Counterfactual Explanations via Inverse Constraint Programming

Anton Korikov
Department of Mechanical & Industrial Engineering, University of Toronto, Canada

J. Christopher Beck
Department of Mechanical & Industrial Engineering, University of Toronto, Canada

Abstract

It is increasingly recognized that automated decision making systems cannot be black boxes: users require insight into the reasons that decisions are made. Explainable AI (XAI) has developed a number of approaches to this challenge, including the framework of counterfactual explanations where an explanation takes the form of the minimal change to the world required for a user's desired decisions to be made. Building on recent work, we show that for a user query specifying an assignment to a subset of variables, a counterfactual explanation can be found using inverse optimization. Thus, we develop inverse constraint programming (CP): to our knowledge, the first definition and treatment of inverse optimization in constraint programming. We modify a cutting plane algorithm for inverse mixed-integer programming (MIP), resulting in both pure and hybrid inverse CP algorithms. We evaluate the performance of these algorithms in generating counterfactual explanations for two combinatorial optimization problems: the 0-1 knapsack problem and single machine scheduling with release dates. Our numerical experiments show that a MIP-CP hybrid approach extended with a novel early stopping criteria can substantially out-perform a MIP approach particularly when CP is the state of the art for the underlying optimization problem.

2012 ACM Subject Classification Computing methodologies → Planning and scheduling

Keywords and phrases Explanation, Inverse Optimization, Scheduling

Digital Object Identifier 10.4230/LIPIcs.CP.2021.35

Funding This research was supported by the Natural Sciences and Engineering Research Council of Canada.

1 Introduction

As automated decision making systems continue to make high-impact decisions [9], the need to provide insight into why decisions were made has become crucial. Black box solvers are becoming less and less acceptable. In fact, recent EU legislation [19] argues that users substantially affected by an automated decision have a right to an explanation. As a consequence, there has been a surge of research aimed at explaining algorithmic decisions to users, particularly in machine learning (ML) [3]. In contrast, in constraint programming (CP) and mathematical programming, work on explaining decisions has been more limited, with the majority of explainability research focused on explaining infeasibility through the identification of minimal sets of infeasible constraints [11, 5]. We introduce new techniques based on counterfactual explanations to explain optimal decisions made by discrete constraint-based optimization systems.

For example, consider a manufacturer placing several orders for part deliveries, specifying priority values on each order. After seeing the initial schedule, the manufacturer wants an explanation. They ask "Why was the schedule not different? Why is order A not delivered in two days, and B in five days?"

We address such questions using *counterfactual explanations*, which present the questioner with the minimal change to the world required for the hypothetical situation described in their query to have occurred. In previous work [16], we formalized a counterfactual explanation as an optimization problem, which we showed to be a generalization of inverse optimization [13]. We then developed solution methods for single variable questions and explanations.

In this paper, we present the first multi-variate counterfactual explanation approach for discrete optimization systems with linear objectives, showing that we can harness inverse optimization algorithms when the questioner is interested in a partial assignment of decision variables, such as in the delivery example above. In Section 3, we formulate this kind of explanation as a *Partial Assignment Nearest Counterfactual Explanation* (PA-NCE) problem. In Section 4, we prove that the PA-NCE can be solved in two steps by first finding an optimal solution to the original decision problem with the addition of the user's assignments and then solving an inverse optimization problem.

We then turn our attention to discrete inverse optimization algorithms, used for the second stage of generating explanations but also applicable to other inverse problems. In Section 5 we develop three new inverse algorithms by modifying an existing mixed integer programming (MIP) cutting plane algorithm [26] to use CP. This results in one pure inverse CP algorithm and two MIP-CP hybrids. To our knowledge, this marks the first use of CP for inverse optimization. We also introduce a novel early stopping criteria, showing it is beneficial for inverse CP. We then show through numerical experiments in Sections 6 – 8 that a hybrid MIP-CP approach extended with our early stopping criteria can outperform alternatives when CP is state-of-the-art for the initial problem. The final sections discuss limitations and related work.

2 Background

2.1 Counterfactual Explanations

In a counterfactual explanation [25], a user would like to know why a set of facts c led to a decision x. They first ask a *contrastive question* "Why was the decision x and not $\bar{x}$?" A *counterfactual explanation* presents the user with an alternative set of facts d, typically minimally different from the initial facts c, which would have resulted in decision $\bar{x}$. The term *counterfactual* (meaning contrary to the facts) refers to the observation that neither $\bar{x}$ nor d were present in the initial (or, *factual*) context, and originates in studies of counterfactual reasoning [8]. An advantage of using counterfactual explanations for automated decision making systems is that the user is not required to understand the inner workings of the algorithm; a significant benefit for interacting with complex solvers. Further, explicitly presenting a user with a set of facts that would have led to a different outcome not only helps them understand the decision, but also empowers them to contest or act to change it [25].

As in the Explainable AI (XAI) literature [18], we refer to a counterfactual decision $\bar{x}$ specified by a user as a *foil*. In the formulation developed in our previous work [16], the user is interested in multiple counterfactual decisions, implicitly described by a *foil set*, a feasible set, X_ψ. We studied the question "Why was the decision x and not one of the decisions in X_ψ?", and defined the problem of generating explanations to such questions as the *Nearest Counterfactual Explanation (NCE)* problem.

2.2 Nearest Counterfactual Explanation (NCE)

In a standard (or *forward*) optimization problem $\langle c, f, X\rangle$, the purpose is to find values for decision vector $x \in X \subseteq \mathbb{R}^n$ given a parameter vector $c \in \mathcal{C} \subseteq \mathbb{R}^n$ which optimize an objective function $f : \mathcal{C} \times X \to \mathbb{R}$. For a minimization objective, the goal is to find an optimal x^* so that $f(c, x^*) = \min_x\{f(c, x) : x \in X\}$. In this paper, we focus on problems with linear objectives and either binary decision variables, $x \in \{0, 1\}^n$, or integer decision variables, $x \in \mathbb{N}_0^n$. If f is omitted from a forward problem $\langle c, X\rangle$, it is assumed that $f(c, x) = c^T x$.

The Nearest Counterfactual Explanation (NCE) problem [16] starts with x^*, an optimal solution to the forward problem $\langle c, f, X\rangle$ for which the user requires an explanation. Specifically, the user wants an explanation of why the solution did not also satisfy an additional set of constraints, not initially captured in X. Let these additional constraints describe a feasible set $\psi \subseteq \mathbb{R}^n$ and be called *foil constraints*, and assume that $x^* \notin \psi$. For example, if a user asks "Why was I not scheduled to receive my COVID-19 vaccine in April rather than June?", the foil constraints restrict the vaccination appointment to be in April.

The user is interested in those solutions in the intersection of X and ψ and described by the foil set: $X_\psi = X \cap \psi$. The counterfactual question asked by the user is then "Why is the solution x^* and not one of the foils in X_ψ?"

