Counterfactual Explanations for Discrete Optimization

by

Anton Korikov

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

Department of Mechanical and Industrial Engineering
University of Toronto

Counterfactual Explanations for Discrete Optimization

Anton Korikov
Master of Applied Science
Department of Mechanical and Industrial Engineering
University of Toronto
2022

Abstract

This thesis develops the first application of counterfactual explanations to optimal solutions of discrete optimization problems. The techniques studied respond to a contrastive question: why did the optimal solution did not satisfy a previously unstated specification? The explanations take the form of alternative objective parameters which would have resulted in the optimal solution satisfying the additional specification, while minimally perturbing the initial objective parameters. Such explanations are formalized as the Nearest Counterfactual Explanation (NCE) problem, which is based on a variant of inverse optimization. A novel cutting plane algorithm is developed to solve NCEs which explain problems with linear objectives and constraints. Two special cases of NCEs are also studied, based on restrictive question and answer forms.

Acknowledgements

I would like to thank my thesis advisor, Christopher Beck, for his amazing guidance. His vision, knowledge, and commitment to excellence were essential to developing the contributions of this thesis. He has also taught me a great deal about being a researcher.

Thank you to Alexander Shleyfman for his help in developing the theoretical contributions in Chapter 4, and for being a co-author on the first conference paper published from this work [1].

I would also like to thank my second committee member, Merve Bodur, for all of her insightful feedback, and for teaching me much of the foundation of discrete optimization.

Thank you to all of my lab mates from TIDEL. They were always there when I needed their help, and I always found our discussions immensely valuable.

Finally, thank you to all of the researchers from various conferences and presentations who have shared their ideas about this work with me. They have made this thesis much better.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Automated systems are increasingly being deployed to make high-impact decisions [2, 3]. However, there is also a growing recognition of the negative outcomes that result from the implementation of under-scrutinized AI systems [2, 3, 4, 5]. For instance, Eubanks [3] has shown that automated decision systems, designed with no malicious intent, can propagate inequalities and biases across services providing child care or combating homelessness. In an effort to mitigate such undesirable effects by encouraging more interpretable decision-making, newly passed legislation in the European Union [2] states that an individual subjected to an adverse automated decision has a right to an explanation. While the legality and form of such a right to explanation is a topic of debate [6], there is widespread consensus on the need to make automated decision systems more explainable [4, 5, 7].

This thesis develops the first application of counterfactual explanations [8, 9] to optimal solutions of discrete optimization problems. A counterfactual explanation is a response to a contrastive question posed by a person (the explainee) asking why a decision was not different in a specified way; for example, a borrower may ask "Why was I denied a loan instead of approved?". The explanation then takes the form of the minimal change to the world that would lead to the decision being different in the way specified by the explainee. For instance, the borrower may be told "You would have been approved a loan if your income was $10,000 higher."

In this thesis, contrastive questions asking why the optimal solution of a discrete optimization problem does not lie in a different region of the feasible set are answered using counterfactual explanations based on objective parameters. First, the explainee identifies some additional constraints, not present in the initial problem definition, that are not satisfied by the initial optimal solution and asks "Why did the optimal solution not satisfy these additional constraints?". For instance, if an

optimization problem is used to schedule a production order at an automotive factory, a manager may ask "Why was the brake order not completed before next week?". Then, finding an explanation involves computing a modified objective vector so that an optimal solution to the modified problem would satisfy the additional constraints, and so that the initial objectives are minimally perturbed. In the scheduling example above, if the orders are scheduled based on some priority levels, an explanation may be "The brake order would have been completed within one week if its priority was at least two levels higher." The manager could then produce a new schedule with the brake priority increased by two levels, and assess whether this change is worthwhile given any changes to the rest of the schedule. In general, such explanations are meant to help an explainee better understand the effects of objective parameters on optimal decisions, and if necessary, change these parameters or contest objectives they believe to be unfair [8]. The practical use of counterfactual explanations is discussed further in Sections 2.3 and 3.4.

1.1 Contributions

This thesis formulates counterfactual explanations as an optimization problem called the Nearest Counterfactual Explanation problem (NCE), which can be interpreted as a variant of inverse optimization (Section 2.6.1). Solution approaches to NCEs are considered within the scope of explaining discrete optimization problems with linear objectives and constraints. The feasibility of NCEs, and thus the existence of explanations, is investigated and a set of feasibility conditions are identified. In addition, practical considerations such as privacy and the meaningfulness of objective parameters to a person are discussed.

As will be discussed in Chapter 3, NCEs are challenging problems to solve. However, several real-world cases can be identified where the form of the contrastive question and the objective parameters that the explainee is interested in can be used to simplify the solution process. One such case, studied in Chapter 4, is when all the information concerning the explainee in the objective function is isolated to one decision variable and one objective parameter. In these settings, it is shown that solving an NCE requires solving at most a logarithmic number of slightly modified versions of the initial problem. Another special case of NCEs, studied in Chapter 5, occurs when the explainee wants to know why a subset of variables were not assigned to a set of specified values. It is shown how NCEs of this kind can be solved with the help of classical inverse optimization algorithms (Section 2.6.2), and numerical experiments are used to test several explanation algorithms.

With the help of insights drawn from studying the special cases of NCEs above, the general NCE

is revisited. A solution algorithm, *NCXplain* (Algorithm 6.1), is developed by modifying a well-known inverse optimization cutting plane algorithm [10], relying on recent advances in quadratic programming solvers. Numerical simulations demonstrate the process of using NCEs to explain optimization problems, including the formulation of contrastive questions given an initial optimal solution. In these simulations, it is shown how the conditions for the feasibility of an NCE can be met, leading to the guaranteed existence of explanations when meeting these conditions is possible. Finally, future research directions for NCE-based explanations are identified, based on ideas in the literature on inverse optimization (Section 2.6) as well as the literature on counterfactual explanations in machine learning (Section 2.3.3).

1.2 Organization of Thesis

In addition to providing a literature review (Chapter 2), this thesis is organized as follows.

Chapter 3 formulates the problem of finding an explanation as the NCE and discusses its relationship to inverse optimization. Several feasibility conditions for the NCE are discussed, one being that the alternative feasible region identified by the explainee in the contrastive question must be non-empty. Then, some practical aspects of using NCEs in the real world are examined, including privacy, trivial explanations, and outcomes when explanations do not exist.

Chapter 4 studies the first special case of NCEs, called the Univariate-NCE (U-NCE) in which both the question and explanation isolate a single decision variable and a single objective parameter, respectively. Several use-cases are given for the U-NCE, with the primary one being an explainee desiring an explanation concerning only themselves in a multi-user system. When the contrastive question is based on a binary decision variable, it is shown that solving the U-NCE requires only solving a slightly modified version of the initial problem. If the decision variable in the question is integer, a solution method is proposed that involves solving at most a logarithmic number of modified initial problems.

Chapter 5 studies the second special case of NCEs, called Partial Assignment NCEs (PA-NCE), in which the explainee asks why the optimal solution did not satisfy a partial assignment. As with the U-NCE, an important application of the PA-NCE is enabling an explainee to isolate decisions that concern themselves from decisions related to other people. It is proven that a PA-NCE can be solved in two steps: 1) finding the best solution in the user-specified region; and 2) solving a classical inverse optimization problem based on the solution from the first step. To explain problems which are constraint programs, Chapter 5 introduces inverse constraint programming (CP) as a novel

inverse optimization methodology. The inverse CP algorithms include pure inverse CP and two inverse mixed-integer linear programming (MILP)/CP hybrids. Numerical experiments are then performed demonstrating the explanation process using PA-NCEs, and showing that an explanation approach using a MILP/CP hybrid inverse algorithm can outperform alternatives when CP is the state of the art for the initial problem.

