\boldsymbol{x} \in \mathbb{R}^n$, the symbol $\|\cdot\|$ denotes a general norm on $\mathbb{R}^n$, $\|\cdot\|_p$ denotes the $p$-norm for $p \geq 1$ on $\mathbb{R}^n$, and $\|\cdot\|_\infty$ denotes the maximum norm on $\mathbb{R}^n$.

## 2 Problem Statement

Let $n$ and $d$ be positive integers. Given a feasible region $\mathcal{X} \subseteq \mathbb{Z}^n$, consider an integer programming problem[1] (IP) which is formulated as follows:

$$\mathrm{IP}(\boldsymbol{c}, \mathcal{X}): \quad \underset{\boldsymbol{x}}{\text{maximize}} \quad \boldsymbol{c}^\top \boldsymbol{x} \tag{1a}$$

$$\text{subject to} \quad \boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{Z}^n, \tag{1b}$$

where $\boldsymbol{x} \in \mathcal{X}$ is a decision variable and $\boldsymbol{c} \in \mathcal{C}$ is an unknown parameter vector. Here, $\mathcal{C} \subseteq \mathbb{R}^n$ is a domain in which the vector $\boldsymbol{c}$ lives. Let $\boldsymbol{x}^\star(\boldsymbol{c}) \in \mathcal{X}$ be an optimal solution to the $\mathrm{IP}(\boldsymbol{c}, \mathcal{X})$. The exact value of a vector $\boldsymbol{c}$ is not known in advance, but a covariate $\boldsymbol{z} \in \mathbb{R}^d$ correlated with $\boldsymbol{c}$ is available. In other words, we have access to a historical dataset $\mathcal{D}_N = \{(\boldsymbol{z}_i, \boldsymbol{x}_i^\star)\}_{i=1}^N$, which consists of a pair of a covariate and an optimal solution. Here, for any $i \in [N]$, $\boldsymbol{x}_i^\star$ is an optimal solution to the $\mathrm{IP}(\boldsymbol{c}_i, \mathcal{X})$, where $\boldsymbol{c}_i$ is the true cost vector realized when the covariate is $\boldsymbol{z}_i$.

In contrast to the predict-then-optimize setting, the contextual inverse optimization setting is more challenging because true cost vectors are not included in the dataset. Thus, even simple two-stage methods cannot be directly applied. Indeed, the predict-then-optimize setting can be reduced to the contextual inverse optimization setting by solving the optimization problem for the true cost vectors included in the dataset.

The primary objective of the contextual inverse optimization setting is to obtain a prediction model that maps a covariate to an unknown coefficient vector of a nominal optimization problem, which leads to a quality decision. Denote the prediction model by $f(\cdot; \boldsymbol{\Theta})\colon \mathbb{R}^d \to \mathcal{C}$ where $\boldsymbol{\Theta}$ is a model parameter to be trained by some methods.

In the typical supervised learning setting, the performance metric is considered how close the predicted cost $\boldsymbol{c}_{\text{pred}} = f(\boldsymbol{z}; \boldsymbol{\Theta})$ is to the true cost $\boldsymbol{c}$. However, in the problem setting described above, the ultimate goal is to obtain an optimal solution $\boldsymbol{x}^\star(\boldsymbol{c}_{\text{pred}})$ that is close to the optimal solution $\boldsymbol{x}^\star(\boldsymbol{c})$. To measure how well the prediction model performs, the *regret* for the maximization problem in Eq. (1) is defined as

$$\mathrm{Regret}(\boldsymbol{c}_{\text{pred}}; \boldsymbol{c}) := \boldsymbol{c}^\top \boldsymbol{x}^\star(\boldsymbol{c}) - \boldsymbol{c}^\top \boldsymbol{x}^\star(\boldsymbol{c}_{\text{pred}}), \tag{2}$$

which measures the difference between the optimal objective value $\boldsymbol{c}^\top \boldsymbol{x}^\star(\boldsymbol{c})$ and the objective value $\boldsymbol{c}^\top \boldsymbol{x}^\star(\boldsymbol{c}_{\text{pred}})$ when the decision is $\boldsymbol{x}^\star(\boldsymbol{c}_{\text{pred}})$ with ground truth $\boldsymbol{c}$. Note that the metric becomes 0 when the optimal solution with the predicted cost vector is identical to that of the true optimal solution.

[1] This study can naturally be extended to mixed-integer linear programming problems; however, in this paper, we focus on integer programming problems.

## 3 Algorithm

Denote the integer programming problem of interest by $\mathrm{IP}(\boldsymbol{c}, \mathcal{X})$ formulated in Eq. (1) where $\mathcal{X} \subseteq \mathbb{Z}^n$ is a feasible region and $\boldsymbol{c} \in \mathbb{R}^n$ is a cost vector of the objective function. Let $\boldsymbol{x}^0 \in \mathcal{X}$ be a feasible solution to the problem $\mathrm{IP}(\boldsymbol{c}, \mathcal{X})$. For a reference vector $\boldsymbol{r} \in \mathbb{R}^n$, the inverse optimization problem is defined as the following optimization problem:

$$\mathrm{IOP}(\boldsymbol{r}, \boldsymbol{x}^0, \|\cdot\|): \tag{3a}$$

$$\underset{\boldsymbol{q}}{\text{minimize}} \quad \|\boldsymbol{q} - \boldsymbol{r}\| \tag{3b}$$

$$\text{subject to} \quad \boldsymbol{x}^0 \in \arg\max \left\{ \boldsymbol{q}^\top \boldsymbol{y} \;\middle|\; \boldsymbol{y} \in \mathcal{X} \right\}, \tag{3c}$$

$$\boldsymbol{q} \in \mathbb{R}^n. \tag{3d}$$

Notice that the optimal solution to the above optimization problem in Eq. (3) determines the optimal profit vector $\boldsymbol{q}^\star$ that minimizes the distance from a given $\boldsymbol{r}$ with respect to a norm $\|\cdot\|$ such that a given feasible solution $\boldsymbol{x}^0$ be an optimal solution to the modified optimization problem $\mathrm{IP}(\boldsymbol{q}^\star, \mathcal{X})$.

