Objective-Based Counterfactual Explanations for Linear Discrete Optimization

Anton Korikov[0009−0003−4487−9504] and J. Christopher Beck[0000−0002−4656−8908]

Department of Mechanical & Industrial Engineering, University of Toronto, Toronto, Canada
{korikov,jcb}@mie.utoronto.ca

Abstract. Given a user who asks why an algorithmic decision did not satisfy some conditions, a counterfactual explanation takes the form of a minimally perturbed input that would have led to a decision satisfying the user's conditions. Building on recent work, this paper develops techniques to generate counterfactual explanations for linear discrete constrained optimization problems. These explanations take the form of a minimally perturbed objective vector that induces an optimal solution satisfying the newly stated user constraints. Drawing inspiration from the inverse combinatorial optimization literature, we introduce a novel non-convex quadratic programming algorithm to generate such explanations. Furthermore, we develop conditions for the existence of an explanation, addressing a limitation of past approaches. Finally, we discuss several future directions for explanations in discrete optimization such as actionable and sparse explanations.

1 Introduction

As the use of automated decision-making systems has increased, research has turned toward the question of providing explanations for the decisions that are made [12, 13]. Such explanations enhance people's ability to interact with automated systems, improving performance of deployed systems [8] and facilitating better human oversight [7]. While much of explainability research has focused on deep learning (e.g., [13]), explainability is also important for model-based decision making systems, such as those studied in Constraint Programming, Operations Research, and Artificial Intelligence. Unlike deep learning models, declarative models are often decomposable into human-understandable symbols (e.g., costs, weights, priorities, etc.), yet decision algorithms are typically too complex or involve too many steps for a human to easily follow, making it difficult for people to contemplate relationships between modelling choices and algorithmic decisions [12]. Working in the context of AI planning, Smith [21] therefore proposed that explainability techniques are needed to support human reasoning about the effects of modeling choices on algorithmic decisions. Explainable AI Planning (XAIP) has since emerged as a rapidly growing research area [3], successful both in developing techniques specialized to AI planning [10,

4] as well as integrating broader explainable AI (XAI) research (e.g., contrastive explanations [17]).

In the field of optimization, an explainability literature similar in scope to XAIP has yet to emerge. Most explainability research in optimization has focused on explaining why a problem instance is infeasible, typically by identifying minimal sets of conflicting constraints [20], and there is little work on explaining feasible or optimal decisions [12]. Furthermore, integration between explainability research in optimization and XAI research at large has been limited.

Aiming to address this gap, our recent work [14, 15] applied the XAI technique of counterfactual explanations [24] to optimal solutions of discrete optimization problems. Given an explainee[1] asking why an algorithmic decision was not different in a specific way, a counterfactual explanation presents minimally perturbed inputs to the algorithm that would have led to the decision being different in the way specified. In the framework introduced in our past work [14], an explainee asks why an optimal solution to a discrete optimization problem did not satisfy a previously unstated set of constraints. An explanation then takes the form of a minimally perturbed objective vector, so that, with the perturbed objective vector replacing the original one, an optimal solution would satisfy both the initial and new constraints.

Example 1. (Production Scheduling). A manager at a factory examines a monthly production schedule computed by an optimization system, the objective of which is determined by job priority levels and completion times. She asks: "Why were jobs 1 and 2 not completed in less than one week?". The explanation shows her the minimal change to the job priorities in the upcoming month so that jobs 1 and 2 would be completed in under one week in an optimal schedule. After receiving the explanation, the manager can either choose to keep the initial job priorities and accept the initial schedule or to produce a new schedule with the modified priority levels.

The task of generating such objective-based counterfactual explanations is a valuable research direction for several reasons. Objectives are the result of modeling choices, however, complex optimization algorithms make it difficult for a person to understand how a particular modeling choice leads to certain solution features [21]. For instance, in Example 1, the manager desires more information on how the job priority values (a modeling choice) impact whether jobs 1 and 2 are completed within a week (a solution feature). While such information can take many forms (see Section 7), one of the standard forms studied in XAI is a minimal change to a set of problem inputs that induces the solution feature in question [24]. The explanation in Example 1 takes this form. The goal of providing such information is to facilitate counterfactual reasoning, a fundamental human reasoning strategy [11], about an algorithm's inputs and outputs. Furthermore, when explainees are shown how decisions can be changed, they are empowered to contest or act to change decisions they believe are wrong [24].

[1] A person receiving an explanation.

Finally, research on counterfactual explanations in machine learning is developing rapidly, and many opportunities for cross-pollination exist between this literature and explainable optimization (see Section 7).

Furthermore, an *objective-based* explanation formulation allows connections to be made between explanation and inverse combinatorial optimization [6]. Given a possibly suboptimal solution, inverse optimization aims to find the minimal change to an optimization problems objectives such that the given solution becomes optimal. The use of inverse optimization as a methodological basis for objective-based explanation is discussed further in Sections 2 and 4.

However, while our past work [14, 15] defined an explanation framework for discrete optimization, the algorithms we used could only compute explanations for limited forms of questions. Specifically, an explainee could only ask why a subset of variables did not satisfy a partial assignment, or why a single variable did not satisfy a linear constraint (see Section 2.2). Their experiments also showed that for these restricted questions, no explanation existed for many natural problem instances.

Addressing these limitations, we introduce a novel, non-convex quadratic programming algorithm which can compute explanations for any linear discrete optimization problem when the user's question can be represented by linear or quadratic constraints. This new algorithm is inspired by a well-known algorithm in inverse combinatorial optimization [25], and capitalizes on recent advances in commercial solvers. Numerical simulations are performed to evaluate the new algorithm and demonstrate the explanation process for two combinatorial optimization problems. Additionally, we establish conditions for the existence of an explanation and apply them to design experimental problem instances. Finally, several future directions are identified, such as actionable and sparse explanations, and the limitations of the methods in this paper are discussed.

2 Background

2.1 Counterfactual Explanations

Given an algorithm which computes output p from input k, a counterfactual explanation responds to a *contrastive question* of the form: "Why p, and not some other output $q \in Q$?" [24]. Here, Q, called a *foil set*, is a set of alternative outputs, and each alternative output $q \in Q$ is called a *foil*. Given such a question, a counterfactual explanation shows the explainee an alternative input l which would have led to the output being in Q, with l typically selected such that it minimally perturbs k.[2]

2.2 Nearest Counterfactual Explanations

Given an objective vector $c \in \mathcal{D} \subseteq \mathbb{R}^z$, the purpose of a standard (or *forward*) optimization problem $\mathcal{FW}\langle c, f, X\rangle$ is to find values for a decision vector $x \in$

[2] *Counterfactual* means "contrary to the facts". The alternative input l and outputs $q \in Q$ are contrary to the initial input k and output p, respectively.