Chapter 6 returns to the general NCE defined in Chapter 3. It develops a solution algorithm, called *NCXplain*, which extends a cutting plane algorithm for inverse MILP [10], in part by introducing a new set of bilinear quadratic constraints. Due to recent progress in discrete optimization solvers, these constraints can be implemented directly in Gurobi 9.0. Numerical simulations are then carried out, demonstrating that *NCXplain* can be used to answer much more complex questions than those in Chapters 4 and 5.

Chapter 7 concludes the thesis, identifying several directions for future work.

Chapter 2

Literature Review

2.1 Introduction

While the machine learning (ML) community has produced a great amount of explainability (or interpretability)[1] research over the last five years [12], explainability in optimization has attracted far less attention in this period [13]. Much of the work on explainability in optimization was done well before 2016 [14, 15, 16, 17, 18, 19], and has focused on explaining infeasibility, not optimality [13]. While some new research on explainable optimization has emerged in the last five years [20, 21, 22, 23], all of it builds on existing ideas about infeasibility-based explanations.

In contrast, in AI planning, the recent surge of explainable AI (XAI) research has inspired several new approaches to be adopted, including that of contrastive explanations [24, 25, 26, 27, 28]. Research interest towards explainability in AI planning has increased considerably, as evidenced by several recent workshops on explainable AI Planning (XAIP) [29]. One popular form of contrastive explanation, especially in ML, is that of counterfactual explanations [8]. Inspired by the work on contrastive explanation research in AI planning, this thesis applies the technique of counterfactual explanations to optimal solutions of discrete optimization problems.

Section 2.2 reviews XAI concepts and methods, looking at the processes by which explainability techniques provide useful information to the explainee. Then, Sections 2.3 and 2.4 review counterfactual explanations in ML and contrastive explanations in AI planning, respectively. Section 2.5 surveys past work on explanations in discrete optimization. Finally, Section 2.6 focuses on inverse combinatorial optimization [30], which forms part of the methodological basis for this thesis.

[1] Explainability and interpretability will be used interchangeably, as they are in the literature (e.g. [5, 11]).

2.2 Explainable AI

2.2.1 Explainability concepts

A structured outline of XAI is somewhat challenging. Lipton, Doshi-Velez, and Kim [5, 31] argue that much of the literature suffers from ambiguous or under-justified claims, in which a technique is argued to achieve interpretability without an adequate discussion of how, when, or under what assumptions. Likely due to these issues, as well as the nascent nature of the literature, there is limited coherency between XAI surveys [12, 32, 33, 34, 35], with most using quite different frameworks for classifying explanation approaches.

Rather than review each available framework, this discussion will be organized using a key distinction that appears consistently across the literature: *transparency* versus *post-hoc explanations* [5]. The transparency[2] of an AI is the extent to which its decision-making algorithm is understandable to people. Transparency can be considered on two levels: the level of the entire algorithm, called *simulatability*; and at the level of individual parts of the algorithm, called *decomposability* [5]. Further, given a decision algorithm and an accurate, complete description of it, we can define the algorithm's *intrinsic transparency* as the extent to which it is understandable to people without any additional information. When an AI's intrinsic transparency is insufficient to satisfy the need for an explanation, post-hoc explanations can be used to provide additional information about the AI's decisions or its decision-making process. For instance, each input to an intrinsically opaque (non-transparent) deep neural network (DNN) may be assigned an importance value representing the input's impact on the output [37, 38]. Though it is useful to distinguish between the concepts of transparency and post-hoc explanations, they are not mutually exclusive. In fact, in the previous example, a post-hoc explanation attempts to increase transparency by providing information, using feature importance, about the algorithm's decision-making mechanism.

While transparency and post-hoc explainability are used quite consistently across the XAI literature [32, 33, 34, 35], there remains little coherency in classifications of post-hoc methods. For instance, Du *et al.* [34] propose an organization based on *local* versus *global* methods. Local explanations are only applicable for a single decision instance; for example, in a medical prediction system, a local explanation may address the question "Why is John's predicted illness the flu?". In contrast, global explanations can address any decision the AI is capable of making, which, in our

[2]While this definition is generally accepted in the XAI literature (e.g. [4, 32, 33]), other areas may use the term differently. For instance, some work in the human factors literature uses transparency to connote that some part of a technology is invisible to the user, providing them with a unobstructed interface to "look through" while completing a task [36].

medical example, might respond to the question "How much of an impact does the presence of a headache have on the prediction of flu?". Alternatively, Lipton [5] considers four categories of post-hoc explanations: local explanations, verbal explanations, visual explanations, and explanations by example.

Instead of selecting one of such classifications for this review, it will be argued that a useful way to organize post-hoc explanations is based on the kind of information they convey. This information can broadly be divided into three categories: 1) relationships between inputs and outputs that increase transparency, 2) examples of inputs and outputs that do not increase transparency, and 3) relationships between the AI and human beliefs and concepts. These categories are introduced briefly below and discussed in more detail in Section 2.2.3.

The first category of information consists of relations between inputs and outputs that reveal something about how the decision mechanism works beyond its intrinsic transparency. This information is meant to increase the algorithm's transparency through an additional relation that describes some effect that inputs have on outputs; thus, we call this a *proxy relation.* For instance, such a proxy relation was established in the previous example by the importance levels assigned to the inputs of a DNN representing the inputs' impacts on the decision. Similar notions describing proxy models, also called surrogate models, exist in the literature [39, 32, 12], however, most commonly, they describe a function that could replace the real decision algorithm and thus serve as an explanation of it. Our definition is more general because a proxy relation does not need to provide decision-making capability. For example, identifying feature importance levels is not enough to produce decisions, but still increases transparency.

In contrast, a second category of information that could be communicated with a post-hoc explanation consists of examples of inputs and outputs that do not provide insights into *how* the decision mechanism works. For instance, case-based explanations [11], identify decisions that are similar to each other. As another example, counterfactual explanations [8] identify inputs that would have resulted in different outputs. In contrast to proxy functions, neither of these two methods rely on any understanding of the mechanism by which decisions are made, and are often called example-based explanations [32].

Finaly, post-hoc explanations could impart a third type of information: a relationship between an AI and beliefs or concepts that are meaningful to an explainee. We call this type of information a relationship to a *human concept or belief*, and note that these concepts and beliefs may or may not be used by the AI. For instance, in work by Kim *et al.* [4], given an image classifier that predicts whether an image contains a zebra, a person may ask "How important are *stripes* to this

prediction?". The classifier may not have any internal representation of the notion of stripes, so a meaningful explanation needs to establish a relationship between the human concept of stripes and the AI's decision mechanism.

Similar to the distinction between transparency and post-hoc explanations, these three categories are not mutually exclusive. Though the information conveyed by some post-hoc explanations is well characterized by a single category, it is possible for an explanation to provide information from multiple categories, as will be shown shortly. Figure 2.1 summarizes the concepts defined in this section for classifying explainability techniques. We now turn to a more detailed discussion of each concept, presenting examples from the XAI literature.