Using an available dataset $\mathcal{D}_N = \{(\boldsymbol{z}_i, \boldsymbol{x}_i^\star)\}_{i=1}^N$, we aim to find a model parameter $\boldsymbol{\Theta}^\star$ so that the optimal solution to $\mathrm{IP}(\widehat{\boldsymbol{c}}_{\text{new}}, \mathcal{X})$, where $\widehat{\boldsymbol{c}}_{\text{new}} = f(\boldsymbol{z}_{\text{new}}; \boldsymbol{\Theta}^\star)$, is close to the optimal solution to $\mathrm{IP}(\boldsymbol{c}_{\text{new}}, \mathcal{X})$. Here, $\boldsymbol{z}_{\text{new}}$ is a new observation of the context, and $\boldsymbol{c}_{\text{new}}$ is a ground-truth cost vector of IP of interest. In light of the above, for each data point $(\boldsymbol{z}_i, \boldsymbol{x}_i^\star)$, we first predict a cost vector by the current model $\widehat{\boldsymbol{c}}_i = f(\boldsymbol{z}_i; \boldsymbol{\Theta})$ and obtain a perturbed vector $\widetilde{\boldsymbol{c}}_{i,k} = \widehat{\boldsymbol{c}}_i + \boldsymbol{\varepsilon}_k$ where $\boldsymbol{\varepsilon}_k \sim \mathcal{N}(\boldsymbol{0}, \sigma \boldsymbol{I}_n)$ for some $\sigma > 0$ and $k = 1, 2, \ldots, M$. Then, we solve the inverse optimization problem in Eq. (3) for each perturbed cost vector, and obtain their solutions $\widetilde{\boldsymbol{q}}_{i,k}$ for $k \in [M]$. Finally, we update the current model parameter as the optimal solution to the following optimization problem:

$$\boldsymbol{\Theta}^\star \in \arg\min_{\boldsymbol{\Theta}} \left\{ \frac{1}{2} \sum_{i=1}^N \frac{1}{M} \sum_{k=1}^M \|f(\boldsymbol{z}_i; \boldsymbol{\Theta}) - \widetilde{\boldsymbol{q}}_{i,k}\|_2^2 \right\}, \tag{4}$$

where $\{\widetilde{\boldsymbol{q}}_{i,k}\}_{i \in [N], k \in [M]} \subseteq \mathbb{R}^n$ are optimal solutions to $\mathrm{IOP}(\widetilde{\boldsymbol{c}}_{i,k}, \boldsymbol{x}^\star, \|\cdot\|)$ for some norm $\|\cdot\|$ in Eq. (3). We present the pseudocode of the proposed algorithm to solve the contextual inverse optimization in Algorithm 1.

The proposed update rule in Eq. (4) is related to minimize the loss function defined by $\ell(\boldsymbol{\Theta}) := \mathbb{E}_{(\boldsymbol{z}, \boldsymbol{x}) \sim P} [h(\boldsymbol{\Theta}; \boldsymbol{z}, \boldsymbol{x})]$ where $h(\boldsymbol{\Theta}; \boldsymbol{z}, \boldsymbol{x}) := \frac{1}{2} \mathbb{E}_{\boldsymbol{q}(\boldsymbol{x}) \sim P_{\boldsymbol{x}}} \left[ \|f(\boldsymbol{z}; \boldsymbol{\Theta}) - \boldsymbol{q}(\boldsymbol{x})\|_2^2 \right]$. Here, $P$ and $P_{\boldsymbol{x}}$ denote the underlying distribution of the data and the distribution on the set of all coefficient vectors whose optimal solution to $\mathrm{IP}(\boldsymbol{c}, \mathcal{X})$ is $\boldsymbol{x}$, respectively. By minimizing $\ell(\boldsymbol{\Theta})$, it is expected that minimizing the model parameter $\boldsymbol{\Theta}$ can be obtained to prescribe a high-quality solution based on the new context $\boldsymbol{z}$.

### 3.1 Linear Model Case

We discuss the case of a linear model $f(\boldsymbol{z}; \boldsymbol{\Theta}) = \boldsymbol{z}\boldsymbol{\Theta}$ where $\boldsymbol{z} \in \mathbb{R}^{1 \times d}$. Let $\mathcal{L}(\boldsymbol{\Theta})$ be an objective function of the convex quadratic optimization problem in Eq. (4). Taking gradient of the objective function of the problem in Eq. (4), $\mathcal{L}(\boldsymbol{\Theta})$, with respect to $\boldsymbol{\Theta}$, we have

$$\nabla_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\Theta}) = \sum_{i=1}^N \frac{1}{M} \sum_{k=1}^M \boldsymbol{z}_i^\top \left( \boldsymbol{z}_i \boldsymbol{\Theta} - \widetilde{\boldsymbol{q}}_{i,k}^\top \right).$$