The explanation problem is to find the minimal change to the initial parameter vector c which would make a foil in X_ψ optimal for the forward problem. If $d \in \mathcal{C}$ is the modified parameter vector, where $\mathcal{C}$ can be used to express any restrictions on the feasible values of d, and $||\cdot||$ is some norm, the NCE problem $\langle c, \mathcal{C}, f, \psi, X, ||\cdot||\rangle$ [16] is

$$\min_{d\in\mathcal{C}} ||d - c|| \tag{1}$$

$$\text{s.t.} \quad \min_{x\in X_\psi} f(d, x) = \min_{x\in X} f(d, x). \tag{2}$$

Throughout this paper, we use an L_1 norm, and assume that none of the parameters in c are present in the constraints that define the feasible set X, meaning that our explanations only involve changes to objective parameters, and not constraint parameters.

We previously showed that the NCE is a generalization of *inverse optimization*, proposing that, for many problems, inverse optimization algorithms may applied [16]. However, we did not develop inverse optimization approaches, instead considering a restricted set of problems where only one parameter is allowed to change and inverse optimization is unnecessary. Our work here lifts this single variable restriction and develops a method to generate multi-variate explanations using full inverse optimization.

2.3 Inverse Optimization

While forward optimization seeks a variable assignment that satisfies a set of constraints and optimizes an objective function, inverse optimization tries to find the minimal change in the objective function such that a given feasible variable assignment is optimal. Given a forward problem $\langle c, f, X\rangle$, and a feasible solution $x^d \in X$, the inverse optimization problem is to find the minimal modification to the parameter vector c so that x^d becomes optimal [6].

As in the NCE, if $d \in \mathcal{C}$ is the modified parameter vector, then the inverse optimization problem $\langle c, \mathcal{C}, f, x^d, X, ||\cdot||_1\rangle$ is

$$\min_{d\in\mathcal{C}} ||d - c||_1 \tag{3}$$

$$\text{s.t.} \quad f(d, x^d) = \min_{x\in X} f(d, x). \tag{4}$$

The NCE is a generalization of the inverse optimization problem. In the latter, we find the values of d which make a *single* given solution optimal, while in the former, we find a d such that a *set* of solutions contains an optimal one.

Inverse Mixed Integer Programming. Much of the work in inverse optimization has focused on inverse linear optimization, where the forward problem is a linear program [13]. The work on inverse mixed integer optimization is relatively sparse with a cutting plane algorithm, *InvLP-MIP*, being the standard solution technique [26]. In Section 5, we modify this algorithm to use CP and an early stopping criteria. *InvLP-MIP* is an iterative, two-level approach where a master problem searches for a d that minimizes $||d - c||_1$ such that x^d is at least as good as all currently known forward solutions. Then, the subproblem attempts to generate a new forward solution that is better than x^d for the current d vector. If no such forward solution exists, the current d vector is optimal. Otherwise, the improving solution is added to the known forward solution set and the master is re-solved in the next iteration.

Formally, based on Wang [26], we are given a forward mixed-integer program $\langle c, X\rangle$ where $X = \{x \in \mathbb{R}_+ : Ax \leq b, x_I \in \mathbb{N}_0\}$ with $A \in \mathbb{R}^{k\times n}$, $b \in \mathbb{R}^k$, and $I \subseteq \{1, ..., n\}$, a known solution x^d, and feasible set $\mathcal{C}$ for parameter values. The L_1 norm objective of the inverse problem is linearized using $g, h \in \mathbb{R}^n_+$, such that $c - d = g - h$. Intuitively, the magnitude of the change to the parameter c_i is represented by g_i if it is negative and h_i if it is positive. Any d for which the forward problem $\langle d, X\rangle$ is unbounded can be avoided by adding the constraint $A^T y \geq d$, $y \in \mathbb{R}^k$, ensuring that d results in a feasible dual. Let $\mathcal{S}^0$ be the, initially empty, set of known feasible solutions to the forward problem. The master problem $\mathcal{MP}$ is:

$$\min_{y,g,h} \; g + h \tag{5}$$

$$\text{s.t } A^T y \geq c - g + h \tag{6}$$

$$(c - g + h)^T x^d \leq (c - g + h)^T x^0 \qquad \forall x^0 \in \mathcal{S}^0 \tag{7}$$

$$g, h \in \mathbb{R}^n, \quad y \in \mathbb{R}^k, \quad (c - g + h) \in \mathcal{C} \tag{8}$$

Constraint (7) forces the objective value of x^d to be at least as good as any known solution. The optimal solution from the master problem $d^* = (c - g^* + h^*)$ is used to construct the subproblem $\mathcal{SP}$ $\langle d^*, X\rangle$: $\min_x\{d^{*T} x : x \in X\}$. The complete algorithm is:

▶ **Definition 1** (Algorithm: *InvLP-MIP* [26]).
1. *Initialize* $\mathcal{S}^0 = \emptyset$.
2. *Solve* $\mathcal{MP}$ *to obtain* $d^* = (c - g^* + h^*)$.
3. *Solve* $\mathcal{SP}$ $\langle d^*, X\rangle$ *to get optimal solution* x^0. *If* $d^{*T} x^d \leq d^{*T} x^0$, *stop:* d^* *is optimal for the inverse problem. Otherwise, update* $\mathcal{S}^0 = \mathcal{S}^0 \cup \{x^0\}$ *and return to step 2.*

3 Problem Formulation

We now formulate a class of counterfactual explanation problems that can be solved with the help of inverse optimization. In particular, we explore the case when the user's contrastive question specifies a set of assignments to a subset of decision variables.

3.1 Example: Explaining a Delivery Schedule

Recall our delivery example where clients place a set of orders, specifying a priority level for each order. Expressed as an optimization problem, let the decision variables $x \in X \subseteq \mathbb{N}^n_0$ represent the delivery dates for each order, and the objective coefficients $c \in \mathcal{C} \in \mathbb{N}^n_0$ represent order priorities. Let the scheduling problem $\langle c, X\rangle$ be to minimize the priority weighted delivery dates, that is, to find $\min_{x\in X} c^T x$.

Upon seeing the initial optimal schedule x^*, one of the clients asks "Why was order A not scheduled to arrive in two days, and B in five days?" As we did in the NCE, we use this question to formulate a set of foil constraints, in this case giving $\psi = \{x : x_A = 2, x_B = 5\}$. Observe that these foil constraints take the form of assignments to a subset of variables.