$X \subseteq \mathbb{R}^n$ which optimize an objective function $f : \mathcal{D} \times X \to \mathbb{R}$. In a minimization problem, the goal is to find an optimal x^* so that $f(c, x^*) = \min_x\{f(c, x) : x \in X\}$. If no optimization direction is specified, we assume minimization.

A counterfactual explanation process for an optimal solution x^* to a forward problem $\mathcal{FW}\langle c, f, X\rangle$ can be modeled using a Nearest Counterfactual Explanation (NCE) problem [14]. The explainee must first describe a set of alternative solutions $X_\psi \subset X$, and ask the contrastive question "Why x^* and not a solution $x^\psi \in X_\psi$?". As per Section 2.1, X_ψ is called the foil set and each solution $x^\psi \in X_\psi$ is called a foil. To define the foil set, the explainee must specify an additional set of constraints describing a feasible set $\psi \subset \mathbb{R}^n$, with $x^* \notin \psi$. These additional constraints are called *foil constraints*, and the foil set is defined as $X_\psi = X \cap \psi$.

Example 2. (Production Schedule - Contrastive Question). In Example 1, assume that a schedule for n jobs is generated by solving a $\mathcal{FW}\langle c, f, X\rangle$ with a decision vector $x \in X \subseteq \mathbb{N}_0^n$, where x_i represents the number of days before job i is completed. In the objective vector $c \in \mathcal{D} \subseteq \mathbb{N}_0^n$, c_i represents the priority of job i, and the objective is to minimize $f = c \cdot x$,[3] the sum of priority weighted completion times. Given $n = 4$ and $c = [4, 4, 3, 3]$, an optimal solution is $x^* = [8, 13, 1, 3]$, prompting the manager to ask why jobs 1 and 2 were not completed in under one week. In this case, the foil constraints are $x_1 \leq 7$ and $x_2 \leq 7$, and the contrastive question is "Why x^* and not an $x^\psi \in X_\psi$?", where $X_\psi = X \cap \psi$ and $\psi = \{x \in \mathbb{N}_0^n : x_1 \leq 7, x_2 \leq 7\}$.

The NCE addresses this type of question by searching for a counterfactual objective vector $d \in \mathcal{D} \subseteq \mathbb{R}^z$ that would lead to one of the foils $x^\psi \in X_\psi$ being optimal to the modified problem $\mathcal{FW}\langle d, f, X\rangle$, such that d is minimally perturbed from the initial objective vector c. This perturbation is measured by some norm $||\cdot||$, assumed to be L_1 if unspecified. If such a d is found, an explanation is: "A solution $x^\psi \in X_\psi$ would have been optimal if the objective vector had been d instead of c." Formally, assuming a minimization forward objective, the $\mathcal{NCE}\langle c, \mathcal{D}, f, \psi, x^*, X, ||\cdot||\rangle$ is

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

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

If the optimization direction of the underlying forward problem is maximization, the minimization terms in constraint (2) are replaced with maximization terms.

Example 3. (Production Scheduling - Explanation). Assume the manager from Examples 1 & 2 is interested in explanations where job priorities can be adjusted to integers between 1 and 5, giving $\mathcal{D} = \{d \in \mathbb{N}^4 : 1 \leq d_i \leq 5,\ \forall i \in \{1, ..., 4\}\}$. If an optimal solution to the resulting $\mathcal{NCE}\langle c, \mathcal{D}, f, \psi, x^*, X\rangle$ is $d^* = [5, 5, 2, 2]$, the explanation is "Jobs 1 and 2 would have finished in under a week if their

[3] Where clear from context, $c \cdot x$ is used as shorthand for $c^T x$.

priorities were increased to the maximum level (5) and the priorities of the other two jobs were both one level lower (2)".

NCEs provide a general way to model objective-based explanations of optimal forward solutions. However, previous solution methods could only solve NCEs for two restricted question types: questions about a single variable [14] and questions that ask why x^* did not satisfy a partial assignment [15]. Both methods, in addition, restrict some components of d from being perturbed. In fact, neither of these methods can solve the NCE described by Examples 1 - 3. Also, other than observing that it is necessary for $X_\psi \neq \emptyset$ in a feasible NCE, we previously did not formally study NCE feasibility conditions.

The main contribution of this paper is a novel quadratic programming algorithm which can solve any NCE where the forward problem is a discrete linear optimization problem, the foil constraints are linear or quadratic, and the norm is L_1. This new algorithm is inspired by inverse combinatorial optimization [25].

2.3 Inverse Combinatorial Optimization

Given a forward problem $\mathcal{FW}\langle c, f, X\rangle$ and a feasible target solution $x^d \in X$, the inverse optimization problem is to find a new objective vector $d \in \mathcal{D} \subseteq \mathbb{R}^z$, minimally perturbed from c, so that x^d becomes optimal. Given some norm $||\cdot||$, the inverse optimization problem $\mathcal{INV}\langle c, \mathcal{D}, f, x^d, X, ||\cdot||\rangle$ [6] is

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

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

The inverse optimization problem can be interpreted as a special case of the NCE where the foil set is the singleton $X_\psi = \{x^d\}$. Though most inverse optimization algorithms have focused on continuous optimization [5], methods also exist for discrete optimization, with the standard technique being the *InvMILP* algorithm for inverse Mixed Integer Linear Programming (MILP) [25]. Our new algorithm is inspired by *InvMILP*.

A $\mathcal{MILP}\langle c, X\rangle$ is a forward problem $\mathcal{FW}\langle c, f, X\rangle$ with $c \in \mathcal{D} \subseteq \mathbb{R}^n$, $f = c \cdot x$, and $X = \{x \in \mathbb{R}^n_+ : Ax \leq b, x_I \in \mathbb{N}_0\}$ with $A \in \mathbb{R}^{v \times n}$, $b \in \mathbb{R}^v$, and $I \subseteq \{1, ..., n\}$. An inverse MILP, $\mathcal{INV}_{\mathcal{MILP}}\langle c, \mathcal{D}, x^d, X\rangle$, is an inverse problem where the forward problem is a $\mathcal{MILP}\langle c, X\rangle$, $\mathcal{D} \subseteq \mathbb{R}^n$, and the norm is L_1.