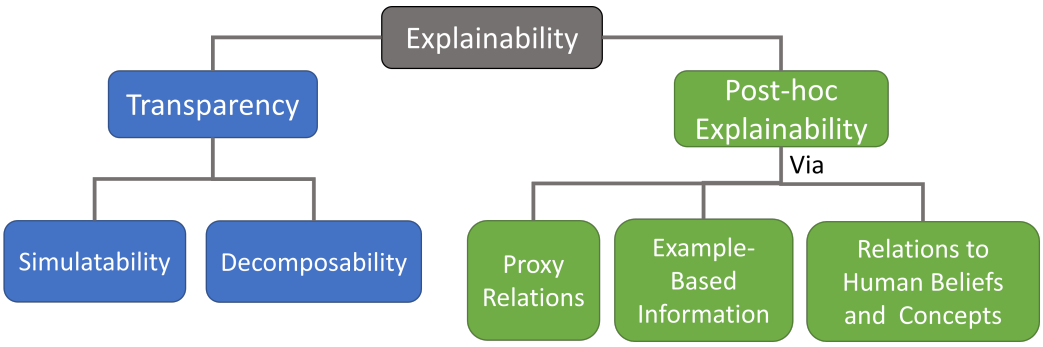


Figure 2.1: Explainability Concept Map

2.2.2 Transparency

Simulatability

The strongest notion of a transparent algorithm is one that a person can comprehend in its entirety, called simulatability. By this definition, given the inputs, a person should be able to simulate the steps of the algorithm to reproduce its output in a reasonable amount of time [5]. Given that a person understands the meaning of the inputs, this type of interpretability is claimed to be exhibited by sparse linear models [40] and sparse decision trees [41]. Sparsity is typically beneficial for simulatability, since, given an algorithm group (e.g., linear models, decision trees), algorithms with n components (e.g. variables, decision nodes) are generally more simulatable than algorithms with $n + 1$ components [40, 41].

Whether or not an algorithm is simulatable depends not only on the algorithm itself, but also on the capabilities of the explainee and the task for which an explanation is needed. For instance, an algorithm designer is usually more capable of understanding an algorithm than a layperson. Additionally, the task at hand influences what constitutes a reasonable amount of time for simulating

the algorithm. For example, a driver selecting a route might only spend seconds attempting to understand a routing algorithm, while a lawyer arguing that an automated loan approval system is discriminative may spend weeks. However, Lipton [5] argues that this ambiguity in how much time should be considered reasonable may only comprise several orders of magnitude, due to natural limits on human cognitive abilities.

In the context of constraint satisfaction and optimization problems, some algorithms used on very small problem instances could be considered simulatable. For instance, if a solver follows simple inference rules to solve a Sudoku or small logic puzzle, a person could follow the same inference steps to recreate the solution. Sqalli and Freuder [42] explore this type of interpretability for small logic puzzles, observing that their solver's inference traces closely resemble human generated explanations. However, after the number of decision steps in these algorithms exceeds a certain threshold, they are no longer simulatable because the number of steps is too large.

Decomposability

Transparency can also be considered at the level of the algorithm's components, called decomposability. From this perspective, how clear is the meaning of each parameter or variable? Are individual steps and calculations understandable? For instance, linear models are considered highly decomposable, assuming their inputs are understandeable, because they can be split into meaningful terms and coefficients [43]. Similarly, a person may be able understand which features (e.g. age, gender) are used by a single node in a decision tree, even though the entire tree may be too large to be simulatable [5].

Since most research on explainability, driven by the rise of neural methods, has focused on ML, there has been little recent discussion of the transparency of AI techniques that use some form of symbolic reasoning [44]. However, approaches which reason using symbols that have a human-understandable meaning, by definition, exhibit a certain level of decomposability. For instance, in constrained optimization paradigms such as constraint programming (CP) or mixed integer programming (MIP), there is typically a human-understandable meaning attached to the variables, constraints, objective function, and coefficients. Similarly, AI planning algorithms often use human-interpretable states, actions, goals, or costs, making them decomposable at the level of these symbols [44]. These algorithms may also be decomposable at the level of individual steps: for instance, a single inference step performed by a CP solver to fill an empty square in a Sudoku puzzle may be easy to understand, given that a person understands the rules of Sudoku. In contrast to declarative, model-based methods, DNNs are much more difficult to decompose into distinct, human-understandable

steps and components.

Using Intrinsic Transparency

Given that some algorithms are more intrinsically transparent than others, one XAI research direction is the design of efficient and intrinsically transparent algorithms that require no additional explanation [41]. Some authors even argue that no other algorithms should be used for high-stakes decisions [41]. However, other researchers caution against premature dismissals of more powerful but more opaque algorithms simply because they are not intrinsically transparent, arguing that post-hoc methods are a possible way to meet the goals of XAI [5].

2.2.3 Post-hoc Explanations

For algorithms whose intrinsic transparency is not sufficient to meet the demands for interpretability, post-hoc explanations provide additional information that aims to meet these needs. An advantage of post-hoc explanations is that they allow for separate approaches to decision-making and explanation; one method, perhaps efficient but opaque, can be used to make the decision, followed by a second method to explain it [5]. We now present a more thorough analysis of post-hoc methods based on the three categories of information that were identified above: proxy relations, example-based information, and relations to human beliefs and concepts.

Proxy Relations

Recall that proxy relations attempt to increase transparency by identifying a new relationship between inputs and outputs that reveals something about how the decision mechanism works. These relations may or may not be faithful to the real decision algorithm to varying degrees. Three examples of methods that use proxy relations are: local approximations, feature importance methods, and explanations learned from people.

- **Local Approximations:** Instead of attempting to explain how an algorithm would make a decision for any input, local explanations attempt to explain a single decision. In the work of Ribeiro *et al.* [38], the decision of any classifier is explained by learning a sparse linear model in the vicinity of a given input. When a sparse linear model is trained on a small region, it may achieve comparable performance to the initial classifier for predictions in that region. However, the new model's performance may be unpredictably bad for far-away inputs [8].

- **Feature Importance Methods:** Feature importance methods identify values that represent how much impact each input has on the output. These methods include local linear approximations discussed above, since the weight assigned to each term in a linear function can be interpreted as its importance coefficient. Another well known method is SHapley Additive exPlanation (SHAP) [37] in which a feature's importance is represented by its Shapley value: a game-theoretic measure which represents the marginal contribution of a feature to the outcome (i.e. the effect of the feature being present as opposed to absent) averaged across all possible combinations of features.

- **Learned Explanations:** People regularly explain how certain facts lead to certain outcomes. Algorithms can learn to recreate these human explanations and use them as proxy relations. For example, Krening *et al.* [45] train one algorithm to play a video game, and a second one to generate verbal explanations of strategy based on a training set of human explanations. Such explanations are not necessarily faithful to the decision – the explanation algorithm simply minimizes the difference between predicted explanations and previously observed explanations. As another example, McAuley and Leskovec [46] build a recommender system where a user's text reviews are considered as explanations of their item ratings. The recommender system is simultaneously trained to predict a user-item rating and the likelihood of topics, learned via a topic model, appearing in a review. It then outputs the most likely topics as an explanation of its rating.

Example-Based Information

While the previous post-hoc examples work by providing some additional understanding of how the decision mechanism works, post-hoc methods can also communicate information using examples of inputs and outputs that do not increase transparency. In fact, Wacther *et al.* argue that a benefit of this kind of information is that it may be easier to understand for users who do not have any previous algorithmic knowledge [8]. Two broad classes of methods that convey this kind of information are case-based explanations and counterfactual explanations.