In an NCE, the explanation takes the form of changes to objective parameters c; in this example, the order priorities. However, an explanation that could include changes to *any* order priority c_i, $i \in \{1, ..., n\}$, might not be very useful to our client. While our client is able to increase the priorities of their own orders by paying more, they have no control or knowledge of orders from other clients. Furthermore, it may be important to protect the privacy of other clients. Finally, our client may only want an explanation for a *subset* of their own orders. For these reasons, we assume that our client is primarily interested in an explanation involving changes *only* to the order priorities associated with the deliveries in their question, c_A and c_B. Such an explanation might look like: "If you increase the delivery priorities of part A from a Level 1 to a Level 3 (for \$50 extra), and part B from a Level 1 to a Level 4 (\$30 extra), the parts will be delivered on the dates you specified."

3.2 The Partial Assignment NCE

We now formulate a specific NCE problem to model cases such as our delivery example. We begin with a contrastive question from a user who is interested in a subset of m variables x_i, $i \in \mathcal{M} \subseteq \{1, ..., n\}$, $m = |\mathcal{M}|$, and desires to know why they were not assigned to specific values, $x_i^p, i \in \mathcal{M}$. We use this question to formulate the partial assignment foil constraints, giving $\psi = \{x : x_i = x_i^p \ \forall\ i \in \mathcal{M}\}$.

The user desires an explanation in terms of *only* changes to the parameters $c_i, i \in \mathcal{M}$ associated with the variables in the subset $\mathcal{M}$. The motivation for this is similar to the delivery example: explanations containing parameters associated with other users may not be useful or secure, and, additionally, this restriction allows the questioner to isolate a specific subset of variables. In an NCE, the feasible values for the modified parameters d are defined by the feasible set $\mathcal{C}$, so we can require any parameters not in $\mathcal{M}$ to retain their initial values by setting $\mathcal{C} = \{d : d_j = c_j \ \forall\ j \in \mathcal{M}^C\}$, where $\mathcal{M}^C = \{1, ..., n\} \setminus \mathcal{M}$.

▶ **Definition 2** (Partial Assignment Nearest Counterfactual Explanation (PA-NCE))**.** *The Partial Assignment NCE is an NCE $\langle f, c, \mathcal{C}, X, ||\cdot||, \psi\rangle$ in which $\psi = \{x : x_i = x_i^p \ \forall\ i \in \mathcal{M}\}$ where $\mathcal{M} \subseteq \{1, ..., n\}$, $x^p \in \mathbb{R}^m$, $m = |\mathcal{M}|$, $\mathcal{C} = \{d : d_i = c_i \ \forall\ i \in \mathcal{M}^C\}$, and $\mathcal{M}^C = \{1, ..., n\} \setminus \mathcal{M}$.*

4 Theoretical Results

Both the NCE and inverse optimization aim to find a minimally perturbed $d \in \mathcal{C}$; the former such that a set of solutions X_ψ contains an optimal solution and the latter such that a particular solution x^d is optimal. For a PA-NCE with a linear objective, we show that there exists an $x^\psi \in X_\psi$ which is optimal for *any* feasible $d \in \mathcal{C}$. In particular, we prove that such an x^ψ is given by an optimal solution to the initial forward problem plus the foil constraints, $\langle c, X_\psi\rangle$. This result implies that we can solve a PA-NCE in two steps: first, solving $\langle c, X_\psi\rangle$ to find x^ψ, and then solving the analogous inverse optimization problem with $x^d = x^\psi$.

▶ **Theorem 3.** *The Partial Assignment NCE $\langle c, \mathcal{C}, c^T x, X, ||\cdot||_1, \psi\rangle$ is equivalent to the Inverse Optimization Problem $\langle c, \mathcal{C}, c^T x, x^d, X, ||\cdot||_1\rangle$ with $x^d \in \arg\min\{c^T x : x \in X_\psi\}$.*

Proof. Observe that the NCE and Inverse Optimization problems differ only by the left-hand sides of Constraints (2) and (4). To show the two problems are equivalent, it is sufficient to show that

$$f(d, x^d) = d^T x^d = \min_x \{d^T x : x \in X_\psi\}, \quad \forall d \in \mathcal{C}. \tag{9}$$

From the optimality of x^d to the problem $\langle c, X_\psi \rangle$ and by separating the objective into the contributions from the variables $i \in \mathcal{M}$ and the variables $j \in \mathcal{M}^C$,

$$c^T x^d = \sum_{i \in \mathcal{M}} c_i x_i^d + \sum_{j \in \mathcal{M}^C} c_j x_j^d \leq \sum_{i \in \mathcal{M}} c_i x_i + \sum_{j \in \mathcal{M}^C} c_j x_j, \quad \forall x \in X_\psi. \tag{10}$$

The form of the foil constraint set ψ requires that $x_i^p = x_i^d = x_i$ for all $i \in \mathcal{M}$, so

$$\sum_{i \in \mathcal{M}^C} c_j x_j^d \leq \sum_{i \in \mathcal{M}^C} c_j x_j, \quad \forall x \in X_\psi. \tag{11}$$

Due to the constraints in $\mathcal{C}$ from Definition 2, $d_j = c_j$ for $j \in \mathcal{M}^C$, so the above inequality is valid for all values of $d \in \mathcal{C}$. Further, because the values of x_i, $i \in \mathcal{M}$, are identical in all foils, the contributions from the components in $\mathcal{M}$ are also equivalent given a value of d. Adding this contribution to both sides, we get

$$\sum_{i \in \mathcal{M}} d_i x_i^d + \sum_{i \in \mathcal{M}^C} c_j x_j^d \leq \sum_{i \in \mathcal{M}} d_i x_i + \sum_{i \in \mathcal{M}^C} c_j x_j, \quad \forall x \in X_\psi, \quad \forall d \in \mathcal{C}. \tag{12}$$

Thus $d^T x^d \leq d^T x$ for all $x \in X_\psi$ and all $d \in \mathcal{C}$, satisfying (9) and completing the proof. ◀

All our results continue to hold if the user specifies a full assignment of variables $x^p \in \mathbb{R}^n$ instead of a partial one, so that $\mathcal{M}^C = \emptyset$. In this case, the foil set is a singleton, $X_\psi = \{x^p\}$, so we can skip the first step of finding the optimal foil and proceed directly to the inverse problem with $x^d = x^p$.

5 Inverse Constraint Programming

In order to generate counterfactual explanations for constraint programs, we are interested in solving the PA-NCE for problems in which the forward problem is a constraint program. Given the connection shown above between the PA-NCE and inverse optimization, we can solve the PA-NCE by solving the forward problem using CP to find an optimal foil and then by formulating the inverse optimization problem as a constraint program. For the latter, we adopt the *InvLP-MIP* [26] approach, generalizing it to CP.