To solve such inverse MILPs, *InvMILP* (Algorithm 1) uses an iterative, two-level approach where a master problem $\mathcal{MP}_{\mathcal{INV}}$ (5) - (8) is initialized with a set $\mathcal{S}^0 \subseteq X$ of known extreme points of $conv(X)$, the convex hull of X. $\mathcal{MP}_{\mathcal{INV}}$ then searches for a d, minimizing $||d - c||_1$, such that x^d is at least as good of a solution to $\mathcal{MILP}\langle d, X\rangle$ as any any point in $\mathcal{S}^0$. If such a d is found, a subproblem $\mathcal{MILP}\langle d, X\rangle$ is solved to optimality, returning an extreme point x^0. If x^0 gives a better objective value for $d \cdot x$ than x^d, then x^0 is added to $\mathcal{S}^0$ and the algorithm proceeds to the next iteration of $\mathcal{MP}_{\mathcal{INV}}$. *InvMILP* continues iterating either until the subproblem finds that x^d is optimal to $\mathcal{MILP}\langle d, X\rangle$,

Algorithm 1: *InvMILP* [25].

```
Inputs: INV_{MILP}⟨c, D, x^d, X⟩.
Output: d*.
Step 1: Initialize S^0 ← ∅.
Step 2: Solve MP_{INV}⟨c, D, x^d, X, S^0⟩.
  If infeasible, return INFEAS.
  Otherwise, get d^i = (c − g^i + h^i).
Step 3: Solve MILP⟨d^i, X⟩ to get x^0.
  If d^{i,T} x^d ≤ d^{i,T} x^0, stop. Return d^i = d*.
  Otherwise, update S^0 = S^0 ∪ {x^0} and return to Step 2.
```

in which case d is an optimal solution to the inverse problem, or until the master problem is found infeasible, which will occur if the inverse problem is infeasible.

To formulate the master problem $\mathcal{MP}_{\mathcal{INV}}$ (5) - (8), the objective $||d - c||_1$ is first linearized using $g, h \in \mathbb{R}^n_+$, such that $c - d = g - h$: the magnitude of the change to parameter c_j is represented by g_j if the change is negative and h_j if it is positive. Constraints (6) force x^d to be at least as good a solution to $\mathcal{MILP}\langle d, X\rangle$ as any point in $\mathcal{S}^0$. Finally, to avoid any d for which the forward problem is unbounded, Wang introduces the decision variable $u \in \mathbb{R}^v_+$ and adds the constraint $A^T u \geq d$ (7), ensuring that d results in a feasible dual problem. Thus, $\mathcal{MP}_{\mathcal{INV}}\langle c, \mathcal{D}, x^d, X, \mathcal{S}^0\rangle$, a linear program, is given by

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

$$\text{s.t } (c - g + h)^T x^d \leq (c - g + h)^T x^0 \qquad \forall x^0 \in \mathcal{S}^0 \tag{6}$$

$$A^T u \geq c - g + h \tag{7}$$

$$(c - g + h) \in \mathcal{D},\; g \in \mathbb{R}^n_+,\; h \in \mathbb{R}^n_+,\; u \in \mathbb{R}^v_+. \tag{8}$$

The complete *InvMILP* algorithm, which has been proven to terminate finitely [25], is given by Algorithm 1.

Noticeably absent from the discrete inverse optimization literature are algorithms capable of handling changes to constraint parameters. Such constraint parameter changes would add a degree of difficulty to the inverse optimization problem since they could induce the existence of multiple alternative feasible sets. The absence of such constraint-based inverse optimization methods is the reason that we choose to study objective-based explanations, as opposed to explanations based on changes to both objectives and constraints.

3 Problem Definition

This paper focuses on NCEs where the forward problem is a $\mathcal{MILP}\langle c, X\rangle$, the foil constraints defining X_ψ are linear or quadratic, and $||\cdot||$ is L_1. Such an NCE is denoted $\mathcal{NCE}_{\mathcal{MILP}}\langle c, \mathcal{D}, \psi, x^*, X\rangle$, and Examples 1 - 3 are examples of an $\mathcal{NCE}_{\mathcal{MILP}}$, given that the forward scheduling problem is a MILP.

Definition 1. *($\mathcal{NCE}_{\mathcal{MILP}}$). An $\mathcal{NCE}_{\mathcal{MILP}}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ is an $\mathcal{NCE}\langle c, \mathcal{D}, f, \psi, x^*, X, ||\cdot||\rangle$ where the forward problem is a $\mathcal{MILP}\langle c, X\rangle$, ψ is defined by linear or quadratic constraints, $f = c^T x$, and $||\cdot||$ is L_1. A feasible $\mathcal{NCE}_{\mathcal{MILP}}$ solution, d, must not result in an unbounded $\mathcal{MILP}\langle d, X\rangle$.*

3.1 Existence of an Explanation

We now introduce conditions for the existence of a feasible, non-trivial solution to an $\mathcal{NCE}_{\mathcal{MILP}}$, defined as any $d \in \mathcal{D}$ feasible to (1)-(2) such that $d \neq \mathbf{0}$. While our past work [15] showed that NCE infeasibility can be an issue, no NCE feasibility conditions have been established other than the observation that it is necessary for $X_\psi \neq \emptyset$. We formalize a simple, necessary condition (Proposition 1) as well as a sufficient condition (Theorem 1) for $\mathcal{NCE}_{\mathcal{MILP}}$ feasibility, and Section 5 uses these conditions to design experimental instances.

Proposition 1 *For an $\mathcal{NCE}_{\mathcal{MILP}}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ to have a non-trivial feasible solution, X_ψ cannot lie entirely in the interior region of $conv(X)$.*

Proof. If X_ψ lies entirely in the interior region of $conv(X)$, X_ψ cannot contain an optimal solution to $\mathcal{MILP}\langle d, X\rangle$ for any $d \neq \mathbf{0}$. □

Next, we present a sufficient feasibility condition for an $\mathcal{NCE}_{\mathcal{MILP}}$. Intuitively, assuming minimization, setting a single variable x_j to its minimal value in the feasible set will lead to an optimal $\mathcal{MILP}\langle d, X\rangle$ solution if d_j is greater than zero while all other components of d are zero. Formally, for all $i \in \{1, ..., n\}$, let $x_{i,\min} = \min_x\{x_i : x \in X \subseteq \mathbb{R}^n_+\}$ and $x_{i,\max} = \max_x\{x_i : x \in X \subseteq \mathbb{R}^n_+\}$. Also, let $\mathcal{D}^{f,+}_j = \{d \in \mathbb{R}^n_+ : 0 < d_j \leq d^{UB}_j, d_i = 0 \ \forall \ i \neq j\}$, where $d^{UB}_j = \max_d\{d_j : d \in \mathcal{D}\}$ and $j \in \{1, ..., n\}$.