- **Case-based Explanations:** People often provide case-based explanations, for example doctors relying on case studies to justify treatment decisions [5], and furthermore, case-based reasoning has been demonstrated to be a fundamental human decision-making strategy [47, 48, 49]. For example, skilled firemen have been shown to make decisions by matching new situations to prototypical cases where certain decisions are appropriate [48]. Kim *et al.* [11] argue that

not only is case-based XAI useful for explanations, but it also allows AI to be more directly integrated with human decision-making processes – one of the underlying motivations for XAI. One example of this approach is work by Caruana *et al.* that provides a case-based explanation for a DNN prediction by finding the k-nearest data points in the training set which result in similar hidden layer activations [50]. This method is also used by word2vec, which learns word embeddings, to examine which words a model considers similar [51]. In Bayesian methods, Kim *et al.* [11] explain clustering results by identifying a prototype that best represents the characteristics of each cluster; for instance, if an AI explains that *Scream* is a prototype for a cluster of movies, a human might infer that the cluster represents horror films. Another Bayesian example is work by Doshi-Velez *et al.* [52], who explain predicted document topics based on their similarity to topics in pre-defined human ontologies, such as the ACM CCS ontology [53].

- **Counterfactual Explanations:** In contrast to case-based explanations, which rely on similar decisions, counterfactual explanations reveal alternative inputs that would have led to different decisions [8]. Given a decision, these explanations address the contrastive question: "Why was the decision not different in some way?" (e.g. "Why was I denied a loan instead of approved?"). The explanation takes the form of a set of hypothetical inputs that would have resulted in the decision being different (e.g. "If your income had been $\$10,000$ higher, you would have been approved"). Typically, it is desirable for these hypothetical inputs to be minimally perturbed from the initial ones. Counterfactual explanations are discussed in considerably more detail in Section 2.3.

Relations to Human Beliefs and Concepts

A third kind of information that post-hoc explanations can convey are relationships between the AI and human beliefs and concepts: beliefs and concepts that are meaningful to people, but may not be used by the AI to make decisions. Kim *et al.* [4] point out that the space of mathematical concepts used by an AI is often different from the space of concepts in which human reasoning takes place. There are many reasons why this could be the case, including that people and AI often use different information, have different goals, and represent knowledge differently. For instance, a deep image classifier performs inference using low-level pixels inputs and neural activations, while a person understands images through high-level concepts such as animals or body parts [4]. In declarative, model-based AI systems, a person may have a different mental model of a system than the model

used by the AI; for instance, given an AI planning problem, a solver and a person may have different beliefs about goals, actions, states, or costs [54]. Thus, it may be useful for an explanation to establish a relationship between the beliefs and concepts meaningful to the explainee and those used by the AI. In fact, based on his review of social science research, Miller [55] argues that such relationships are an important part of explanations because explanations are *social*. That is, explanations should attempt to reflect not only the beliefs of the explainer, but also how those beliefs relate to the beliefs of the explainee. Three examples of post-hoc methods which use this kind of information are model reconciliation, concept activation vector techniques, and counterfactual explanations which allow people to define their own questions.

- **Model Reconciliation:** In declarative, model-based settings where discrepancies between human mental models and AI models are possible, such as the AI planning example above, Chakraborti *et al.* [29, 54] argue that it can be useful to frame an explanation as a model reconciliation process. Specifically, this process consists of finding a sequence of updates to the two models until the human and AI models are in agreement.

- **Concept Activation Vectors:** Kim *et al.* [4] explain a DNN classifier by evaluating how important a custom, human-defined concept is to a model's decision, for instance addressing the question "How important is the concept of stripes to the classification of zebra?". They first allow a user to define two sets of inputs: one set where each input contains the concept, such as striped objects, and a second where the inputs do not have the concept, such as random objects. After the two sets are mapped to a latent representation by the neural network, a hyperplane is learned separating the two sets in the latent space, which allows a vector orthogonal to this hyperplane and in the direction of the set including the concept to be defined. This vector is called the *concept activation vector*, and, given some decision, the directional derivative of the output score with respect to this vector can be used to represent the importance that the human concept has on the DNN's output. Not only does this post-hoc method establish a relationship with a human concept, but it also identifies the importance of this human concept to the decision, thus increasing transparency through a proxy relation. This technique demonstrates that it can be useful for post-hoc explanations to convey more than one kind of information.

- **Counterfactual Explanations Using Human Questions:** Some counterfactual explanations allow people to use contrastive questions to describe custom properties of decisions they are interested in. For instance, when inspecting automated loan decisions, a person might

ask "Why were no residents of neighbourhood A' selected to receive a loan?", even though the algorithm may not have used any representation of neighborhood A' in its decision. An explanation that answers this question not only provides example-based information, but also establishes a relationship between the initial decision and the explainee-defined concept in the question. Like the model reconciliation and concept activation approaches, a human introduces a new concept that the AI must incorporate into its knowledge representation and respond to in an explanation.

2.2.4 Summary

This section reviewed several approaches to explainability in AI, based on the distinction between transparency and post-hoc explainability. Some simple algorithms, such as sparse decision trees, can be simulated by people from start to finish, but most modern automated decision systems are not transparent in this way. Though they may not be simulatable, some algorithms are decomposable, meaning that they can be split into understandeable steps and components. By definition, this kind of transparency is exhibited to some extent by any decision mechanism which reasons with human-understandable symbols, and can typically be attributed to discrete optimization algorithms which often use meaningful variables, parameters, objectives, and constraints. However, when the intrinsic transparency of an algorithm is insufficient, post-hoc methods may help meet the demands of explainability. Used in addition to the real algorithm, a proxy relation between inputs and outputs may increase transparency by providing auxiliary information about the decision mechanism. Alternatively, post-hoc explanations may provide example-based information which does not rely on knowledge about how decisions are made. Finally, explanations may attempt to establish explicit relationships between human beliefs and concepts and an AI's decisions or decision-making process.

2.3 Counterfactual Explanations

This section focuses on the form and benefits of counterfactual explanations, including why they were selected for this thesis, and reviews recent research on this technique in ML.

2.3.1 The Form of Counterfactual Explanations

Given some initial inputs g' and an initial output p from a decision algorithm, a counterfactual explanation responds to a *contrastive question* of the form: "Why p, and not some other decision

$q \in Q$?" [8, 55]. Here, Q is a set of alternative decisions, called a *foil set*. Each hypothetical decision $q \in Q$ is called a *foil*, and, by definition, $p \notin Q$. For instance, a customer may ask, "Why was my shipment not scheduled to arrive before next month?", in which case the foil set Q would consist of all possible decisions where the customer's shipment is scheduled next month. Given a contrastive question, a counterfactual explanation presents the explainee with a set of alternative inputs h' which would have led to the decision being in Q. Typically, an explanation aims to find alternative inputs h' that are associated with the minimal modification of the initial inputs g'. In our shipping example, such an explanation might be the statement: "Your delivery would have arrived next month if you had paid at least $\$4,000$ more." *Counterfactual* means "contrary to the facts", so these explanations are called counterfactual explanations because the hypothetical inputs h' and hypothetical decisions $q \in Q$ are contrary to the actual inputs g' and decision p, respectively.

2.3.2 Benefits of Counterfactual Explanations

As an explanation method, counterfactual explanations have several desirable properties, including that they do not require an understanding of the decision mechanism, address the motivations of XAI, and support human counterfactual reasoning. Wachter *et al.* [8] argue that because counterfactual explanations do not require a person to comprehend how the algorithm works, they are better suited to general users who may have no knowledge of algorithmic decision-making. Further, counterfactual explanations can empower an explainee to understand, contest, or act to change a decision [8]. Regarding understanding, *counterfactual reasoning* has been demonstrated to be a fundamental human reasoning strategy, functioning by invoking a comparison between an imagined event and a factual event [56]. Counterfactual explanations help facilitate this type of human reasoning across complex AI systems, thereby also addressing the general motivation of better integrating human and AI decision-making. To help contest a decision, a person may receive an explanation showing that a counterfactual relationship between the algorithm's inputs and outputs is unacceptable according to their beliefs. For instance, a rejected job applicant may discover that she would have been offered a job if she was male instead of female. This kind of information can empower explainees, such as our job applicant, to contest automated decisions. Finally, counterfactual explanations can enable people to act to change the inputs in a way that will result in a different decision. For instance, in the delivery example above, the customer may be able to pay $4,000 to expedite their shipment.