Due to the discrete nature of CP, we are primarily interested in problems where the objective coefficients are integral, $c, d \in \mathbb{N}_0^n$, which are also more difficult for MIP based inverse optimization than problems with continuous cost coefficients. Let $\mathcal{MP}_{d \in \mathbb{N}_0^n}$ be an $\mathcal{MP}$ with the constraint $g, h \in \mathbb{R}_+^n$ in (8) replaced with $g, h \in \mathbb{N}_0^n$. Such a master problem can no longer use LP, but can be formulated and solved using MIP. We call the variation of *InvLP-MIP* which uses MIP for the master problem *InvMIP-MIP*.

We can also formulate both the master and subproblems with CP. Let an $\mathcal{MP}_{d \in \mathbb{N}_0^n}$ model defined using CP be called $\mathcal{MP}_{\mathcal{CP}}$ and an $\mathcal{SP}$ model $\langle d^*, X \rangle$ defined using CP be called $\mathcal{SP}_{\mathcal{CP}}$. Examples for specific problems are provided in Section 6. We call *InvCP-CP* the algorithm which solves these models using CP and follows the cutting plane approach of *InvLP-MIP*. To our knowledge, this is the first use of CP for inverse optimization.

▶ **Definition 4** (*InvCP-CP*). *Follow steps 1-3 in InvLP-MIP, using CP to solve $\mathcal{MP}_{\mathcal{CP}}$ and $\mathcal{SP}_{\mathcal{CP}}$, instead of using LP and MIP to solve $\mathcal{MP}$ and $\mathcal{SP}$, respectively.*

Algorithm 1 *InvMIP-MIP*(ESC).

```
1  Inputs: $c, \mathcal{C}, \gamma, x^d, X$
2  Step 1: Initialize $\mathcal{S}^0 \leftarrow \emptyset$
3  Step 2: Solve $\mathcal{MP}_{d \in \mathbb{N}_0^n}$ to get optimal solution $d^*$
4  Step 3: while TRUE:
5              get next feasible solution $x^i$ to $\mathcal{SP}\ \langle d^*, X \rangle$
6              if $d^{*T} x^i \leq d^{*T} x^d$:
7                  if $x^i$ optimal:
8                      if $d^{*T} x^i == d^{*T} x^d$:
9                          Stop. $d^*$ is optimal to the inverse problem.
10                     else:
11                         Update $\mathcal{S}^0 \leftarrow \mathcal{S}^0 \cup \{x^i\}$
12                         go to step 2
13                 else:
14                     if $d^{*T} x^i < d^{*T} x^d$:
15                         if cumulative time spent in $\mathcal{SP}$ $\geq \gamma$:
16                             Update $\mathcal{S}^0 \leftarrow \mathcal{S}^0 \cup \{x^i\}$
17                             go to step 2
```

Duality. A general consideration of duality and unbounded objectives is beyond our scope, however most constraint programs involve finite domains and therefore have bounded objectives. In such cases, the dual constraints (6) in the master problem are unnecessary. If unbounded objectives could exist, it may not always be possible to formulate the dual constraints using CP; we have not yet developed a way to deal with this case. We show in Section 6 that the problems in our experiments are guaranteed to have bounded objectives.

5.1 Hybrid Approaches

In addition to a pure inverse CP algorithm, we also define hybrid inverse algorithms which use both MIP and CP. Specifically, we define *InvMIP-CP* as the algorithm that solves the master problem $\mathcal{MP}_{d \in \mathbb{N}_0^n}$ with MIP, and the subproblem $\mathcal{SP}_{\mathcal{CP}}$ with CP. Similarly, we define *InvCP-MIP* as the algorithm that solves the master problem $\mathcal{MP}_{\mathcal{CP}}$ with CP, and the subproblem $\mathcal{SP}$ with MIP.

5.2 Early Stopping Criteria

In each of the above inverse algorithms, the subproblem is solved to optimality at every iteration and its optimal solution x^0 is added to the master. However, if a feasible, but not necessarily optimal solution x^f has been found which gives a better objective value than the foil, $d^{*T} x^f < d^{*T} x^d$, then a valid cut can be generated by adding x^f to $\mathcal{S}^0$. While a better forward solution may yield a stronger cut in the master, we may wish to balance the strength and computational expense of a new cut by implementing an early stopping criteria: if a such a solution x^f is found in a given iteration, we can stop solving the $\mathcal{SP}$ after γ seconds.

We define our early stopping criteria (ESC) algorithm in Algorithm 1 using *InvMIP-MIP* as a base. It can be applied to each of the inverse algorithms discussed above. In line (5) the solver returns a feasible solution which is not worse than the previous one. The time in the $\mathcal{SP}$ (line 15) refers to the time since the most recent master solution was found.

6 Models

We test our counterfactual explanation approach for two forward problems: the 0-1 knapsack (KP) and single machine scheduling with release dates, $1|r_j|\sum w_jC_j$. The KP was selected because it is NP-complete [20], has a simple structure, and is easy to understand. While it is of practical importance, including as part of many more complex problems, its simplicity (yet NP-completeness) allowed us to develop our methods before moving to more complex hard combinatorial problems. The scheduling problem was selected because CP often performs well in scheduling, matching a potential use case for CP based explanation techniques (i.e. explainable scheduling). It is also a relatively simple (though strongly NP-Hard [17]) scheduling problem to test our methods on. Both problems have finite domains and are therefore guaranteed to have a bounded objective.

6.1 0-1 Knapsack Problem

In the 0-1 KP, we are given a set of $n \in \mathbb{N}$ items, a profit vector $c \in \mathbb{N}_0^n$, a weight vector $w \in \mathbb{N}_0^n$, and a knapsack capacity $W \in \mathbb{N}_0$. The objective of the 0-1 KP is to maximize the sum of the profits of the items that are included in the knapsack, without having the sum of the weights of those items exceed W.

CP Model. The CP model uses a packing global constraint, specifically *binPackingCapa* [12]. The first argument of this constraint is a set of bins, with each bin $\langle l, W_l\rangle$ associated with an index $l \in \mathbb{N}_0$ and a capacity $W_l \in \mathbb{N}$. The second argument is a set of items, with each item $\langle x_i, w_i\rangle$ corresponding to decision variable $x_i \in \mathbb{N}_0$ identifying which container the item is placed in and an item weight $w_i \in \mathbb{N}$. The constraint ensures that all items are placed in a container such that the sum of item weights in any container does not exceed its capacity. The CP model for 0-1 KP is

$$\max\ c^T x \tag{13}$$

$$\text{s.t. } binPackingCapa(\{\langle 0,\infty\rangle, \langle 1, W\rangle\}, \{\langle x_i, w_i\rangle | i \in \{1,...,n\}\}) \tag{14}$$

$$x \in \{0,1\}^n. \tag{15}$$

The choice of whether to place an item in container 1 or container 0 is equivalent to the decision of including or excluding that item in the knapsack, respectively.