Theorem 1 *An $\mathcal{NCE}_{\mathcal{MILP}}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ has a non-trivial feasible solution if all following conditions hold:*

1. – *If the forward optimization direction is minimization, $\exists\ \tilde{x}^\psi \in X_\psi$ and $\exists\ j \in \{1, ..., n\}$ so that $\tilde{x}^\psi_j = x_{j,\min}$.*
   – *If the forward optimization direction is maximization, $\exists\ \tilde{x}^\psi \in X_\psi$ and $\exists\ j \in \{1, ..., n\}$ so that $\tilde{x}^\psi_j = x_{j,\max}$.*
2. $\mathcal{D}^{f,+}_j \subseteq \mathcal{D}$.
3. $\exists M \in \mathbb{R}$ *where* $M > d^{UB}_j \tilde{x}^\psi_j$.

Proof. For any $d^f \in \mathcal{D}^{f,+}_j$, the non-negative term $d^f_j x_j$ is the only component contributing to the forward objective value $(d^f)^T x$. If the forward objective is minimization, no minimization of $d^f_j x_j$ is possible below $d^f_j \tilde{x}^\psi_j$. Similarly, if the forward objective is maximization, no maximization of $d^f_j x_j$ is possible above $d^f_j \tilde{x}^\psi_j$. Condition (3) ensures the objective is bounded. Thus, $\tilde{x}^\psi$ is optimal to $\mathcal{MILP}\langle d, X\rangle$ for any $d^f \in \mathcal{D}^{f,+}_j$. □

An analogous theorem can be defined for negative d^f_j values that isolate a non-positive objective component $d^f_j x_j$, which we omit in the interests of space.

4 The *NCXplain* Algorithm

This section introduces *NCXplain*, a novel non-convex quadratic programming algorithm which optimally solves an $\mathcal{NCE}_{\mathcal{MILP}}$. Letting $\mathcal{S}$ be the set of all extreme points of $conv(X)$ and decision vector $x^{\psi} \in X_{\psi}$ be a foil, the $\mathcal{NCE}_{\mathcal{MILP}}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ can be expressed as:

$$\min_{d, x^{\psi}, u} \ ||d - c||_1 \tag{9}$$

$$\text{s.t. } d \cdot x^{\psi} \leq d \cdot x^0 \qquad \forall\ x^0 \in \mathcal{S} \tag{10}$$

$$A^T u \geq d \tag{11}$$

$$x^{\psi} \in X_{\psi},\ d \in \mathcal{D},\ u \in \mathbb{R}^v_+. \tag{12}$$

Constraints (10) force a foil to have a forward objective no worse than any extreme point of $conv(X)$, and have a non-convex, quadratic left-hand side which is bilinear in d and x^{ψ}. Constraints (11) ensure that $\mathcal{MILP}\langle d, X\rangle$ is bounded by forcing its dual problem to be feasible.

NCXplain (Algorithm 2) follows a similar cutting plane approach to *InvMILP* (Algorithm 1), with the main difference being *NCXplain*'s quadratic master problem $\mathcal{MP}_{\mathcal{NCE}}$ (13)-(17) and stopping conditions. The $\mathcal{NCE}_{\mathcal{MILP}}$ objective is linearized using $d = c - g + h$, where $g, h \in \mathbb{R}^n_+$. Then, taking the $\mathcal{NCE}_{\mathcal{MILP}}$ (9) - (12) and relaxing constraints (10) by replacing $\mathcal{S}$ with a set of known extreme points $\mathcal{S}^0 \subseteq \mathcal{S}$ gives the $\mathcal{MP}_{\mathcal{NCE}}$ $\langle c, \mathcal{D}, \psi, x^*, X, \mathcal{S}^0\rangle$:

$$\min_{g, h, x, u} \ g + h \tag{13}$$

$$\text{s.t. } (c - g + h) \cdot x \leq (c - g + h) \cdot x^0, \forall\ x^0 \in \mathcal{S}^0 \tag{14}$$

$$A^T u \geq c - g + h \tag{15}$$

$$x \in X_{\psi},\ (c - g + h) \in \mathcal{D} \tag{16}$$

$$g, h \in \mathbb{R}^n_+, u \in \mathbb{R}^v_+. \tag{17}$$

Given an optimal $\mathcal{MP}_{\mathcal{NCE}}$ solution $(d^i, x^{\psi,i}, u^i)$ at iteration i of *NCXplain*, solving a subproblem $\mathcal{MILP}\langle d^i, X\rangle$ to get an optimal extreme point $x^{0,i}$ allows *NCXplain* to either show d^i is an optimal solution to the $\mathcal{NCE}_{\mathcal{MILP}}$ or add a new extreme point of $conv(X)$ to $\mathcal{S}^0$. The complete *NCXplain* algorithm is given by Algorithm 2, and its properties are formalized by Lemmas 1 - 2 and Theorem 2. Both the master problem and the MILP subproblem can be modelled in Gurobi 9.0+ due to recent advances allowing non-convex quadratic constraints such as (14) to be expressed directly.

Lemma 1. *NCXplain only terminates in Step 3 if d^i is feasible for $\mathcal{NCE}_{\mathcal{MILP}}$.*

Proof. (Lemma 1). If $d^i \cdot x^{0,i} = d^i \cdot x^{\psi,i}$ (Case 1), then the foil $x^{\psi,i}$ is optimal to $\mathcal{MILP}\langle d^i, X\rangle$. If $d^i \cdot x^{0,i} < d^i \cdot x^{\psi,i}$ but $x^{0,i} \in X_{\psi}$ (Case 2), then $x^{0,i}$ is a foil and optimal to $\mathcal{MILP}\langle d^i, X\rangle$. □

Algorithm 2: *NCXplain*.

```
Inputs: NCE_{MILP}⟨c, D, ψ, x*, X⟩
Output: d*
Step 1: Initialize S^0 ← x*.
Step 2: Solve MP_{NCE} ⟨c, D, ψ, x*, X, S^0⟩.
  If infeasible, return INFEAS.
  Else get (d^i, x^{ψ,i}, u^i) with d^i = (c − g^i + h^i).
Step 3: Solve MILP⟨d^i, X⟩ to get x^{0,i}.
  If d^i · x^{0,i} = d^i · x^{ψ,i} (Case 1)
   Stop and return d* = d^i.
  Elif d^i · x^{0,i} < d^i · x^{ψ,i} and x^{0,i} ∈ X_ψ (Case 2)
   Stop and return d* = d^i.
  Else (Case 3)
   Update S^0 = S^0 ∪ {x^{0,i}}, go to Step 2.
```

Lemma 2. *Let $\mathcal{S}^{0,i}$ be $\mathcal{S}^0$ during iteration i of NCXplain. If in Step 3, $x^{0,i} \in \mathcal{S}^{0,i}$, NCXplain must terminate. If $x^{0,i} \notin \mathcal{S}^{0,i}$, then NCXplain either terminates or a new extreme point of $conv(X)$ is added to $\mathcal{S}^0$ before iteration $i+1$.*