From his review of social science research on explanations, Miller [55] concludes that good explanations should be *contrastive, social* and *selective*, which are all characteristics of counterfactual

explanations. Contrastive explanations address not why event p occurred, but why it occurred instead of event q. Counterfactual explanations are naturally contrastive due to their reliance on contrastive questions. As discussed in Section 2.2.3, *social* explanations account for the relationship between the beliefs of the explainer and the beliefs of the explainee. Counterfactual explanations are able to reflect the beliefs of the explainee by allowing them to define the foil set Q. Finally, counterfactual explanations are selective: out of the many alternative explanations that are possible, these methods are able select a single explanation, usually based on the minimal change to the initial inputs. These benefits of counterfactual explanations have led to the development of a sizeable literature on counterfactual explanations in XAI [9].

One of the goals of this thesis is to adapt an explainability technique from the general XAI literature and apply it to discrete optimization. Of the many approaches outlined in Section 2.2, counterfactual explanations are particularly well-suited for this purpose because they provide example-based information which is agnostic of the decision mechanism. Thus, while the content of an explanation may change if the problem being solved changes, such as if a scheduling decision is explained instead of an image classification decision, counterfactual explanations will not change because of differences in decision algorithms. This property removes a level of difficulty in transferring research between areas of AI which use very different decision algorithms, such as the areas of ML and discrete optimization. Though part of the contribution of this thesis is to adapt counterfactual explanations to the structure of discrete optimization problems, there is a rapidly growing amount of research on the general desiderata of counterfactual explanations such as diversity [8, 57, 58] and actionability [59, 60] which may be readily transferable to discrete optimization in the future without much modification (see Chapter 7).

2.3.3 Counterfactual Explanations in Machine Learning

This section focuses on recent research on counterfactual explanations in supervised learning, looking briefly at strategies for computing explanations, as well as some proposed desiderata of explanations such as sparsity, diversity, actionability, and causality.

Computation

Counterfactual explanations for supervised learning are introduced in Wachter *et al.*'s seminal work [8], which discusses some of the above benefits and presents a method for computing counterfactual explanations for differentiable models. Given some prediction function $h : \mathcal{X} \longrightarrow \mathcal{Y}$, an initial

input $\dot{x} \in \mathcal{X}$, and an initial decision $\dot{y} \in \mathcal{Y}$, a counterfactual explanation finds a minimally modified counterfactual input $\ddot{x}$ such that $h(\ddot{x})$ results in a desired counterfactual prediction $\ddot{y}$. The authors generate explanations by solving an optimization problem which minimizes the loss $\mathcal{L}^{\dot{y}}(\cdot,\cdot)$ of the counterfactual prediction $h(\ddot{x})$ being different from the desired prediction $\ddot{y}$ together with the loss $\mathcal{L}^{\dot{x}}(\cdot,\cdot)$ due to the counterfactual input $\ddot{x}$ differing from the initial input $\dot{x}$:

$$\arg\min_{\ddot{x}\in\mathcal{X}} \mathcal{L}^{\dot{y}}(h(\ddot{x}),\ddot{y}) + \mathcal{L}^{\dot{x}}(\ddot{x},\dot{x}). \tag{2.1}$$

If this optimization problem, which may, or may not be, convex [61], is differentiable, it can be solved with a gradient-based method such as stochastic gradient descent, and if it is not differentiable, with a gradient-free method such as Downhill-Simplex [61]. Looveron and Klaise [62] improve Wacther *et al.*'s initial optimization objective by adding terms to keep the generated counterfactuals on the data manifold of $\mathcal{X}$ and speed up optimization. Russell [57] also proposes a method to keep computed counterfactuals coherent with the initial data distribution, which may include discrete inputs, suggesting a MIP formulation for cases when the underlying classifier is linear. In general, however, Artelt and Hammer [61] claim that, currently, the most efficient methods for computing counterfactual explanations are specific to the underlying prediction model, and provide a survey of these methods.

Sparsity

Wachter *et al.* [8] also propose that sparsity is a desiderata of counterfactual explanations because it is easier to understand a few large changes to inputs rather than many very small changes, and suggest that an L_1 norm be used as part of the distance metric between inputs $\mathcal{L}^{\dot{x}}(\cdot,\cdot)$ to induce sparsity. Mothilal *et al.* [58] suggest a post-hoc sparsity enhancement process which greedily restores individual features in the counterfactual input $\ddot{x}$ to their initial values as long as the prediction $h(\ddot{x})$ does not change. In another approach, Le *et al.* [63] add a term to (2.1) which constrains the maximum number of features in $\ddot{x}$ that can change.

Diversity

Another goal of counterfactual explanations proposed by Wachter *et al.* [8] is that, sometimes, they should produce a diverse set of explanations. Wachter *et al.* suggest that when (2.1) is non-convex, local minima encountered during optimization can be used to create this diverse set. In his MIP formulation for linear classifiers, after a counterfactual explanation is generated, Russel [57] adds a

constraint preventing any subsequent explanation from returning the same result. Mothilal *et al.* [58] modify the optimization problem (2.1) to search for a set of explanations instead of a single one, aggregating the losses over this set and adding a loss term to represent the diversity of the set using a determinantal point process.

Actionability

A third desiderata for counterfactual explanations is that, in many contexts, they should be actionable; that is, explainees should be able to take steps to modify the inputs as described by the counterfactual explanation. In reality, some input features can be changed through action, while others cannot. An explanation that recommends actionable steps such as "If you increase your income by $10,000/year, you will get a loan," is usually more useful than one that suggests impossible actions such as "If you become 5 years younger and change your country of birth to Canada, you will get a loan." For this reason, Ustun *et al.* [59] suggest a formulation in which there is a set of actions representing feasible changes to the initial input $\dot{x}$, and each action is associated with some cost. They then reformulate the optimization problem (2.1) to find an action with minimal cost that results in the desired counterfactual prediction $\ddot{y}$. Karimi *et al.* [60] extend this work, introducing a causal model which captures more general relationships between the costs of actions and the resulting changes to input features.

Causality

Karimi *et al.*'s work [60] is also an example of research that addresses a fourth goal of counterfactual explanations, namely, that changes to inputs should respect some causal structure. For instance, the explanation "If you start and complete a four-year degree, while your age remains the same, you will receive a loan," does not make any sense. Mothilal *et al.* [58] propose a post-hoc filtering approach which filters a set of counterfactual explanations to remove those which do not respect known causal relations. Mahajan *et al.* [64] suggest a more elaborate approach in which a partial causal model can be used to define the loss term $\mathcal{L}^{\dot{x}}(\cdot,\cdot)$ based the degree to which causal relations to the initial input $\dot{x}$ are respected by the counterfactual input $\ddot{x}$. They also suggest a second approach for cases when causal models are not available *a priori,* in which users give feedback on the feasibility of proposed counterfactual explanations and a variational autoencoder is trained to generate new explanations that are consistent with user feedback.