MIP Model. Let $x \in \{0,1\}^n$ be a decision vector where $x_i = 1$ if an item is included in the knapsack and 0 otherwise. The MIP model is

$$\min_x\{c^T x : w^T x \le W, x \in \{0,1\}^n\}. \tag{16}$$

6.2 Single Machine Scheduling with Release Dates, $1|r_j|\sum w_jC_j$

In the $1|r_j|\sum w_jC_j$ problem, we are given $n \in \mathbb{N}$ jobs, with each job $i \in \{1,...,n\}$ having a processing time $p_i \in \mathbb{N}$, a weight[1] $c_i \in \mathbb{N}$, and a release date $r_i \in \mathbb{N}$. The objective is to minimize the weighted sum of completion times of all jobs given that no two jobs can be processed at the same time, no jobs can start before their release dates, and no jobs can be interrupted (no preemption).

[1] This problem is typically defined with w representing the job weights. To be consistent with our notation, we replace w with c, though we continue to refer to the problem by its typical name, $1|r_j|\sum w_jC_j$.

CP Model. We represent the jobs with a set of interval variables $\{I_i\}\ \forall\ i \in \{1, ..., n\}$, defined with the notation $intervalVar(p_i, [s_i, e_i])$, where the possible values of I_i are the intervals $\{[s_i, e_i) : s_i, e_i \in \mathbb{N}_0, s_i + p_i = e_i\}$. The model is

$$
\begin{aligned}
\min & \sum_{i=1}^{n} c_i e_i && && (17)\\
\text{s.t. } & NoOverlap(\{I_1, ..., I_n\}) && \forall\ i \in \{1, ..., n\} && (18)\\
& s_i \geq r_i && \forall\ i \in \{1, ..., n\} && (19)\\
& I_i = intervalVar(p_i, [s_i, e_i]) && \forall\ i \in \{1, ..., n\}. && (20)
\end{aligned}
$$

Constraint (18) is the $NoOverlap$ global constraint that forces jobs to be processed one at a time. Constraints (19) ensure that jobs do not start before they are released.

Time-Indexed MIP Model. As several MIP formulations exist for $1|r_j|\sum w_j C_j$, we use a time-indexed formulation due to its strong performance over a variety of instances [15]. Let $x_{i,t} \in \{0, 1\}$ be a binary decision variable which is 1 if job i is scheduled to start at time t, and 0 otherwise. With T as the time horizon, an upper bound on latest completion time of any job, the model is

$$
\begin{aligned}
\min & \sum_{i=1}^{n} \sum_{t=0}^{T-p_i} c_i(t + p_i)x_{i,t} && && (21)\\
\text{s.t. } & \sum_{t=0}^{T-p_i} x_{i,t} = 1 && \forall\ i \in \{1, ..., n\} && (22)\\
& \sum_{i=1}^{n} \sum_{s=\max(0,t-p_i+1)}^{t} x_{i,s} \leq 1 && \forall\ t = 0, 1, ..., T-1 && (23)\\
& \sum_{t=0}^{r_i-1} x_{i,t} = 0 && \forall\ i \in \{1, ..., n\} && (24)\\
& x_{i,t} \in \{0, 1\} && \forall\ i \in \{1, ..., n\},\ \forall\ t \in 1, ..., T-1. && (25)
\end{aligned}
$$

Constraints (22) force each job to start exactly once. Constraints (23) ensure no two jobs are processed at the same time. Finally, constraints (24) enforce the release dates.

7 Experimental Setup

The goal of our experiments is to test the generation of counterfactual explanations for the PA-NCE. We do this by solving an initial forward problem, generating a user query, and solving the resulting PA-NCE, focusing on the latter.

7.1 Problem Instance Generation

To generate PA-NCE instances, we create and solve a forward problem instance and then generate a set of foil constraints that form the user query.

0-1 KP Instances. Each forward problem consists of $n \in \{20, 30, 40\}$ items. For all instances, profit c_i and weight w_i are both drawn independently from the random uniform distribution $[1, R]$ with $R = 1000$. The knapsack capacity is $W = \max\{\lfloor P \sum_{i=1}^{n} w_i \rfloor, R\}$, with $P = 0.5$. Each instance was solved to produce an optimal solution, x^*.

To generate the user query, $m \in \{5, 10, 15\}$ items were randomly selected from $\{1, ..., n\}$ to create the set $\mathcal{M}$. Each user assignment $x_i^p, i \in \mathcal{M}$ was set to the opposite value of x_i^*: 0 if $x_i^* = 1$ and 1 if $x_i^* = 0$. We obtain X_ψ by adding the corresponding set of assignment constraints $x_i = x_i^p$, $\forall i \in \mathcal{M}$, to the original constraint set X for MIP and CP, respectively. Recall that in the PA-NCE, any restrictions on the feasible values of the modified parameters d are expressed through the feasible set $\mathcal{C}$. In addition to specifying that only the parameters $d_i, i \in \mathcal{M}$, can change, we also add the constraint $d \in \mathbb{N}_0^n$. We generated 20 problem instances for each combination of (n, m).

This instance generation procedure may result in an infeasible query if $\mathcal{M}$ forces the knapsack to contain items that exceed its capacity. In this case, a new random set $\mathcal{M}$ was generated until a non-empty foil set X_ψ was found.

$1|r_j|\sum w_jC_j$ Instances. Forward instances of size $n = \{5, 10, 15\}$ were generated with the random uniform distributions $p_i \in [10, 100]$, $c_i \in [1, 10]$, and $r_i \in [0, \lfloor qP \rfloor]$, where $q = 0.4$ and $P = \sum_{i=1}^n p_i$. The time horizon T was calculated as $T = \lfloor qP \rfloor + P$. We generate 20 instances for each value of (n, m), with tuple values given in Section 8.

The generation of a feasible set of foil constraints to assign the start times of a subset of jobs is non-trivial for this problem. In an optimal solution for a given complete sequence of jobs, all jobs are left-shifted subject to the release time constraints. Therefore, an arbitrarily chosen start time for a job will not form part of an optimal solution unless it happens to be equal to the job's release date or to the completion time of another job in some optimal sequence. Following the simple query generation approach used with the knapsack problem is therefore likely to result in many infeasible explanation problems.

Therefore, to generate instances more likely to have feasible explanations, we follow a different approach, although the infeasibility of some PA-NCEs remains an issue (see Section 9). We create a random permutation $(a_i)_{i \in \mathcal{M}}$ of m jobs in a randomly chosen subset $\mathcal{M}$. We then solve the original forward problem to optimality, constraining the jobs in $\mathcal{M}$ to follow the selected permutation. Finally, we select the start times in the user query to be the start times of the jobs in $\mathcal{M}$ from this solution.