Proof. (Lemma 2). If $x^{0,i} \in \mathcal{S}^{0,i}$, then due to constraint (14), $x^{\psi,i}$ must satisfy $d^i \cdot x^{\psi,i} \leq d^i \cdot x^{0,i}$, but due to the optimality of $x^{0,i}$ to $\mathcal{MILP}\langle d^i, X\rangle$, $d^i \cdot x^{\psi,i} \not< d^i \cdot x^{0,i}$. Thus, $d^i \cdot x^{\psi,i} = d^i \cdot x^{0,i}$ (Case 1), and *NCXplain* must terminate. If $x^{0,i} \notin \mathcal{S}^{0,i}$ and *NCXplain* does not terminate (Case 3), the new extreme point of $conv(X)$, $x^{0,i}$, is added to $\mathcal{S}^0$. □

Theorem 2 *NCXplain will optimally solve an $\mathcal{NCE}_{\mathcal{MILP}}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ or prove it is infeasible in a finite number of iterations.*

Proof. (Theorem 2). Let $X_{\mathcal{MP}}$ and $X_{\mathcal{NCE}}$ represent the solution sets of $\mathcal{MP}_{\mathcal{NCE}}$ (13) - (17) and $\mathcal{NCE}_{\mathcal{MILP}}$ (9) - (12), respectively. Since the two problems have the same objective, differ only in constraints (10) and (14), and $\mathcal{S}^0 \subseteq \mathcal{S}$, then $X_{\mathcal{NCE}} \subseteq X_{\mathcal{MP}}$ and $\mathcal{MP}_{\mathcal{NCE}}$ is a relaxation of $\mathcal{NCE}_{\mathcal{MILP}}$. Thus, if an $\mathcal{MP}_{\mathcal{NCE}}$ is found infeasible in Step 2, then $\mathcal{NCE}_{\mathcal{MILP}}$ must also be infeasible. Similarly, if a solution $(d^i, x^{\psi,i}, u^i)$ is optimal to an $\mathcal{MP}_{\mathcal{NCE}}$ and d^i is feasible to $\mathcal{NCE}_{\mathcal{MILP}}$, then d^i must also be optimal to $\mathcal{NCE}_{\mathcal{MILP}}$.

By these observations and Lemmas 1 and 2, in any iteration, *NCXplain* either terminates having proven the $\mathcal{NCE}_{\mathcal{MILP}}$ is infeasible, terminates having found an optimal $\mathcal{NCE}_{\mathcal{MILP}}$ solution d^*, or continues after adding a new extreme point of $conv(X)$ to $\mathcal{S}^0$. Because $\mathcal{S}$ is a finite set, the number of iterations before $\mathcal{S}^0 = \mathcal{S}$ is finite, and when $\mathcal{S}^0 = \mathcal{S}$, Lemma 2 implies that *NCXplain* must terminate since any extreme point $x^{0,i}$ obtained in Step 3 is in $\mathcal{S}$. □

5 Experimental Method

Simulations demonstrating our explanation approach and testing *NCXplain* were carried out based on two forward MILP problems. These experiments were performed in three steps, focusing on the last:

1. Optimally solving a $\mathcal{MILP}\langle c, X\rangle$ instance to get x^*.
2. Simulating a contrastive question and creating a $\mathcal{NCE}_{\mathcal{MILP}}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ instance.
3. Optimally solving the $\mathcal{NCE}_{\mathcal{MILP}}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ with *NCXplain*.

We do not numerically compare *NCXplain* to alternatives because it is the first algorithm capable of solving an $\mathcal{NCE}_{\mathcal{MILP}}$.

5.1 Forward Problems

The two forward MILP problems were the 0-1 knapsack problem (KP) and the single machine scheduling with release dates problem, $1|r_j|\sum w_jC_j$. The KP was selected because it is NP-complete [19], has a simple structure, and is easy to understand. The scheduling problem was chosen because it matches a potential use case for $\mathcal{NCE}_{\mathcal{MILP}}$ based explanations (Examples 1 - 3) and is a relatively simple, though strongly NP-Hard [16] problem.

0-1 Knapsack Problem (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 $W \in \mathbb{N}_0$, with $W < \sum_{i=1}^{n} w_i$. A decision variable $x_i \in \{0, 1\}$, $i \in \{1, ..., n\}$, is assigned to 1 if item i is included in the knapsack and 0 otherwise, and the complete KP is $\max_x\{c \cdot x : x \in X\}$, $X = \{x \in \{0, 1\}^n : w \cdot x \leq W\}$. Problem instances of sizes $n \in \{250, 500, 1000\}$ were generated using independent random uniform distributions $c_i \in [1, R]$ and $w_i \in [1, R]$ with $R = 1000$, where $W = \max\{\lfloor P\sum_{i=1}^{n} w_i\rfloor, R\}$ with $P = 0.5$.

Scheduling Problem ($1|r_j|\sum w_jC_j$) There are $n \in \mathbb{N}$ jobs with each job $i \in \{1, ..., n\}$ having a processing time $q_i \in \mathbb{N}$, a weight[4] $c_i \in \mathbb{N}$, a release date $r_i \in \mathbb{N}_0$, and a completion time $t_i^c \in \mathbb{N}_0$. The objective is to minimize the weighted sum of all completion times, $c \cdot t^c$, given that no job can start before its release date or be interrupted and no two jobs can be processed at the same time. Letting $x_{i,t} \in \{0, 1\}$ be a decision variable which is 1 if job i starts at time t and 0 otherwise, and T be an upper bound on latest completion time of any

[4] Though w is typically used for job weights, we use c instead to keep notation consistent throughout the paper.