Summary

Many researchers have recently pursued counterfactual explanations in ML, leading to the emergence of a literature which has quickly moved beyond basic applications of counterfactual explanations to formulations incorporating more advanced notions such as sparsity, diversity, actionability, and causality. This thesis aims to take a first step towards the development of a similar literature on counterfactual explanations in discrete optimization.

2.4 Contrastive Explanations in AI planning

Before turning to discrete optimization, we briefly review recent work on contrastive explanations in AI planning, which, like discrete optimization, relies on declarative models. As mentioned in Section 2.3, contrastive explanations answer contrastive questions of the form "Why p and not q?", where q is an alternative outcome to p. Counterfactual explanations are a subset of contrastive explanations,[3] answering these questions by finding the counterfactual inputs which would have made q occur instead of p. However, not all contrastive explanations rely on presenting the user with such counterfactual inputs [65]. For instance, in a planning setting, given the question "Why the initial plan π and not a different plan π'?", a contrastive explanation may compare the two plans and show that π' has a higher cost.

This latter form of explanation is explored in recent AI planning work [24, 25, 26] in which a contrastive plan π' is generated based on some user-specified properties and then compared to the initial plan π to show that π' is worse, no better, or impossible. Borgo *et al.* [24] compute a contrastive plan by allowing a user to select a set of actions in an initial plan and replace each action with a suggested one. Other research [25, 26] generates contrastive plans for temporal and numeric planning, facilitating user requests to include or exclude actions at certain states or times, or to change the order of actions in the initial plan.

Eifler *et al.* [27, 28] build on this research, computing not a single contrastive plan, but the properties shared by all plans that incorporate a user suggestion. Specifically, they formulate user suggestions and plan properties using soft goals and oversubscription planning [66], a setting in which it may be impossible to satisfy all soft goals. Then, the explanation for a user asking why all the soft goals $\hat{g} \in A$ were not achieved is given by one or more sets of goals $B_i \in B$, in which

[3] This terminology is slightly confusing because counterfactuals are actually used by both contrastive and counterfactual explanations. Both explanations involve a counterfactual outcome q, however counterfactual explanations rely on *a second* counterfactual, in the form of alternative inputs which would have led to q, to provide the explanation. Contrastive explanations do not necessarily use this second counterfactual as the explanation method.

the goals $\hat{g} \in B_i$ cannot *all* be met if the user suggestion is satisfied; that is, one or more goals $\hat{g} \in B_i$ must be foregone for each set $B_i \in B$. For instance, in an oversubscription routing setting, the explanation to "Why did you not transport person p_4?" might be "Because then I would not be able to transport one of people p_1, p_2, or p_3." A user study [67] of this explanation method indicated that such explanations enable users to find better trade-offs between soft goals. This approach also has similarities to Minimal Unsatisfiable Set based methods in discrete optimization, which will be reviewed in Section 2.5.

Finally, recent work [68], done simultaneously with this thesis, has considered using counterfactual explanations to explain certain planning problems, such as shortest path planning, as optimization problems. A user is expected to provide a fully-defined alternative plan (e.g. a different route) and inverse optimization is then used to compute the minimal change to the cost coefficients (e.g. travel costs) in the optimization problem so that the user's plan becomes optimal. This method, as well as its differences with the methods proposed in this thesis, is explained in more detail in Section 2.6.

Thus, research on contrastive explanations has very recently made progress in AI planning. Because AI planning models and plans are often decomposable into human understandable elements, these explanations can benefit from using interpretable costs, actions, goals, or states to justify decisions and communicate with explainees. The next section discusses how the decomposability of constrained optimization and satisfaction frameworks also leads to similar benefits for explanation.

2.5 Explanations in Constrained Optimization & Satisfaction

This section reviews work on explanations in constrained discrete optimization and satisfaction. Like AI planning, the models and algorithms are typically decomposable into human understandable elements, such as variables, parameters, constraints, and objective functions, enabling explanations to provide people with meaningful information in terms of these components. We first review Squalli and Freduer's [42] use of inference traces, which provide explanations of small constraint satisfaction problems (CSPs) solvable with inference only. We then turn to infeasibility-based explanations, which have been the predominant focus of the explainability literature [13]. Finally, we review a method from Rossi and Sperduti [69] which elicits user preferences over soft constraints from user preferences over solutions, and shows that their problem formulation is very similar to that used by counterfactual explanation techniques.

2.5.1 Simulatability

As mentioned in Section 2.2.2, Sqalli and Freuder [42] argue that inference-only algorithms, applied to small CSP instances, are simulatable. Specifically, the authors use logic puzzles as a case study, and provide explainees with a trace of the algorithm's inference steps, presented using text templates. For example, in a puzzle where musicians are matched with songs and instruments, an explanation of a single inference step is *"Guitar player must be Love composer because Guitar player can't be any of the others."*. The authors argue that sequences of phrases such as these are almost identical to human explanations provided at the back of logic puzzle booklets.

However, this algorithm tracing approach has some disadvantages. Firstly, it cannot handle search. The authors acknowledge that a trace of search steps is likely much more difficult for a person to understand than a trace of inference steps, involving difficult-to-follow sequences of phrases such as "I tried to assign this value, but found there was no solution, so I tried a different value". Secondly, this method requires custom text templates for each inference type. Finally, once the problem instance size exceeds some threshold, it inevitably leads to traces that are too long for a person to follow from start to finish. Thus, most research has not followed an algorithm tracing approach, instead pursuing post-hoc explanation methods. However, recently, Bogearts *et al.* [20] and Gamba *et. al.* [21] have followed up on Sqalli and Freuder's work, combining their inference sequence based method with post-hoc infeasibility explanations, discussed further in Section 2.5.2.

2.5.2 Infeasibility Based Explanations

By far, most post-hoc approaches have relied on explanations of why a constrained optimization or satisfaction problem instance is infeasible [13]. Most methods identify sets of constraints that, together, lead to infeasibility. Recently, some researchers have also used these constraint sets to explain why any decision other than the decision made would be infeasible [20, 21, 22].

Finding Conflicting Constraint Sets

One such kind of constraint set is called a Minimal Unsatisfiable Set (MUS), also known as an Irreducible Inconsistent Subsystem in the MIP literature [23]. If C is an unsatisfiable constraint set, then $M \subseteq C$ is an MUS of C iff M is unsatisfiable and all $M' \subset M$ are satisfiable [14]. After being shown an MUS (or set of MUSes), a person can identify not only that *there is* a conflict among constraints, but also the subset(s) of constraints responsible for the conflict. Thus, an MUS-based explanation provides additional information about how the inputs (the constraints) lead to

the output (infeasibility), which fits the definition of a proxy relation.

Another type of constraint set used to explain infeasibility is called a Minimal Correction Subset (MCS). M is an MCS of C iff $C \setminus M$ is satisfiable, but, for any $M' \subset M$, $C \setminus M'$ is unsatisfiable [23]. Therefore, removing an MCS M from the initial constraint set C restores feasibility. Though they have not yet been been discussed as such in the literature, MCS-based explanations are counterfactual explanations, because they identify a minimal change to the constraint set C, in terms of constraints M to be removed, that would result in a feasible instance.

Various methods have focused on generating MUSes and MCSes effectively. These include approaches to: generate a single MUS [14, 15, 16], out of which QuickXPlain [14] is especially well-known; enumerate all MUSes, such as the MARCO [70] approach; or compute MCSes [17, 18, 19]. Additionally, due to the exponential number of MUSes and MCSes [23], some methods allow users to express preferences over constraints and then generate the sets containing the least preferred constraints [14, 71, 72]. Recently, Senthooran *et al.* [23] combined some of these computational approaches into a user-centred infeasibility explanation system.