Specifically, the constraints added to the forward problem were, for CP,

$$endBeforeStart(I_j, I_i) \qquad \forall\ i, j \in \mathcal{M}, a_i > a_j, \tag{26}$$

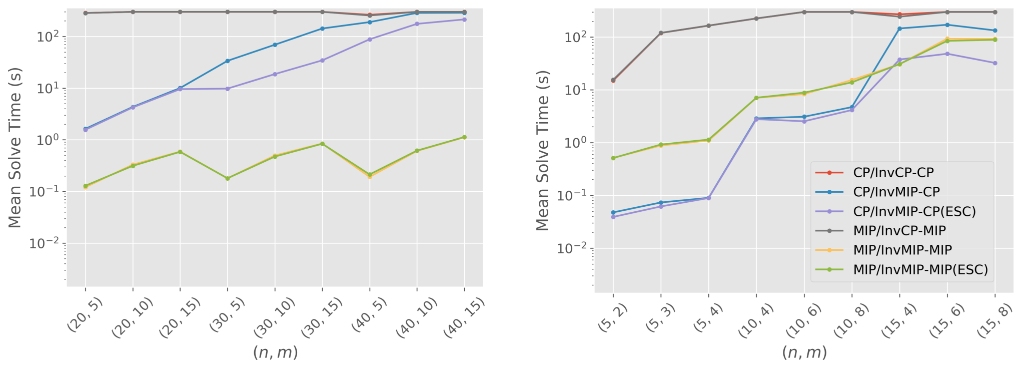
which forces the end e_j of interval variable I_j to be less than or equal to the start s_i of interval variable I_i, $e_j \leq s_i$. For MIP, the constraints added were

$$\sum_{t=0}^{T-p_j} tx_{j,t} < \sum_{t=0}^{T-p_i} tx_{i,t} \qquad \forall\ i, j \in \mathcal{M}, a_i > a_j. \tag{27}$$

Finally, we add $d \in \mathbb{N}^n = \{1, 2, ...\}^n$ as one of the constraints that define $\mathcal{C}$ in the PA-NCE, restricting the modified weights to be positive integers.

7.2 PA-NCE Solution

Having generated our PA-NCE instances, $\langle c^T x, c, \mathcal{C}, X, ||\cdot||_1, \psi \rangle$, we solve them using our two-step approach (Section 4): first finding the optimal foil x^ψ by solving $\langle c, X_\psi \rangle$, and then solving the corresponding inverse problem using x^ψ as the known solution. We test two groups of algorithms, based on whether the forward problems were solved with CP or with MIP. Notice that there are multiple forward problems involved in solving each PA-NCE instance. First, there is the optimal foil problem $\langle c, X_\psi \rangle$. Then, each iteration of the inverse



(a) 0-1 KP. **(b)** Single Machine Scheduling.

Figure 1 PA-NCE Mean Solve Times.

algorithm uses a new d to solve a subproblem $\langle d, X\rangle$. We refer to the algorithms that use CP for these forward problems as forward CP based, and the algorithms that use MIP for these forward problems as forward MIP based.

For each of these two algorithm groups, we tested three inverse algorithms. The first used CP in the master problem, while the second used MIP. The third inverse algorithm applied the ESC to the subproblem when MIP was used for the master. We name the complete two-step algorithms by first specifying the technique used to solve the optimal foil problem (CP or MIP), and then specifying the inverse algorithm. For example, we call *CP/InvMIP-CP* the algorithm that uses CP to find the optimal foil (step one) and *InvMIP-CP* to solve the inverse problem (step two). We tested six such two-step algorithms in total.

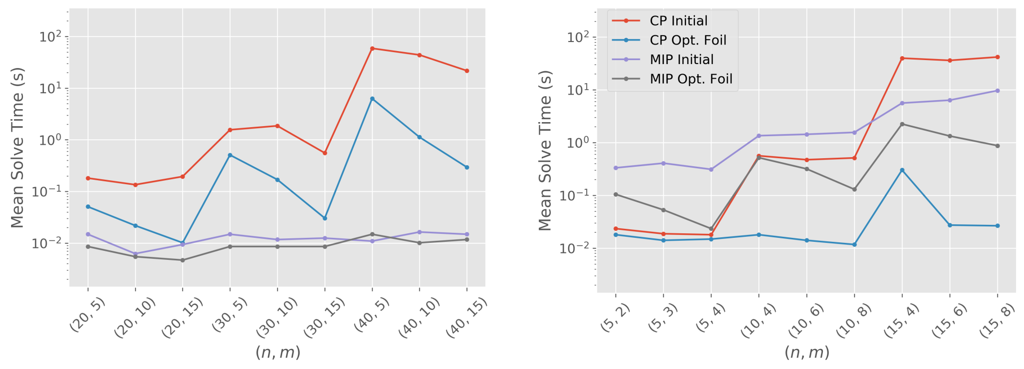
We also track the performance of CP and MIP for the initial forward problem $\langle c, X\rangle$. While this is part of instance generation and not explanation generation, it is a useful proxy for the performance of the solvers in the subproblem in the inverse algorithms. Additionally, for all inverse algorithms, we take advantage of having an initial forward solution x^* to initialize the set of known solutions as $\mathcal{S}^0 = \{x^*\}$.

7.3 Computational Details

All two-stage algorithms were run for a *global* time limit t_{max} of 300 seconds (for both stages together). If a PA-NCE instance was not solved within the global time limit, then t_{max} was recorded as the solve time. For all inverse algorithms that used the ESC, the subproblem time limit γ was set to 1 second. The MIP solver used was ILOG CPLEX V12.10 and the CP solver was ILOG CPOptimizer V12.10. Experiments were run on a single core of a 2.5 GHz Intel Core i7-6500U CPU and all reported times are CPU times.

8 Experimental Results

Overall Performance. Figure 1 shows the solution times for the two-stage algorithms for PA-NCEs and Figure 2 the solution times for the optimal foil problem and the initial forward problem – note the log-scale on the y-axes. For the 0-1 KP, the *MIP/InvMIP-MIP* and *MIP/InvMIP-MIP*(ESC) algorithms are by far the most effective, likely due to the strong MIP performance for the forward problem. As mentioned, initial forward problem



(a) 0-1 KP. **(b)** Single Machine Scheduling.

Figure 2 Mean Solve Times for Initial Forward and Optimal Foil Problems.

performance, shown in Figure 2a, is a good proxy for subproblem performance. This result is not surprising given that MIP solvers are typically very good at working with knapsack constraints.

For single-machine scheduling, the best performing algorithm overall is *CP/InvMIP-CP*(ESC), partly due to the superiority of CP over MIP on the forward problem for instances with $n \leq 10$ (Figure 2b). However, for instances with $n = 15$, the success of *CP/InvMIP-CP*(ESC) is due to the ESC modification, as demonstrated by its improvement over the unmodified *CP/InvMIP-CP* algorithm in Figure 1b.