**Algorithm 1:** Algorithm for Contextual Inverse Optimization Problem (full batch size)

**Input:** Dataset $\mathcal{D}_N = \{(\boldsymbol{z}_i, \boldsymbol{x}_i^\star)\}_{i=1}^N$, $\mathrm{IOP}(\cdot, \cdot, \|\cdot\|)$ as an oracle, $T > 0$, $\sigma > 0$, $M > 0$, and $\boldsymbol{\Theta}_0$.
**Output:** Model parameter $\boldsymbol{\Theta}$.
**Initialization:** Set $t \leftarrow 0$, and $\boldsymbol{\Theta}^{(t)} \leftarrow \boldsymbol{\Theta}_0$.

```
while t < T do
    for i = 1, 2, ..., N do
        ĉ_i ← f(z_i; Θ^(t)).
        for k = 1, 2, ..., M do
            Draw ε_k ~ N(0, σ I_n).
            c̃_{i,k} ← ĉ_i + ε_k.
            q̃_{i,k} ← IOP(c̃_{i,k}, x_i^⋆, ‖·‖).
    Update Θ^(t) by Eq. (4).
    Update t ← t + 1.
return Θ^(T).
```

Define $\overline{\boldsymbol{q}}_i := \frac{1}{M} \sum_{k=1}^M \widetilde{\boldsymbol{q}}_{i,k}$. Then, by introducing matrices $\boldsymbol{Z} := \left[\boldsymbol{z}_1^\top, \boldsymbol{z}_2^\top, \ldots, \boldsymbol{z}_N^\top\right]^\top \in \mathbb{R}^{N \times d}$ and $\overline{\boldsymbol{Q}} = [\overline{\boldsymbol{q}}_1, \overline{\boldsymbol{q}}_2, \ldots, \overline{\boldsymbol{q}}_N] \in \mathbb{R}^{n \times N}$, we have

$$\nabla_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{Z}^\top \boldsymbol{Z} \boldsymbol{\Theta} - \boldsymbol{Z}^\top \overline{\boldsymbol{Q}}^\top.$$

By solving a linear system $\nabla_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{O}$ under the condition that $\boldsymbol{Z}^\top \boldsymbol{Z}$ is invertible, we have the following closed form for the optimization problem of Eq. (4):

$$\boldsymbol{\Theta}^\star = (\boldsymbol{Z}^\top \boldsymbol{Z})^{-1} \boldsymbol{Z}^\top \overline{\boldsymbol{Q}}^\top. \tag{5}$$

Unlike learning-based methods, the proposed method does not require setting a learning rate, which eliminates the need for hyperparameter tuning. Algorithm 1 has two parameters, $M$ and $\sigma$, but we confirm that setting both to 1 achieves sufficiently high performance in Section 4.

### 3.2 Inverse Integer Programming Problem

To implement Algorithm 1, it is necessary to have an inverse optimization oracle $\mathrm{IOP}(\cdot, \cdot, \|\cdot\|)$ for some norm $\|\cdot\|$. Below, we review the prior work on solving inverse problems for integer programs [Ahuja and Orlin, 2000; Heuberger, 2004].

The inverse integer programming problems under $\ell_1$ norm are known to be reformulated into a linear programming problem [Schaefer, 2009]. However, the number of both decision variables and constants grows exponentially, which makes it impractical to solve in a realistic time. It has been shown in [Wang, 2009] and [Bulut and Ralphs, 2021] that inverse integer programming problems can be solved by a cutting-plane method. Given $\boldsymbol{x}^0$ and a reference point $\boldsymbol{r}$, the cutting-plane algorithm solves the following problem at the $t$th iteration:

$$\underset{\boldsymbol{q}}{\text{minimize}} \quad \|\boldsymbol{q} - \boldsymbol{r}\| \tag{6a}$$

$$\text{subject to} \quad \boldsymbol{q}^\top \boldsymbol{x} \leq \boldsymbol{q}^\top \boldsymbol{x}^0 \quad \forall \boldsymbol{x} \in \mathcal{E}_t, \tag{6b}$$

where $\mathcal{E}_t = \left\{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^t\right\}$ is the optimal solutions generated in the previous iterations so far. After obtaining the

**Algorithm 2:** A Cutting-Plane Algorithm for Inverse Integer Programming Problem (c.f., [Wang, 2009])

**Input:** Integer program $\mathrm{IP}(\boldsymbol{c}, \mathcal{X})$, feasible solution $\boldsymbol{x}^0 \in \mathcal{X}$, and reference point $\boldsymbol{r}$.
**Output:** Cost vector $\boldsymbol{q}$ such that the optimal solution to revised problem $\mathrm{IP}(\boldsymbol{q}, \mathcal{X})$ is $\boldsymbol{x}^0$.

1 **Initialization:** Set $\boldsymbol{q}_1 = \boldsymbol{r}$, $\boldsymbol{x}_1$ be an optimal solution to problem in Eq. (7), $\mathcal{E}_1 \leftarrow \{\boldsymbol{x}_1\}$, and $t \leftarrow 1$.
2 **while** $\boldsymbol{q}_t^\top (\boldsymbol{x}^t - \boldsymbol{x}^0) > 0$ **do**
3 Solve optimization problem (6) and obtain $\boldsymbol{q}_t$.
4 Solve integer program (7) and obtain $\boldsymbol{x}_t$.
5 Update $\mathcal{E}_{t+1} \leftarrow \mathcal{E}_t \cup \{\boldsymbol{x}_t\}$ and $t \leftarrow t + 1$.
6 **return** $\boldsymbol{q}_t$.

optimal solution $\boldsymbol{q}_t$ to the problem (6), the algorithm solves the following integer programming problem:

$$\underset{\boldsymbol{x} \in \mathcal{X}}{\text{maximize}} \quad \boldsymbol{q}_t^\top \boldsymbol{x}. \tag{7}$$

If the condition $\boldsymbol{q}_t^\top \left(\boldsymbol{x}^t - \boldsymbol{x}^0\right) \leq 0$ satisfies, then the algorithm terminates and outputs $\boldsymbol{q}_t$ as an optimal solution to the inverse optimization problem; otherwise update $\mathcal{E}_{t+1} = \mathcal{E}_t \cup \{\boldsymbol{x}^t\}$ and repeat the above procedure. We present the pseudo-code of the cutting plane algorithm in 2. It has been proven that Algorithm 2 terminates in a finite number of iterations and outputs an optimal solution to the inverse integer programming problem [Wang, 2009, Theorem 1, Theorem 2].