job, a MILP model for $1|r_j|\sum w_j C_j$ is

$$\min_x \sum_{i=1}^{n} \sum_{t=0}^{T-q_i} c_i(t+q_i)x_{i,t} \tag{18}$$

$$\text{s.t.} \sum_{t=0}^{T-q_i} x_{i,t} = 1, \quad \forall\ i = 1, ..., n \tag{19}$$

$$\sum_{i=1}^{n} \sum_{s=\max(0,t-q_i+1)}^{t} x_{i,s} \leq 1,\ \forall\ t = 0, ..., T-1 \tag{20}$$

$$\sum_{t=0}^{r_i-1} x_{i,t} = 0, \quad \forall\ i \in \{1, ..., n\} \tag{21}$$

$$x_{i,t} \in \{0,1\}^{n\times(T-1)}. \tag{22}$$

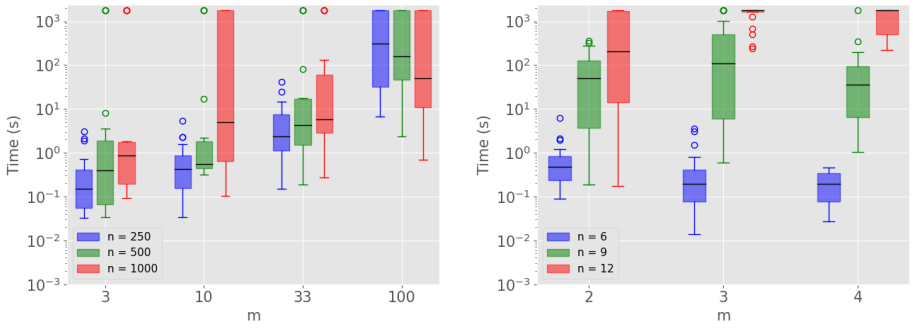
Constraints (19) force each job to start exactly once. Constraints (20) ensure no two jobs are processed at the same time, and constraints (21) enforce the release dates. Problem instances of sizes $n \in \{6, 9, 12\}$ were generated using random uniform distributions $q_i \in [1, 10]$, $c_i \in [1, 10]$, and $r_i \in [0, \lfloor \alpha Q \rfloor]$, where $\alpha = 0.3$ and $Q = \sum_{i=1}^{n} q_i$, and the time horizon T was calculated as $T = \lfloor \alpha Q \rfloor + Q$.

5.2 $\mathcal{NCE}_{\mathcal{MILP}}$ Instances

Knapsack Questions Given a subset of m items $\mathcal{S}_\psi \subseteq \{1, ..., n\}$, $|\mathcal{S}_\psi| = m$, $\mathcal{S}_\psi \neq \emptyset$, and a parameter $\beta_\psi \in (0, 1]$, the simulated question asked "Why were at least $\beta_\psi m$ items from $\mathcal{S}_\psi$ not included in the knapsack?". The foil set corresponding to this question is $X_\psi = \{x \in X : \sum_{j\in\mathcal{S}_\psi} x_j \geq \beta_\psi m\}$. Questions were simulated with $\beta = 0.75$ by randomly selecting m items to form $\mathcal{S}_\psi$ such that $x^* \notin X_\psi$.

Scheduling Questions The simulated question asked why m randomly selected jobs $\mathcal{M} \subseteq \{1, ..., n\}$ were not scheduled earlier, as in Examples 1 - 3. Letting $t^* \in [0, T]^n$ denote job start times in x^*, a $t^\psi \in [0, T]^m$ was created with t_j^ψ representing the maximal counterfactual start time of job $j \in \mathcal{M}$ such that $r_j \leq t_j^\psi < t_j^*$. Then, the question asked "Why was each job $j \in \mathcal{M}$ not completed by $(t_j^\psi + q_j)$, respectively?". This question is represented with the foil set $X_\psi = \{x \in X : \sum_{t=0}^{T-q_j} t x_{j,t} \leq t_j^\psi \quad \forall\ j \in \mathcal{M}\}$. For each job $j \in \mathcal{M}$, the maximal counterfactual start time t_j^ψ was randomly selected from the interval $[t_j^{\psi,LB}, t_j^{\psi,UB}]$, where $t_j^{\psi,UB} = t_j^* - 1, t_j^{\psi,LB} = \lceil r_j + \theta(t_j^* - 1 - r_j) \rceil$, and $\theta = 0.5$.

Non-Empty Foil Sets For both problems, after a foil set was generated, it was checked whether X_ψ was non-empty. If X_ψ was empty, the question data was re-randomized until a non-empty foil set was found, though such cases were rare.



(a) Knapsack $\mathcal{NCE}_{\mathcal{MILP}}$ Problems (b) Scheduling $\mathcal{NCE}_{\mathcal{MILP}}$ Problems

Fig. 1: *NCXplain* Runtime Distributions.

Counterfactual Objectives The set of feasible counterfactual objectives was set to $\mathcal{D} = \{d \in \mathbb{N}_0^n : 0 \leq d_i \leq c_i^{UB} \ \forall i \in \{1, ..., n\}\}$, where $c_i^{UB} \in \mathbb{N}$ is the maximum value for c_i in a forward instance ($c_i^{UB} = 1000$ for KP, $c_i^{UB} = 10$ for $1|r_j|\sum w_j C_j$). Given that X is finite for both forward problems, it is impossible for any $d \in \mathcal{D}$ to result in an unbounded $\mathcal{MILP}\langle d, X\rangle$, so constraints (15) were omitted in these simulations.

$\mathcal{NCE}_{\mathcal{MILP}}$ Feasibility Any $\mathcal{NCE}_{\mathcal{MILP}}$ in these experiments meets Conditions (1)-(3) of Theorem 1, and thus has a non-trivial feasible solution $d^f \in \mathcal{D}_j^{f,+}$. Intuitively, any $d^f \in \mathcal{D}_j^{f,+}$ implies that in the KP, there is no benefit from including any items other than item j, while in the scheduling problem, there is no benefit from achieving an earlier completion time for any jobs other than job j. Specifically, Condition (2) is satisfied since $\mathcal{D}_j^{f,+} \subseteq \mathcal{D}$ for any $j \in \{1, ..., n\}$. Condition (1) is met by the KP instances since the maximal value of any x_j is 1, and any foil $x^\psi \in X_\psi$ must contain at least one component $x_j^\psi = 1$. For the scheduling problem, taking any schedule $x^\psi \in X_\psi$ and left-shifting it causes the first job in the schedule, which we will call job j, to start at its release date r_j. Condition (1) is thus satisfied since there exists a schedule in the foil set where job j is completed at its minimal possible time, $t^c_{j,\min} = r_j + q_j$. Finally, Condition (3) is met since $d_j^f x_{j,\max}$ is bounded from above by c^{UB} for KP, while for scheduling, $d_j^f t^c_{j,\min}$ is bounded from above by $c^{UB}(r_j + q_j)$.