Early Stopping Criteria. For both problems, the early stopping criteria was clearly beneficial for the forward CP based algorithm, *InvMIP-CP*, when n was sufficiently large. It had no effect for smaller problems, because when a subproblem is solved to optimality within γ seconds, the ESC is never met. Interestingly, the ESC did *not* produce improvements the forward MIP based algorithm, *InvMIP-MIP*. Recall that the motivation behind the ESC was to avoid solving the $\mathcal{SP}$ to optimality at each iteration. We speculate that the ESC improves *InvMIP-CP* because it prevents CP from spending an excessive amount of time *proving* that the subproblem solution is optimal. In contrast, we speculate that much less time is spent proving subproblem optimality in *InvMIP-MIP*.

CP for Master Problem. When CP is used to solve the master problem, performance is very poor: *CP/InvCP-CP* and *MIP/InvCP-MIP* both reached the time limit on most instances (Figure 2a and 2b). The simple linear structure of the master problem lends itself particularly well to MIP solving. However, we anticipate that CP may be useful in the master problem for more complex explanation problems, for example to express more complicated constraints on $\mathcal{C}$, providing more control over how the objective parameters are allowed to change.

Instance Breakdown. Tables 1 and 2 provide more detailed data on the performance of the two-stage algorithms in terms of the number of instances solved optimally, proved infeasible, and timed-out. There were a large number of infeasible PA-NCEs for larger scheduling instances even with our approach for generating feasible foils. Even though these instances have feasible foils by construction, there is no guarantee that a cost vector exists that makes

Table 1 Number of 0-1 KP PA-NCE Solutions Optimal (O) and Timeout (T). Algorithm names are split into the first (optimal foil) and second (inverse) stage components.

Foil Alg		CP						MIP					
Inv Alg		CP-CP		MIP-CP		MIP-CP (ESC)		CP-MIP		MIP-MIP		MIP-MIP (ESC)	
n	**m**	O	T	O	T	O	T	O	T	O	T	O	T
20	5	1	19	20	0	20	0	1	19	20	0	20	0
	10	0	20	20	0	20	0	0	20	20	0	20	0
	15	0	20	20	0	20	0	0	20	20	0	20	0
30	5	0	20	20	0	20	0	0	20	20	0	20	0
	10	0	20	19	1	20	0	0	20	20	0	20	0
	15	0	20	14	6	20	0	0	20	20	0	20	0
40	5	3	17	11	9	18	2	3	17	20	0	20	0
	10	0	20	1	19	15	5	0	20	20	0	20	0
	15	0	20	2	18	10	10	0	20	20	0	20	0

Table 2 Number of $1|r_j|\sum w_j C_j$ PA-NCE Solutions: Optimal (O), Infeasible (I), Timeout (T). Algorithm names are split into the first (optimal foil) and second (inverse) stage components.

Foil Alg		CP									MIP								
Inv Alg		CP-CP			MIP-CP			MIP-CP (ESC)			CP-MIP			MIP-MIP			MIP-MIP (ESC)		
n	**m**	O	I	T	O	I	T	O	I	T	O	I	T	O	I	T	O	I	T
5	2	19	0	1	19	1	0	19	1	0	19	0	1	19	1	0	19	1	0
	3	12	0	8	18	2	0	18	2	0	12	0	8	18	2	0	18	2	0
	4	9	0	11	15	5	0	15	5	0	9	0	11	15	5	0	15	5	0
10	4	5	0	15	12	8	0	12	8	0	5	0	15	12	8	0	12	8	0
	6	0	0	20	5	15	0	5	15	0	0	0	20	5	15	0	5	15	0
	8	0	0	20	4	16	0	4	16	0	0	0	20	4	16	0	4	16	0
15	4	3	0	17	6	8	6	7	12	1	4	0	16	8	12	0	8	12	0
	6	0	0	20	0	11	9	3	15	2	0	0	20	4	15	1	5	14	1
	8	0	0	20	1	12	7	1	18	1	0	0	20	2	16	2	2	17	1

the foil optimal. This issue is discussed further in Section 9. In contrast, no instances were infeasible for the inverse knapsack problems as any inverse 0-1 KP has a feasible solution in which any items which are not included in the knapsack by x^d have their profits set to 0.

Optimal Foil Problem. Results for finding the optimal foil are included in Figures 2a and 2b. Similar to most initial problems, MIP performed better for KP while CP performed better for scheduling.

9 Discussion and Limitations

This paper develops methods to find nearest counterfactual explanations for a class of optimization problems and user queries and introduces inverse constraint programming as part of the solution methodology. Here we address the limitations of our approaches.

Counterfactual explanations are independent of the algorithmic decision making process as they identify *changes* in the problem that would result in the user's specifications being met. However, they require that the changeable parameters are meaningful to the user. If this is not the case, then the user requires a higher level explanation of why those parameters are used and how their factual values were derived: questions that touch on ethics, fairness, and the broader human system surrounding the optimization problem [16].

Our approach cannot generate nearest counterfactual explanations for foil sets described by more expressive constraints, as defined in the general NCE. The core challenge is that the constraints admit a set of solutions and the NCE seeks a cost vector such that *any one* of those solutions is optimal. As far as we are aware, there exist no inverse optimization approaches to such generalizations. We were able to solve the PA-NCE because Theorem 3 guarantees the existence of an optimal foil and so standard inverse optimization can be used. As the NCE is a bi-level optimization problem, one direction for future work on the more general problem is to investigate techniques from discrete bi-level optimization [22].

Similarly, the NCE definition and our approach are limited to counterfactual values for parameters in the objective function. Clearly, a user may want explanations in terms of changes to constraints. The challenge is that such changes modify the feasible set of the forward problem, complicating the inverse optimization formulation. We are not aware of any general inverse combinatorial optimization approaches that can handle changes to constraint parameters, though formulations have been investigated for specific inverse problems [27]. One direction for work on such problems may be to combine the approach here with existing work in CP on explaining infeasibilities (Section 10). In continuous optimization, inverse methods allowing changes to constraint parameters exist for linear programs [4].

In our formulation, a counterfactual query is an assignment of a subset of variables. However, we required that an explanation consist only of changes to the parameters corresponding to the variables in the user query. While we argued above that such a restriction is often useful due to parameter relevance and privacy, without it Theorem 3 does not hold and our solution approach does not work. The challenge, as above, is that without this restriction, the NCE requires finding a parameter vector such that any one of a set of solutions is optimal.

Finally, as described in Section 7.1 and shown in Table 2, the restrictions of PA-NCE may make it difficult to generate queries with a feasible explanation. From a user's perspective, just as with forward optimization, a result that says that there is no world in which the user's decisions are possible is not particularly helpful. As far as we are aware this issue has not been addressed in the broader literature of counterfactual explanations. However, the generalization noted above of allowing constraint coefficients to change may be worth pursuing for this challenge as well.