Note that inverse optimization for mixed-integer programming problems has also been studied, and algorithms based on the branch-and-bound [Wang, 2013] and cutting-plane method [Bulut and Ralphs, 2021; Bodur *et al.*, 2022] have been proposed. Thus, the proposed method can be naturally extended to solve contextual inverse mixed-integer programming problems.

### 3.3 Inverse Optimization of Specific Problems

In Section 3.2, we have discussed that any inverse optimization problems can be solved by the cutting-plane algorithm. However, the cutting-plane algorithm may not be efficient because it requires solving optimization problems iteratively as a subproblem at each iteration (c.f., line 3 and 4 in Algorithm 2). In this subsection, we discuss that inverse optimization problems can be solved more efficiently by considering the specific structure of the target optimization problem. Here, we take the $\{0,1\}$-knapsack problem as an example [Roland *et al.*, 2013], but other optimization problems, such as shortest path problems [Burton and Toint, 1992; Call and Holmberg, 2011] and matching problems [Zhang *et al.*, 1999; Huang and Liu, 1999], have also been studied. It should be mentioned that the knapsack problem is an important combinatorial optimization problem because it can be used to model resource allocation under constraints, providing a foundation for optimizing decisions in finance, logistics, and scheduling through combinatorial optimization techniques.

Consider the $\{0,1\}$-knapsack problem in which a decision maker selects a pair of given $n$ items numbered as $\mathcal{I} = [n]$ with the aim of maximizing the sum of utility under a capacity constraint. Formally, the $\{0,1\}$-knapsack problem is formulated as follows:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \boldsymbol{c}^\top \boldsymbol{x} && \text{(8a)} \\ \text{subject to} \quad & \boldsymbol{w}^\top \boldsymbol{x} \leq W, && \text{(8b)} \\ & x_i \in \{0,1\} \quad \forall i \in \mathcal{I}, && \text{(8c)} \end{aligned}$$

where $\boldsymbol{c} = (c_i)_{i=1}^n$ is a utility of items, $\boldsymbol{w} = (w_i)_{i=1}^n$ is items weights, and $W \in \mathbb{N}$ is a maximum weight capacity.

Let $g_i(\mu) \in \mathbb{R}$ be the maximum utility that can be achieved when considering the first $i$ items of the set of items $\mathcal{I}$ and a capacity $\mu \in \{0, 1, \ldots, W\}$. The value of $g_i(u)$ can be determined through the optimal value of the following combinatorial optimization problem:

$$\begin{aligned} g_i(\mu) \coloneqq \underset{x}{\text{maximize}} \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} \quad & \sum_{j=1}^{i} w_j x_j \leq \mu, \\ & x_j \in \{0,1\} \qquad \forall j \in [i]. \end{aligned}$$

The optimal value to the original $\{0,1\}$-knapsack problem in Eq. (8) is equivalent to finding the value of $g_n(W)$. Notice that the $\{0,1\}$-knapsack problem can be solved by the dynamic programming formulated as

$$g_i(\mu) = \begin{cases} g_{i-1}(\mu) & \text{if } \mu < w_i, \\ \max\left\{g_{i-1}(\mu), g_{i-1}(\mu - w_i) + c_i\right\} & \text{if } \mu \geq w_i, \end{cases}$$

with the initial state $g_1(\mu) = 0$ for $\mu \in \{0, 1, \ldots, w_1 - 1\}$ and $g_1(\mu) = c_1$ for $\mu \in \{w_1, w_1 + 1, \ldots, W\}$. This dynamic programming approach can be reduced to a linear programming problem to minimize $g_n(W)$ under the set of linear constraints of $g_i(\mu)$ for $i \in [n]$, $\mu \in \{0, 1, \ldots, W\}$. For a given feasible solution $\boldsymbol{x}^0 \in \mathbb{R}^n$ and a reference point $\boldsymbol{r} \in \mathbb{R}^n$, [Roland *et al.*, 2013] has shown that the following linear programming problem finds a vector $\boldsymbol{d}^\star \in \mathbb{R}_+^n$ with $(\boldsymbol{q}^\star)^\top \boldsymbol{x}^0 = \max\left\{(\boldsymbol{q}^\star)^\top \boldsymbol{x} \;\middle|\; \boldsymbol{x} \in \mathcal{X}\right\}$ under some norm $\|\cdot\|$:

$$\begin{aligned} \underset{\boldsymbol{q}, (g_i(\mu))_{i,\mu}}{\text{minimize}} \quad & \|\boldsymbol{q} - \boldsymbol{r}\| && \text{(10a)} \\ \text{subject to} \quad & \sum_{j \in \mathcal{I}} q_j x_j^0 \geq g_n(W), && \text{(10b)} \\ & \text{The set of constraints of } g_i(\mu), && \text{(10c)} \\ & \boldsymbol{q} \geq \boldsymbol{0}. && \text{(10d)} \end{aligned}$$

where $\boldsymbol{q} \in \mathbb{R}_+^n$ and $(g_i(\mu))_{i,\mu}$ are decision variables. In case we take $\|\cdot\|$ as $\ell_1$ norm or $\ell_\infty$ norm, the above problem is reduced to be a linear programming problem.

## 4 Numerical Experiments

To investigate the effectiveness of the proposed method, we conducted numerical experiments on several optimization problems and compared the proposed method against the existing methods for a predict-then-optimize framework.