5.3 Computational Details

Python 3.9.7 and Gurobi 9.5 were used to implement *NCXplain* for $\mathcal{NCE}_{\mathcal{MILP}}$ instances, as well as to solve the initial $\mathcal{MILP}$ instances. Twenty instances were tested for each value of (n, m) reported in Section 6, using a single core of a 2.6 GHz Intel Core i7-10750H CPU. A time limit of 30 minutes was used

n	m	$t_{F,\mu}$	$t_{F,\sigma}$	$t_{\mathcal{NCE}}$	$t_{\mathcal{MP}}$	$t_{\mathcal{SP}}$	n_{ITR}	n_S
250	3	0.003	0.001	0.5	0.5	0.01	6	20
	10	0.003	0.001	0.9	0.8	0.02	8	20
	33	0.003	0.001	6.8	6.6	0.07	28	20
	100	0.003	0.001	862.4	853.2	1.16	860	11
500	3	0.004	0.001	270.8	259.6	1.16	443	17
	10	0.005	0.001	271.5	259.1	1.39	469	17
	33	0.004	0.001	278.4	270.0	0.96	361	17
	100	0.004	0.001	778.0	770.5	1.18	355	12
1000	3	0.008	0.003	360.7	343.4	2.11	376	16
	10	0.011	0.003	721.3	648.5	10.14	1129	12
	33	0.008	0.003	371.3	351.6	2.25	420	16
	100	0.008	0.003	655.7	640.2	3.00	345	13

Table 1: Knapsack Explanation Results

for *NCXplain*, and if an $\mathcal{NCE}_{\mathcal{MILP}}$ was not solved before this time limit, its runtime was recorded as 30 minutes. Thus, the *NCXplain* runtimes should be interpreted as lower bounds on the true runtimes.

6 Experimental Results

Figure 1 illustrates the *NCXplain* runtime distributions. For *NCXplain*, Tables 1 and 2 report the mean runtime ($t_{\mathcal{NCE}}$), mean number of iterations (n_{ITR}), the number of instances solved optimally before the time limit (n_S), as well as the mean cumulative time in the subproblem ($t_{\mathcal{SP}}$) versus the master problem ($t_{\mathcal{MP}}$). For the initial forward problem $\mathcal{MILP}\langle c, X\rangle$, these tables show the runtime mean ($t_{F,\mu}$) and standard deviation ($t_{F,\sigma}$). All runtimes are in seconds.

No instances were infeasible, demonstrating the successful use of Theorem 1. The number of instances solved (n_S) shows that most $\mathcal{NCE}_{\mathcal{MILP}}$ instances were solved in under 30 minutes, though the solution times for the initial forward problem ($t_{F,\mu}$) were much faster.

As indicated by the values of $t_{\mathcal{MP}}$ versus $t_{\mathcal{SP}}$, *NCXplain* spends 90%-99.9% of its runtime solving master problems (Step 2, Algorithm 2), showing that these non-convex, quadratic problems are significantly harder than the $\mathcal{MILP}\langle d, X\rangle$ subproblems (Step 3, Algorithm 2). That is, it is computationally cheaper to add a new point to $\mathcal{S}^0$ than to solve $\mathcal{MP}_{\mathcal{NCE}}$. A direction for future work may thus be to reduce the number of master problem iterations with a variation of *NCXplain* which adds multiple points to $\mathcal{S}^0$ for each iteration of $\mathcal{MP}_{\mathcal{NCE}}$.

While a larger n almost always resulted in longer $\mathcal{NCE}_{\mathcal{MILP}}$ solve times, the effects of m (the number of items or jobs in a user question), as well as compound effects of n and m, are difficult to observe from our data. Future work should study these effects more rigorously, including investigating whether any phase transitions exist. The existence of phase transitions may explain why the scheduling $\mathcal{NCE}_{\mathcal{MILP}}$s with $n = 6$ became easier as m increased, while those with $n = 12$ became harder (Figure 1b, Table 2).

n	m	$t_{F,\mu}$	$t_{F,\sigma}$	$t_{\mathcal{NCE}}$	$t_{\mathcal{MP}}$	$t_{\mathcal{SP}}$	n_{ITR}	n_S
6	2	0.006	0.002	0.9	0.9	0.01	4	20
	3	0.005	0.002	0.6	0.6	0.02	4	20
	4	0.005	0.002	0.2	0.2	0.02	5	20
9	2	0.010	0.002	89.5	89.3	0.08	7	20
	3	0.011	0.005	467.0	466.6	0.15	13	16
	4	0.012	0.008	150.8	150.3	0.19	15	19
12	2	0.025	0.022	694.8	694.5	0.13	7	15
	3	0.016	0.005	1499.0	1498.4	0.24	13	5
	4	0.021	0.013	1365.4	1364.6	0.33	17	6

Table 2: Scheduling Explanation Results

7 Limitations and Future Work

Algorithmic Improvements Currently, *NCXplain* can only explain small $\mathcal{MILP}$ problems. However, Bodur *et al.* [1] recently showed that *InvMILP* can be sped up by modifying Step 3 of Algorithm 1 to add non-extreme point solutions of $\mathcal{MILP}\langle c, X\rangle$ to $\mathcal{S}^0$ after finding these points using early stopping criteria and trust regions. Additionally, Duan and Wang [9] extend *InvMILP* with a heuristic to parallelize cut generation and compute feasible solutions as upper bounds to the inverse MILP problem. These extensions to *InvMILP* can likely be adapted to *NCXplain* to improve performance and produce feasible solutions to $\mathcal{NCE}_{\mathcal{MILP}}$ before the problem is solved optimally.

Minimizing Decision Perturbation A limitation of the $\mathcal{NCE}_{\mathcal{MILP}}$ is that an optimal solution to $\mathcal{MILP}\langle d^*, X\rangle$ may be arbitrarily far from x^*, the decision being explained. Given an arbitrarily large change to x^*, an explainee may find it difficult to evaluate the effects of perturbing c on the decision, especially if multiple iterations of explanation and objective modification are performed. A future modification to the $\mathcal{NCE}_{\mathcal{MILP}}$ and *NCXplain* could add a term $|x^* - x^\psi|$ to the objective (9), thus optimizing for smaller perturbations to both c and x^*.

Actionability and Sparsity The concepts of actionability [22] and sparsity [18] could be adapted from machine learning (ML) to $\mathcal{NCE}_{\mathcal{MILP}}$ explanations. Since some objective components c_i may be easier to change, or more actionable, than others, a weighted L_1 norm $w^T||d - c||_1$, $w \in \mathbb{R}^n_+$, could replace objective (9), with weight w_i representing the ease of changing parameter c_i. To induce sparsity, an L_0 term measuring the number of perturbed objective components could be added to objective (9), since an explainee may prefer explanations perturbing fewer componenets of c.

Meaningful Objectives A fundamental assumption in an $\mathcal{NCE}_{\mathcal{MILP}}$ is that the objective parameters represented by c and d are meaningful to the explainee. Otherwise, the explainee requires an additional explanation of what these parameters mean before the $\mathcal{NCE}_{\mathcal{MILP}}$ explanation is useful.