In spite of these limitations, the work in this paper substantially extends the scope of counterfactual explanations for optimization-based decisions from a counterfactual query on a single variable to queries on some or all variables, albeit with a restricted query form. Furthermore, for the first time we have defined inverse constraint programming and solved such problems through an adaptation and extension of work in inverse integer programming.

10 Related Work

We build directly on our previous work [16], which formulates the NCE and proposes the connection to inverse combinatorial optimization. However, that work does not develop an inverse optimization based solution approach, relying on the restriction that the query must be a linear constraint over a single variable. This restriction allows NCEs with binary decision variables to be solved in closed form given the solution to a modified problem, and with binary search over a series of modified problems when the decision variables are integers.

In ML, there has been a significant amount of research on counterfactual explanations for classifiers [24], with highly influential work by Wachter *et al.* [25]. Although still nascent, the work has already developed more advanced concepts such as diverse counterfactual explanations [21] which could be extended to counterfactual explanations for optimization.

In AI Planning, a counterfactual explanation approach has been adopted in a number of contexts, for instance to enumerate the shared properties of all possible counterfactual plans [7] and to identify the differences between counterfactual plans and factual plans [10]. Further, Brandao and Maggazzeni [2] recently used inverse optimization to generate explanations for path planning, for which there exists a polynomial inverse algorithm.

In CP, the majority of work on explanations has focused on explaining infeasibility with work on optimality being sparse [11]. To our knowledge there have been no attempts to explain optimality using counterfactual explanations. For constraint satisfaction problems which are solveable with inference only (no search), Sqalli and Freuder [23] generated explanations by tracing the inference steps, and observed that, for the logic puzzles used in their experiments, these explanations were very similar to those generated by humans. Subsequent research has built on this approach [1] while acknowledging the limitation that, for problems that also require search, tracing solver steps does not currently provide understandable explanations.

Explanation of infeasibility in CP has largely dealt with finding minimal sets of unsatisfiable constraints [14], and a parallel literature exists in mathematical programming [5]. We have previously proposed [16] that such explanations could also be viewed as counterfactuals (i.e., a set of constraints that must change) but this connection has not been developed. Freuder [11] provides a recent discussion and overview of explainability in CP.

11 Conclusion

Counterfactual explanations answer a user query asking why, given a set of facts, an initial decision did not satisfy some desired characteristics. The explanation is a hypothetical set of facts that would have satisfied the user's characteristics. Because they do not require the user to understand the inner workings of increasingly complex and uninterpretable algorithms, counterfactual explanations are drawing considerable research attention in AI. We build on recent work on counterfactual explanations for discrete optimization by introducing multi-variate explanation problems and solving them with the help of inverse optimization.

When a user is interested in an alternative, partial set of variable assignments, we show how to generate an explanation in terms of changes to the objective parameters associated with those variables. A counterfactual explanation can be found by first solving the original problem with the addition of the user's partial assignment constraints, and then solving a corresponding inverse optimization problem. We solve the inverse problem with constraint programming through a modification of an existing MIP cutting plane algorithm to develop both pure and hybrid inverse constraint programming algorithms. In addition, we develop a novel early stopping criteria that significantly improves inverse CP on larger problem instances. Finally, through numerical experiments we demonstrate our solution approaches for the 0-1 knapsack problem and a single machine scheduling problem, and show that a hybrid MIP-CP approach can achieve superior performance compared to a pure MIP approach, particularly when CP is state of the art for the underlying optimization problem.

References

1 B. Bogaerts, E. Gamba, J. Claes, and T. Guns. Step-wise explanations of constraint satisfaction problems. In *24th European Conference on Artificial Intelligence (ECAI)*, 2020.

2 M. Brandao and D. Magazzeni. Explaining plans at scale: scalable path planning explanations in navigation meshes using inverse optimization. In *Workshop on XAI, IJCAI-PRICAI*, 2020.

3 J. Burrell. How the machine ‘thinks’: Understanding opacity in machine learning algorithms. *Big Data & Society*, 3(1), 2016.

4 T. C. Y. Chan and N. Kaw. Inverse optimization for the recovery of constraint parameters. *European Journal of Operational Research*, 282(2):415–427, 2020.

5 J. W. Chinneck. *Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods*. Number 118 in International Series in Operations Research and Management Sciences. Springer, 2008.

6 M. Demange and J. Monnot. An introduction to inverse combinatorial problems. *Paradigms of Combinatorial Optimization: Problems and New Approaches*, pages 547–586, 2014.

7 R. Eiffer, M. Cashmore, J. Hoffmann, D. Magazzeni, and M. Steinmetz. A New Approach to Plan-Space Explanation: Analyzing Plan-Property Dependencies in Oversubscription Planning. In *AAAI*. AAAI, 2020.

8 K. Epstude and N. J. Roese. The functional theory of counterfactual thinking. *Personality and social psychology review*, 12(2):168–192, 2008.

9 V. Eubanks. *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin’s Press, 2018.

10 M. Fox, D. Long, and D. Magazzeni. Explainable Planning. *CoRR*, abs/1709.10256, 2017. `arXiv:1709.10256`.

11 E. Freuder. Explaining ourselves: human-aware constraint reasoning. In *AAAI*, 2017.

12 Global Constraint Catalogue. `https://sofdem.github.io/gccat/`. Accessed: 2021-08-5.

13 C. Heuberger. Inverse combinatorial optimization: A survey on problems, methods, and results. *Journal of combinatorial optimization*, 8(3):329–361, 2004.

14 U. Junker. Preferred explanations and relaxations for over-constrained problems. In *AAAI*, 2004.

15 A. B Keha, K. Khowala, and J. W. Fowler. Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering*, 56(1):357–367, 2009.

16 A. Korikov, A. Shleyfman, and C. Beck. Counterfactual explanations for optimization-based decisions in the context of the GDPR. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2021.

17 J. K. Lenstra, A. R. Kan, and P. Brucker. Complexity of machine scheduling problems. In *Annals of discrete mathematics*, volume 1, pages 343–362. Elsevier, 1977.

18 T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

19 Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 2016.

20 D. Pisinger, H. Kellerer, and U. Pferschy. Knapsack problems. *Handbook of Combinatorial Optimization*, page 299, 2013.

21 C. Russell. Efficient search for diverse coherent explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 20–28, 2019.

22 A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

23 M. H. Sqalli and E. C. Freuder. Inference-based constraint satisfaction supports explanation. In *AAAI/IAAI, Vol. 1*, pages 318–325, 1996.

24 S. Verma, J. Dickerson, and K. Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint*, 2020. `arXiv:2010.10596`.

25 S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, 31:841, 2017.

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

27 C. Yang, J. Zhang, and Z. Ma. Inverse maximum flow and minimum cut problems. *Optimization*, 40(2):147–170, 1997.