Specifically, we consider the $\{0,1\}$-knapsack problem formulated in Eq. (8) and the diverse bipartite matching problem

adapted in [Ferber *et al.*, 2020] following the synthetic data generation process studied in [Elmachtoub and Grigas, 2022]. As for existing methods, we employed a ridge regression as a two-stage approach (2s), smart predict-then-optimize (spo+) [Elmachtoub and Grigas, 2022], differentiation of black box combinatorial solvers (dbb) [Vlastelica *et al.*, 2020], a noise contrastive estimation approach (nce) [Mulamba *et al.*, 2021], a learning to rank approach (ltr) [Mandi *et al.*, 2022], and a perturbed optimizers approach (pfy) [Berthet *et al.*, 2020].

We implemented all the existing methods by using the `pyepo` package [Tang and Khalil, 2024] and trained all the methods for 100 number of epochs with the full batch size. For existing methods, we employed Adam optimizer [Kingma and Ba, 2014] with a learning rate $\eta = 10^{-2}$ to minimize each loss function. For the learning to rank methods, we employed pairwise loss and listwise loss, and we set the ratio of calling the optimization oracle as 0.5. For the perturbed optimizer approach, we set both the number of perturbations and the amplitude of the perturbation as 1. For the proposed method (Algorithm 1), we set $T = 100$, $M = 1$, $\sigma = 1$, and we solve the inverse optimization problem by the optimization problem given in Eq. (10) for a $\{0,1\}$-knapsack problem and the cutting-plane algorithm (Algorithm 2) for a diverse bipartite matching problem. For both methods, we use the linear model, $f(\boldsymbol{z}; \boldsymbol{\Theta}) = \boldsymbol{z}\boldsymbol{\Theta}$, to predict a coefficient vector.

It should be mentioned that several approaches to contextual inverse optimization problems have been proposed [Sun *et al.*, 2023; Mishra *et al.*, 2024]; however, these methods are limited to linear programming problems (and quadratic programs) to which optimality conditions can be exploited, and thus we excluded these methods in this experiment.

Experimental programs were implemented in Python 3.11 and run on a Windows 10 PC with an Intel(R) Xeon(R) W-1270 CPU 3.40 GHz and 64 GB RAM. For both the existing and proposed methods, we used the Gurobi Optimizer 11.0.2 [Gurobi Optimization, LLC, 2024] to solve the optimization problems.

### 4.1 Problem Description

Since we have already described the $\{0,1\}$-knapsack problem in Section 3.3, we omit the problem description here. We present the problem definition of the diverse bipartite matching problem adapted in [Ferber *et al.*, 2020], in which the objective is to find an optimal matching between the scientific publications with the desired proportion. This problem is formulated as the following integer programming problem:

$$\underset{\boldsymbol{x}}{\text{maximize}} \quad \sum_{i \in N_1} \sum_{j \in N_2} c_{i,j} x_{i,j} \tag{11a}$$

$$\text{subject to} \quad \boldsymbol{x} \in \mathcal{X}(\boldsymbol{m}, p, q), \tag{11b}$$

where $N_1$ and $N_2$ are the sets of nodes representing scientific publications, $\mathcal{X}(\boldsymbol{m}, p, q)$ is a feasible set, and $\boldsymbol{c} = (c_{i,j})_{\forall i \in N_1, \forall j \in N_2}$ is a reward matrix which indicates the likelihood that a link between each pair of nodes from $N_1$ to $N_2$ exists. Also, $\boldsymbol{m} = (m_{i,j})_{\forall i \in N_1, \forall j \in N_2}$ is a binary matrix indicating the relationship between publications, and $p, q \in [0,1]$ are proportions of sharing each field. The formal description of the constraints is referred to [Ferber *et al.*, 2020].

### 4.2 Synthetic Data Generation

Next, we provide the synthetic data generation process for the optimization problems to be solved. A vector of a covariate $\boldsymbol{z} \in \mathbb{R}^d$ is generated from the normal distribution $\mathcal{N}(\mu, \sigma^2)$ with a mean $\mu = 0$ and variance $\sigma^2 = 1$, and a coefficient vector $\boldsymbol{c} \in \mathbb{R}^n$ is generated from a polynomial mapping from the covariates with additional noise. Formally, each element of the cost vector is generated as per the following model:

$$c_j = \frac{5}{3.5^{\deg}} \left[ \left( \frac{1}{\sqrt{d}} (\boldsymbol{B}^\star \boldsymbol{z})_j + 3 \right)^{\deg} + 1 \right] \cdot \varepsilon_j, \tag{12}$$

where $\deg > 0$ is a fixed positive integer representing a polynomial degree, $\boldsymbol{B}^\star \in \mathbb{R}^{n \times d}$ is a latent true coefficient matrix whose each entry $B^\star_{kl}$ $(k \in [n], l \in [d])$ takes 1 with probability 0.5 and 0 with probability 0.5, and $\varepsilon_j$ is a multiplicative noise term that is sampled from the uniform distribution on $[1 - \bar{\varepsilon}, 1 + \bar{\varepsilon}]$ for some constant parameter $\bar{\varepsilon} > 0$.

For the $\{0,1\}$-knapsack problem in Eq. (8), we set the dimensionality of the covariates as $d = 5$, the number of items as $n = 20$, and the maximum capacity of the knapsack as $W = 30$. The utility vector of each item $\boldsymbol{c}$ is generated according to the polynomial model in Eq. (12). The weights of each item are generated as $w_i = \lfloor \frac{1}{100} u_i \rfloor$ where $u_i \sim \text{Unif}(\{300, 301, \ldots, 799\})$.