Explaining Decisions Via Conflicting Constraint Sets

Researchers have recently adapted MUS-based methods to show that, given an initial decision, different decisions would be infeasible. One set of papers [20, 21] uses MUS-based methods for problems similar to the ones considered by Squali and Freuder's [42], namely, CSPs solvable with inference only and that have a unique solution. In this setting, each inference step i, which assigns a value v_i to a variable x_i, is explained by showing that, if the constraint $x_i \neq v_i$ is added, the problem becomes infeasible. This explanation takes the form of an MUS which identifies which initial constraints and previous inferences, made before step i, conflict with the new constraint $x_i \neq v_i$. The authors also consider that some MUSes may be more interpretable than others, such as if they involve fewer or simpler constraints. Thus, though all sequences of inference steps will lead to the same solution in their setting, a sequence of inference steps which is different than the sequence used by the initial solver may result in more interpretable MUSes. Therefore, the authors define an interpretability function which is assumed to measure how interpretable an MUS is, for instance by using the cardinality of the MUS, and search for the sequence of steps that results in the most interpretable sequence of MUSes according to this function. As with Squali and Freuder's work, this approach can, so far, handle only inference and not search.

In a different paper, Yelamanchili *et al.* [22] use MUSes to explain why it was not possible to include a job in a schedule built with a greedy algorithm. Specifically, they first create an initial

schedule using a greedy Squeaky Wheel Optimization [73] algorithm. At a given step i in this greedy algorithm, it may not be possible to schedule job j given the partial schedule $\mathcal{S}'$ of jobs scheduled before step i, and since no backtracking happens in the algorithm, job j is permanently excluded from the schedule if not scheduled at step i. To explain why job j was not scheduled by the algorithm, the authors compute an MUS from the initial constraints and constraints that j must be in the schedule and that the partial schedule $\mathcal{S}'$ must be respected. The authors argue that, after receiving such an explanation, an explainee may be able to change the parameters of the jobs in $\mathcal{S}'$, such as their resource consumption or processing times, to remove the conflict at step i and thus include job j in the schedule.

2.5.3 Eliciting Constraint Preferences from Solution Preferences

The last method we review is Rossi and Sperduti's [69] approach for acquiring user preferences over a set of soft constraints by allowing users to specify preferences over a set of solutions. It will be shown that, if this method were to be slightly repurposed, it would be a counterfactual explanation technique. In their system, the authors first solve a fuzzy CSP [74] that uses *soft constraints* that may be mutually unsatisfiable, but are associated with user preferences about which constraints are more important to satisfy when it is not possible to satisfy them all. Their system then outputs a set of solutions, organized in descending order of the degree to which the solutions respect the preferences over the soft constraints. For instance, a person searching for a new home on a listing site may indicate the importance of soft constraints such as a minimum square footage, minimum number of bathrooms, and maximum distance from city centre, and then be shown a set of five listings in the order of how well they satisfy their preferences over the constraints.

However, a person may find it easier to express their preferences over the solutions rather than preferences over constraints. Thus, the authors design an iterative system where a user can specify not only their preferences over constraints directly, but where constraint preferences can also be learned after a user specifies preferences over the solutions that are shown to them. For instance, a user looking at a set of three houses given equal preference may say "I like house h_1 better than house h_2, and both better than house h_3". To learn constraint preferences from user feedback that specifies a different preference order for the solutions than the order suggested by the system, a reinforcement learning approach is used. Specifically, the reward function is the degree of agreement between the preferred order of solutions computed by the system and the order acquired through user feedback, the states are sets of constraint preferences, and actions represent changes to those

constraint preferences.

Rossi and Sperduti design this system with the goal of developing a new method to elicit a human's preferences over soft constraints in a fuzzy CSP. However, if the goal was instead to answer the question "Why were the solutions ordered in the sequence a^* and not in an alternative sequence a^ψ?", their method would constitute a counterfactual explanation technique. Specifically, their approach would generate the counterfactual constraint preferences that would have resulted in a^ψ instead of a^*, thereby producing a counterfactual explanation in terms of constraint preferences.

2.5.4 Summary

Roughly five years ago, explainability began attracting significant attention in ML and AI planning. However, many of the works cited in this section [14, 15, 16, 42, 69, 71, 72] date back to well before the start of this period. Thus, Freuder [13] has recently called for a renewed focus on explainability in discrete optimization. Some of the above work [20, 21, 22, 23] represents recent responses to this call. However, all of these new methods rely on existing definitions of explanations as conflicting sets of constraints. In this thesis, a new approach is pursued using counterfactual explanations based on changes to objective parameters for discrete optimization problems. This methodology uses a variant of inverse combinatorial optimization which aims to find minimal changes to the objective parameters of a discrete optimization problem leading to a counterfactual decision.

2.6 Inverse Optimization

This section describes inverse optimization [75] and two of its variants [30, 76], reviewing relevant problem definitions, solution approaches, and Brandao and Magazzeni's [77] work on inverse optimization for plan explanations.

2.6.1 Problem Definitions

Inverse Optimization

In a standard (or *forward*) optimization problem $\mathcal{FW}\langle c, f, X\rangle$, the purpose is to find values for decision vector $x \in X \subseteq \mathbb{R}^n$ given a parameter vector $c \in \mathcal{D} \subseteq \mathbb{R}^s$ which optimize an objective function $f : \mathcal{D} \times X \to \mathbb{R}$. We assume a minimization problem unless otherwise stated, but if specification is needed, $\mathcal{FW}_{\min}\langle c, f, X\rangle$ denotes a minimization problem while $\mathcal{FW}_{\max}\langle c, f, X\rangle$ denotes a maximization problem. This assumption and notation holds for all subsequent types of forward problems. 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 f is omitted from a problem definition, it is assumed that $f(c, x) = c^T x$.

While forward optimization seeks a variable assignment that satisfies a set of constraints and optimizes an objective function, (classical) 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 $\mathcal{FW}\langle c, f, X\rangle$, and a feasible target 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. If $d \in \mathcal{D}$ is the modified parameter vector and $||\cdot||$ is some norm, typically L_1 or L_∞, then the *inverse optimization* problem $\mathcal{IO}\langle c, \mathcal{D}, f, x^d, X, ||\cdot||\rangle$ [78] is

$$\min_{d\in\mathcal{D}} ||d - c|| \tag{2.2}$$

$$\text{s.t. } f(d, x^d) = \min_{x\in X} f(d, x). \tag{2.3}$$

Assignment-Based Partial Inverse Optimization

The first variant of inverse optimization we review is one where, instead of being given a complete target solution $x^d \in X$, we are given a vector of values $x^p \in \mathbb{R}^m$, $m = |\mathcal{M}|$, $\mathcal{M} \subseteq \{1, ..., n\}$, which must be taken on by the optimal solution. We call this the *assignment-based partial inverse optimization* problem, $\mathcal{APIO}\langle c, \mathcal{D}, x^p, X, ||\cdot||\rangle$ [78], given by

$$\min_{d,x} ||d - c|| \tag{2.4}$$

$$\text{s.t. } f(d, x) = \min_{\hat{x}\in X} f(d, \hat{x}) \tag{2.5}$$

$$x_i = x_i^p \qquad \forall\ i \in \mathcal{M} \subseteq \{1, ..., n\} \tag{2.6}$$

$$d \in \mathcal{D},\ x \in X. \tag{2.7}$$

In most of the inverse optimization literature, the above problem is simply called *partial inverse optimization* [30]. However, Wang [76] uses the term partial inverse optimization to describe a more general problem, in which the target solution is partially specified with a set of linear constraints which is not necessarily a set of assignments. To differentiate between the two problems, we refer to (2.4) - (2.7) as *assignment-based* partial inverse optimization, and Wang's [76] formulation, reviewed next, as *constraint-based* partial inverse optimization.