8 Related Work

Our past work [14, 15] discussed in Section 2, introduced the NCE (1) - (2) and solved two restricted versions of it. The only other work we are aware of which uses counterfactual explanations for a model-based optimization problem is that of Brandao *et al.* [2], in which inverse optimization is applied in its classical form to explain a path planning problem. As mentioned in Section 2.3, the inverse optimization problem can be interpreted as a special case of an NCE where the explainee is interested in exactly one alternative solution x^d. Our approach is more general since we enable an explainee to define a set of alternative solutions using linear or quadratic constraints.

In the inverse optimization literature, Wang [26] formulates a variant of inverse optimization which is similar to the $\mathcal{NCE}_{\mathcal{MILP}}$. However, this variant assumes d is continuous, while the $\mathcal{NCE}_{\mathcal{MILP}}$ allows the domain of $d \in \mathcal{D}$ to be an integer or mixed-integer set, such as the set of integer job priorities in the scheduling experiments. Additionally, other than the equivalent of bilinear constraints (10), all constraints in Wang's problem must be linear, while the $\mathcal{NCE}_{\mathcal{MILP}}$ allows constraints (12) defining X_ψ and $\mathcal{D}$ to be quadratic. Most notably, while providing interesting theoretical contributions, Wang's work does not connect inverse optimization with explanation, the focus of our paper.

Finally, an emerging literature on counterfactual explanations exists in ML [23], providing potential for cross-polination with explainability research in declarative optimization, such as the actionability and sparsity extensions proposed in Section 7.

9 Conclusion

This paper presents techniques to respond to users asking why an optimal solution x^* to a linear discrete optimization problem $\mathcal{MILP}\langle c, X\rangle$ did not satisfy some previously unstated constraints. We address such questions by formulating the $\mathcal{NCE}_{\mathcal{MILP}}$ (9)-(12), the solution to which is a counterfactual explanation d: an alternative objective vector minimally perturbed from c so that an optimal solution to $\mathcal{MILP}\langle d, X\rangle$ satisfies the additional user constraints. After establishing feasibility conditions for the $\mathcal{NCE}_{\mathcal{MILP}}$, we introduce *NCXplain*, a non-convex, quadratic cutting-plane algorithm which solves the $\mathcal{NCE}_{\mathcal{MILP}}$. Experiments are performed to simulate explanations for two discrete optimization problems, evaluating *NCXplain* and identifying next steps for improving it. Finally, we discuss future directions for counterfactual explanations in optimization such as actionability, sparsity, and minimizing decision perturbation.

References

1. Bodur, M., Chan, T.C., Zhu, I.Y.: Inverse mixed integer optimization: Polyhedral insights and trust region methods. INFORMS Journal on Computing (2022)

2. Brandao, M., Coles, A., Magazzeni, D.: Explaining path plan optimality: Fast explanation methods for navigation meshes using full and incremental inverse optimization. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 31, pp. 56–64 (2021)
3. Chakraborti, T., Sreedharan, S., Kambhampati, S.: The emerging landscape of explainable automated planning & decision making. In: IJCAI. pp. 4803–4811 (2020)
4. Chakraborti, T., Sreedharan, S., Zhang, Y., Kambhampati, S.: Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In: IJCAI (2017)
5. Chan, T.C., Mahmood, R., Zhu, I.Y.: Inverse optimization: Theory and applications. arXiv preprint arXiv:2109.03920 (2021)
6. Demange, M., Monnot, J.: An introduction to inverse combinatorial problems. Paradigms of Combinatorial Optimization: Problems and New Approaches pp. 547–586 (2014)
7. Doshi-Velez, F., Kortz, M.: Accountability of AI under the law: The role of explanation. Tech. rep., Berkman Klein Center Working Group on Explanation and the Law, Berkman Klein Center for Internet and Society (2017)
8. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)
9. Duan, Z., Wang, L.: Heuristic algorithms for the inverse mixed integer linear programming problem. Journal of Global Optimization **51**(3), 463–471 (2011)
10. Eiffer, R., Cashmore, M., Hoffmann, J., Magazzeni, D., Steinmetz, M.: A New Approach to Plan-Space Explanation: Analyzing Plan-Property Dependencies in Oversubscription Planning. In: AAAI (2020)
11. Epstude, K., Roese, N.J.: The functional theory of counterfactual thinking. Personality and social psychology review **12**(2), 168–192 (2008)
12. Freuder, E.: Explaining ourselves: human-aware constraint reasoning. In: AAAI (2017)
13. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al.: Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In: International conference on machine learning. pp. 2668–2677. PMLR (2018)
14. Korikov, A., Shleyfman, A., Beck, J.C.: Counterfactual explanations for optimization-based decisions in the context of the GDPR. In: International Joint Conferences on Artificial Intelligence (IJCAI) (2021)
15. Korikov, A., Beck, J.C.: Counterfactual Explanations via Inverse Constraint Programming. In: Michel, L.D. (ed.) 27th International Conference on Principles and Practice of Constraint Programming (CP 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 210, pp. 35:1–35:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.CP.2021.35, https://drops.dagstuhl.de/opus/volltexte/2021/15326
16. Lenstra, J.K., Kan, A.R., Brucker, P.: Complexity of machine scheduling problems. In: Annals of discrete mathematics, vol. 1, pp. 343–362. Elsevier (1977)
17. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence **267**, 1–38 (2019)
18. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. pp. 607–617 (2020)
19. Pisinger, D., Kellerer, H., Pferschy, U.: Knapsack problems. Handbook of Combinatorial Optimization p. 299 (2013)

20. Senthooran, I., Klapperstueck, M., Belov, G., Czauderna, T., Leo, K., Wallace, M., Wybrow, M., de la Banda, M.G.: Human-Centred Feasibility Restoration. In: Michel, L.D. (ed.) 27th International Conference on Principles and Practice of Constraint Programming (CP 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 210, pp. 49:1–49:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.CP.2021.49, https://drops.dagstuhl.de/opus/volltexte/2021/15340
21. Smith, D.E.: Planning as an iterative process. In: Twenty-Sixth AAAI Conference on Artificial Intelligence (2012)
22. Ustun, B., Spangher, A., Liu, Y.: Actionable recourse in linear classification. In: Proceedings of the Conference on Fairness, Accountability, and Transparency. pp. 10–19 (2019)
23. Verma, S., Dickerson, J., Hines, K.: Counterfactual explanations for machine learning: A review (2020), NeurIPS Workshop on ML Retrospectives, Surveys and Meta-Analyses
24. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the GDPR. Harv. JL & Tech. **31**, 841 (2017)
25. Wang, L.: Cutting plane algorithms for the inverse mixed integer linear programming problem. Operations Research Letters **37**(2), 114–116 (2009)
26. Wang, L.: Branch-and-bound algorithms for the partial inverse mixed integer linear programming problem. Journal of Global Optimization **55**(3), 491–506 (2013)