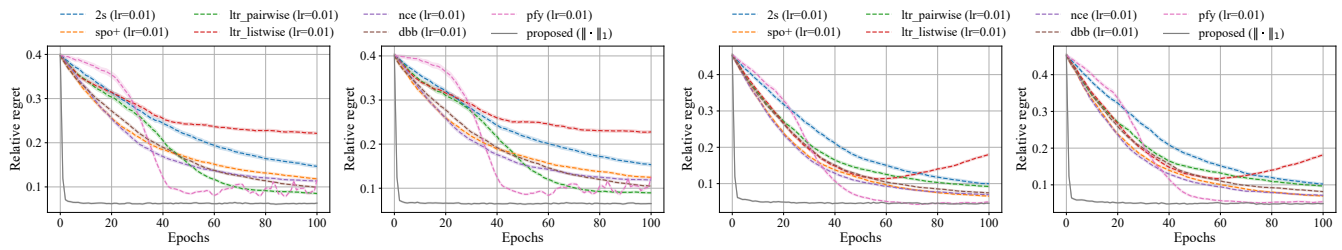
For the diverse bipartite matching problem in Eq. (11), we set the dimensionality of the covariates as $d = 5$, the number of nodes in $N_1$ and $N_2$ as $|N_1| = |N_2| = 5$, the desired proportion as $p = q = 0.25$, and the reward matrix $\boldsymbol{c}$ is generated according to the polynomial model in Eq. (12). Note that the original bipartite matching problem [Ferber *et al.*, 2020] is formulated on graphs sampled from the CORA citation network dataset [Sen *et al.*, 2008], but we consider the synthetic data generation process because the size of the original optimization problem is large, so that the proposed method does not work in realistic time.

For the experiments, we first construct a dataset consists of $\widetilde{\mathcal{D}}_N = \{(\boldsymbol{z}_i, \boldsymbol{c}_i, \boldsymbol{x}^\star_i)\}_{i=1}^N$ where $\{\boldsymbol{z}_i\}_{i=1}^N$ and $\{\boldsymbol{c}_i\}_{i=1}^N$ are generated by the above procedure, and $\{\boldsymbol{x}^\star_i\}_{i=1}^N$ are the optimal solutions to the optimization problem of interest when the unknown parameters are given by $\{\boldsymbol{c}_i\}_{i=1}^N$. Existing predict-then-optimize methods require $\overline{\mathcal{D}}_N = \{(\boldsymbol{z}_i, \boldsymbol{c}_i)\}_{i=1}^N$ as the dataset, whereas our method only require $\mathcal{D}_N = \{(\boldsymbol{z}_i, \boldsymbol{x}^\star_i)\}_{i=1}^N$. For all experiments, we set the number of training samples as $N = 1000$ and the number of test samples as $N = 1000$.

### 4.3 Performance Metrics

In this numerical experiment, especially for a maximization problem, we consider relative regret as a performance measure. For the predicted coefficient vector $\widehat{\boldsymbol{c}}$ of the objective function, the relative regret is defined as

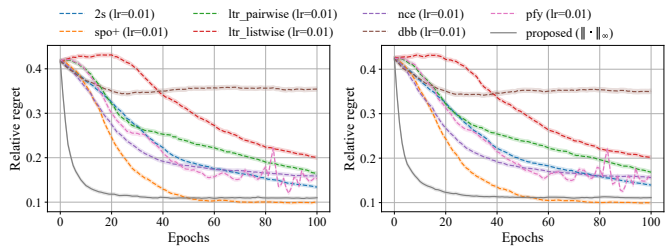
$$\text{Rel-Regret}(\widehat{\boldsymbol{c}}; \boldsymbol{c}) := \frac{\boldsymbol{c}^\top \boldsymbol{x}^\star(\boldsymbol{c}) - \boldsymbol{c}^\top \boldsymbol{x}^\star(\widehat{\boldsymbol{c}})}{\boldsymbol{c}^\top \boldsymbol{x}^\star(\boldsymbol{c})}, \tag{13}$$



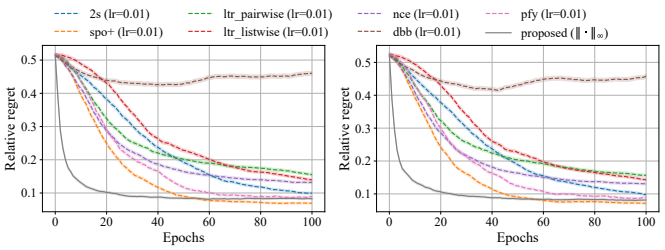

(a) **Knapsack** ($\deg = 2$)




(b) **Knapsack** ($\deg = 4$).




(c) **Matching** ($\deg = 2$)




(d) **Matching** ($\deg = 4$).

Figure 2: The history of relative regret for the $\{0,1\}$-knapsack problem with a polynomial degree $\deg \in \{2,4\}$ (2a, 2b) and for the diverse bipartite matching problem with a polynomial degree $\deg \in \{2,4\}$ (2c, 2d). The left figures of both (2a, 2b) and (2c, 2d) show the result for training data ($n = 1000$), and the right figure shows the result for test data ($n = 1000$).

where $\boldsymbol{c}$ and $\boldsymbol{x}^\star(\cdot)$ are the true coefficient vector and the optimal solution, respectively. For the dataset $\widetilde{\mathcal{D}}_N$, we report the average and standard deviation of the relative regret.

### 4.4 Results

**Performance** In Figure 2, we show the history of an average of relative regret in Eq. (13) for each method to the $\{0,1\}$-knapsack problem with a polynomial degree $\deg \in \{2,4\}$ (Figures (2a,2b)) and the diverse bipartite matching problem with $\deg \in \{2,4\}$ (Figures (2c, 2d)), respectively. The error bands represent an interval $\pm 1$ of the standard deviation from the average. Regarding the proposed method, we report the results of Algorithm 1 with $\mathrm{IOP}(\cdot,\cdot,\|\cdot\|_1)$ for the $\{0,1\}$-knapsack problem, and Algorithm 1 with $\mathrm{IOP}(\cdot,\cdot,\|\cdot\|_\infty)$ for the diverse bipartite matching problem.

As an overall trend, we can see that the proposed method shows a rapid decrease in relative regret in the early stages of learning compared to other predict-then-optimize methods. This is because the proposed method is not a learning-based method, and thus it does not require the learning rate tuning. Among the existing methods, spo+ and pfy perform well, which is consistent with the results reported in numerical experiments in prior work (e.g., [Mandi *et al.*, 2022; Sun *et al.*, 2023; Mishra *et al.*, 2024]). spo+ is the best for the diverse bipartite matching problem. In addition, pfy shows good performance, but the results show unstable when $\deg = 2$ for both problems. Finally, we emphasize again that our proposed method does not need information about the true cost vectors, unlike all other predict-then-optimize methods, except for the perturbed method. Nevertheless, it is remarkable that the proposed method is competitive with the baseline methods.

**Computational time** From Figure 2, we observe that the proposed method performs as well as or better than existing methods. However, the proposed method needs to solve an inverse optimization problem at each epoch, which is expected to result in higher computational costs compared to existing methods. Therefore, we summarized the computational time per epoch in seconds for each method in Table 1. Although the proposed method incurs several tens of times the computational cost compared to existing methods, it has the advantage of demonstrating high performance in the early stages of training, as shown in Figure 2.

**Effect of $M$ in Algorithm 1** So far, we have reported the results of the proposed method with parameter $M = 1$. Here, we examine how Algorithm 1 performs across different values of $M$. In this experiment, we set the maximum number of epochs as $T = 10$. Table 2 presents the mean and standard deviation of the relative regret, and the computational time per epoch for the proposed method with $M$ set to 1, 2, 5, and 10. The results demonstrate that increasing the value of $M$ does not significantly change the performance of the resulting model, which suggests that setting $M = 1$ is sufficient for practical applications.

## 5 Related Work

This section provides related work that aims to incorporate contextual information about a nominal optimization problem into the learning process for a prediction model. We review the methods for contextual inverse optimization. For a comprehensive overview of contextual optimization and decision-focused learning, we refer the reader to review papers [Mandi *et al.*, 2024; Sadana *et al.*, 2024].

Contextual inverse optimization is a field that aims to learn a prediction model for unknown parameters of nominal opti-

| | Existing methods | | | | | | | Algorithm 1 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Problem** | 2s | spo+ | ltr (pairwise) | ltr (listwise) | nce | dbb | pfy | $\|\cdot\|_1$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ |
| **Knapsack** | 1.78 (0.20) | 2.33 (0.18) | 14.17 (3.79) | 2.21 (0.38) | 2.17 (0.41) | 3.16 (0.19) | 2.44 (0.18) | 52.80 (22.51) | 87.71 (30.00) | 59.90 (39.26) |
| **Matching** | 12.03 (1.25) | 17.89 (1.30) | 16.68 (3.53) | 14.66 (3.22) | 15.01 (3.26) | 23.73 (1.28) | 19.22 (1.57) | 147.15 (18.27) | 135.14 (19.46) | 87.11 (14.33) |

Table 1: Computational time (average and standard deviation) per epoch in seconds of existing methods and Algorithm 1.

| | | Algorithm 1 with IOP$(\cdot,\cdot.\|\cdot\|_1)$ | | Algorithm 1 with IOP$(\cdot,\cdot.\|\cdot\|_2)$ | | Algorithm 1 with IOP$(\cdot,\cdot,\|\cdot\|_\infty)$ | |
|---|---|---|---|---|---|---|---|
| **Problem** | $M$ | Rel-Regret | CPU time (s) | Rel-Regret | CPU time (s) | Rel-Regret | CPU time (s) |
| **Knapsack** | 1 | 9.80e-02 (4.78e-02) | 59.1 (47.4) | 1.02e-01 (4.79e-02) | 65.5 (3.2) | 1.09e-01 (5.03e-02) | 38.4 (2.4) |
| | 2 | 9.86e-02 (4.72e-02) | 104.0 (86.4) | 1.03e-01 (4.64e-02) | 116.2 (5.7) | 1.08e-01 (4.76e-02) | 106.1 (72.1) |
| | 5 | 9.64e-02 (4.78e-02) | 175.2 (17.1) | 1.00e-01 (4.62e-02) | 283.6 (14.6) | 1.07e-01 (4.80e-02) | 152.8 (6.7) |
| | 10 | 9.69e-02 (4.68e-02) | 445.3 (38.1) | 1.02e-01 (4.64e-02) | 615.5 (17.5) | 1.09e-01 (4.80e-02) | 364.1 (15.7) |

Table 2: Relative regrets (average and standard deviation) on test data at the final epoch (10 epoch) of Algorithm 1 with $M \in \{1, 2, 5, 10\}$ for $\{0, 1\}$-knapsack problem, and computational time (average and standard deviation) per epoch in seconds.

mization problems using only a historical dataset containing contextual information and corresponding optimal solutions. This framework generalizes contextual optimization, where the historical dataset contains ground-truth unknown parameters that need to be predicted.

For contextual inverse linear programming problems, [Sun *et al.*, 2023] proposed a learning method referred to as maximum optimality margin, which is based on the reduced cost optimality condition [Luenberger and Ye, 2008] under the assumption that the target linear program is non-degenerate. Although any linear programming problems can be converted to non-generate problems by adding an arbitrarily small perturbation as the authors of [Sun *et al.*, 2023] have remarked, it is not trivial that the optimal basis and its complement can be obtained by the given optimal solution. Subsequently, [Mishra *et al.*, 2024] proposed a method of alternating projections onto convex sets that employs the Karush-Kuhn-Tucker (KKT) condition for linear programs. They constructed the convex set $C$ such that the given optimal solution $\boldsymbol{x}^\star$ is an optimal solution to the linear program whose coefficient vector belongs to the element of $C$. This approach enables us to apply it to general contextual linear optimization problems.

Exceptionally, [Berthet *et al.*, 2020] studied a general framework for transforming discrete optimizations into differentiable operations based on stochastic smoothing to prevent the machine learning pipeline from being broken by discrete decisions. Unlike other research on contextual inverse optimization, this framework can be applied as learning method for arbitrary contextual inverse optimization with a linear objective function. This study also shows the connection with Fenchel-Young losses studied in [Blondel, 2019].

## 6 Concluding Remarks

In this paper, we have proposed a general method for solving contextual inverse optimization problems. The proposed method is not limited to linear programming problems as studied in prior work [Sun *et al.*, 2023; Mishra *et al.*, 2024], and can be applied to any linear objective optimization problems, including integer programming problems in which inverse optimization can be employed. We have shown that the proposed algorithm corresponds to an algorithm that minimizes the loss function with respect to solution quality. Moreover, we have demonstrated the effectiveness of the proposed method for several contextual inverse optimization problems, and the results imply that the performance of the proposed method is competitive with existing predict-then-optimize methods including SPO [Elmachtoub and Grigas, 2022] even though our method does not need the ground-truth unknown parameters of the nominal optimization problem.

For future work, we would like to mention three directions. First, the proposed method requires solving inverse optimization problems exactly, which poses a challenge in terms of scalability. Therefore, we are interested in developing fast algorithms to solve inverse optimization through relaxation or other techniques. Second, it would be interesting to extend a theoretical analysis of the proposed method. Specifically, it is an important issue to theoretically guarantee that the optimal solutions prescribed by the model obtained through the proposed method match the true optimal solutions. Finally, this study considered situations in which we can access the true optimal solution, but in reality, it is common that noisy solutions are observed. Based on this, we would like to develop an algorithm for CIO when a sub-optimal solution is given.

## Acknowledgments

This work was supported by Fujitsu Limited. 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.

## References

[Ahuja and Orlin, 2000] Ravindra K Ahuja and James B Orlin. A faster algorithm for the inverse spanning tree problem. *Journal of Algorithms*, 34(1):177–193, 2000.

[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.

[Berthet *et al.*, 2020] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable pertubed 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.*, 2023] Omar Besbes, Yuri Fonseca, and Ilan Lobel. Contextual inverse optimization: Offline and online learning. *Operations Research*, August 2023.

[Blondel, 2019] Mathieu Blondel. Structured prediction with projection oracles. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12145–12156, 2019.

[Bodur *et al.*, 2022] Merve Bodur, Timothy CY Chan, and Ian Yihang Zhu. Inverse mixed integer optimization: Polyhedral insights and trust region methods. *INFORMS Journal on Computing*, 34(3):1471–1488, 2022.

[Bulut and Ralphs, 2021] Aykut Bulut and Ted K. Ralphs. On the complexity of inverse mixed integer linear optimization. *SIAM Journal on Optimization*, 31(4):3014–3043, January 2021.

[Burton and Toint, 1992] D. Burton and Ph. L. Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53(1–3):45–61, January 1992.

[Call and Holmberg, 2011] Mikael Call and Kaj Holmberg. Complexity of inverse shortest path routing. In *Network Optimization*, pages 339–353. Springer Berlin Heidelberg, 2011.

[El Balghiti *et al.*, 2019] Othman El Balghiti, Adam N. Elmachtoub, Paul Grigas, and Ambuj Tewari. Generalization bounds in the predict-then-optimize framework. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[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.

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

[Heuberger, 2004] Clemens Heuberger. Inverse combinatorial optimization: A survey on problems, methods, and results. *Journal of Combinatorial Optimization*, 8(3):329–361, September 2004.

[Hu *et al.*, 2022] Yichun Hu, Nathan Kallus, and Xiaojie Mao. Fast rates for contextual linear optimization. *Management Science*, 68(6):4236–4245, June 2022.

[Huang and Liu, 1999] Siming Huang and Zhenhong Liu. On the inverse problem of linear programming and its application to minimum weight perfect k-matching1research supported in part by a grant from hong kong university of science and technology through dag92/93.bm15.1. *European Journal of Operational Research*, 112(2):421–426, 1999.

[Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014.

[Luenberger and Ye, 2008] David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*. Springer US, 2008.

[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.

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

[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, 8 2021. Main Track.

[Roland *et al.*, 2013] Julien Roland, José Rui Figueira, and Yves De Smet. The inverse $\{0, 1\}$-knapsack problem: Theory, algorithms and computational experiments. *Discrete Optimization*, 10(2):181–192, 2013.

[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.

[Schaefer, 2009] Andrew J. Schaefer. Inverse integer programming. *Optimization Letters*, 3(4):483–489, June 2009.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, September 2008.

[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.

[Vlastelica *et al.*, 2020] Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2020.

[Wang, 2009] Lizhi Wang. Cutting plane algorithms for the inverse mixed integer linear programming problem. *Operations Research Letters*, 37(2):114–116, 2009.

[Wang, 2013] Lizhi Wang. Branch-and-bound algorithms for the partial inverse mixed integer linear programming problem. *Journal of Global Optimization*, 55(3):491–506, 2013.

[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.

[Zhang *et al.*, 1999] Jianzhong Zhang, Zhenhong Liu, and Zhongfan Ma. The inverse fractional matching problem. *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 40(4):484–496, 1999.