Constraint-Based Partial Inverse Optimization

Wang [76] defines a variation of inverse optimization in which, instead of a target solution x^d, we are given an additional set of linear constraints $Lx \leq l$, $L \in \mathbb{R}^{m \times n}, l \in \mathbb{R}^m$, that the optimal solution to the modified forward problem $\mathcal{FW}\langle d, X\rangle$ must satisfy. We call this the constraint-based partial inverse optimization problem,[4] $\mathcal{CPIO}\langle c, \mathcal{D}, l, L, X, ||\cdot||\rangle$, given by

$$\min_{d,x} ||d - c|| \tag{2.8}$$

$$\text{s.t. } f(d, x) = \min_{\hat{x} \in X} f(d, \hat{x}) \tag{2.9}$$

$$Lx \leq l \tag{2.10}$$

$$d \in \mathcal{D},\ x \in X. \tag{2.11}$$

This formulation is a generalization of both inverse optimization (2.2)-(2.3) and assignment-based partial inverse optimization (2.4) - (2.7), since both full and partial assignments can be represented by the linear constraint set (2.10) [76].

2.6.2 Solution Methods

Most research on inverse optimization has been in contexts when the forward problem is convex, such as a linear program (LP) [78, 82, 83] or a conic optimization problem [84]. These approaches use the Karush-Kuhn-Tucker (KKT) conditions to express the bilevel inverse problem as a single level convex problem [85].

Work on solving *inverse combinatorial optimization* [75] problems, which include discrete decision variables, has been limited [78]. One set of papers [86, 87], focusing on mixed integer linear programming (MILP) as the forward problem, reformulates the bilevel inverse problem into a single level linear program using superadditive duality; however, the resulting problem is exponentially large. An alternative solution approach by Wang [10] relies on an iterative, two-level cutting plane method, and forms the basis of some of the algorithms in this thesis.

[4] Likely due to this ambiguity in nomenclature, none of the works [30, 79, 80] citing Wang [76] discuss his constraint-based generalization (2.8) - (2.11) of inverse optimization, instead addressing only his contribution to assignment-based partial inverse optimization. The authors of the works [1, 81] published as a result of this thesis, which rely on a formulation of partial inverse optimization similar to Wang's [76], were not aware of Wang's formulation or his solution methods for it until February 2022.

Cutting Plane Algorithm for Inverse MILP

In Wang's algorithm [10], we are given a forward $\mathcal{MILP}\langle c, X\rangle$, which is a forward problem 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 target solution $x^d \in X$, and a feasible set $\mathcal{D}$ for parameter values, leading to the inverse problem $\mathcal{IO}\langle c, \mathcal{D}, x^d, X\rangle$. The solution algorithm is a two-level iterative approach in which an LP master problem $\mathcal{MP}_{\mathcal{LP}}$ (2.13) - (2.16) is initiated with a set of known solutions $\mathcal{S}^0$. $\mathcal{MP}_{\mathcal{LP}}$ is then used to search for a new objective vector d minimizing $||d - c||_1$ so that, for the modified forward problem $\mathcal{FW}\langle d, X\rangle$, x^d gives a solution no worse than any solution in $\mathcal{S}^0$. Given d^i, an optimal solution to the $\mathcal{MP}_{\mathcal{LP}}$ in a given iteration, a subproblem

$$\mathcal{SP}_{\mathcal{MILP}}\langle d^i, X\rangle = \mathcal{FW}\langle d^i, X\rangle, \tag{2.12}$$

is then solved to optimality to give x^0, an extreme point of $conv(X)$, the convex hull of X. If x^0 is no better a solution to $\mathcal{SP}_{\mathcal{MILP}}\langle d^i, X\rangle$ than x^d, then x^d must be optimal to $\mathcal{FW}\langle d^i, X\rangle$, and d^i constitutes an optimal solution to the inverse problem [10]. Otherwise, the extreme point x^0 is added to the set of known solutions $\mathcal{S}^0$ to generate a new cut, and the algorithm proceeds to the next iteration of the master problem.

To formulate the master problem $\mathcal{MP}_{\mathcal{LP}}$ (2.13) - (2.16), the objective $||d - c||_1$ is first linearized using $g, h \in \mathbb{R}^n_+$, such that $c - d = g - h$. Intuitively, the magnitude of the change to the parameter c_j is represented by g_j if the change is negative and h_j if it is positive. To avoid any d for which the forward problem is unbounded, Wang introduces the decision variable $u \in \mathbb{R}^k$ and add the constraint $A^T u \geq d$ (2.14), ensuring that d results in a feasible dual problem [10]. Finally, constraints (2.15) force x^d to be at least as good as any known solution in $\mathcal{S}^0$ in terms of solution quality for $\mathcal{FW}\langle d, X\rangle$. Thus, $\mathcal{MP}_{\mathcal{LP}}$ is given by

$$\min_{u,g,h} \; g + h \tag{2.13}$$

$$\text{s.t } A^T u \geq c - g + h \tag{2.14}$$

$$(c - g + h)^T x^d \leq (c - g + h)^T x^0 \qquad \forall x^0 \in \mathcal{S}^0 \tag{2.15}$$

$$g, h \in \mathbb{R}^n, \quad u \in \mathbb{R}^k, \quad (c - g + h) \in \mathcal{D}. \tag{2.16}$$

$\mathcal{MP}_{\mathcal{LP}}$ (2.13) - (2.16) is an LP, while the subproblem $\mathcal{SP}_{\mathcal{MILP}}\langle d, X\rangle$ is a MILP. Thus, Wang's algorithm will be called *InvLP-MILP* [10], with the complete definition given by Algorithm 2.1.

Algorithm 2.1: *InvLP-MILP*[10].

Inputs: $\mathcal{IO}\langle c, \mathcal{D}, x^d, X\rangle$
Output: d^*
Step 1: Initialize $\mathcal{S}^0 \leftarrow \emptyset$
Step 2: Solve $\mathcal{MP}_{\mathcal{LP}}$.
    If infeasible, return *INFEASIBLE*.
    Otherwise, get $d^i = (c - g^i + h^i)$.
Step 3: Solve $\mathcal{SP}_{\mathcal{MILP}}$ $\langle d^i, X\rangle$ to get optimal solution x^0.
    If $d^{i,T}x^d \leq d^{i,T}x^0$, stop. $d^i = d^*$ is optimal to $\mathcal{IO}\langle c, \mathcal{D}, x^d, X\rangle$.
    Otherwise, update $\mathcal{S}^0 = \mathcal{S}^0 \cup \{x^0\}$ and return to Step 2.

Improvements to *InvLP-MILP* [10]

Two subsequent works [85, 88] improve on *InvLP-MILP*. Duan and Wang [88] extend Wang's algorithm with a heuristic to parallelize cut generation. This heuristic also iteratively computes feasible solutions as upper bounds for the inverse MILP, addressing the initial algorithm's limitation that no feasible solutions are found until termination. Recently Bodur *et al.* [85] demonstrated several improvements to *InvLP-MILP* based on a theoretical analysis of the feasible region of the inverse MILP problem. Specifically, they showed that it is not necessary to use the extreme points of $conv(X)$ to build $\mathcal{S}^0$ for the algorithm to converge. In particular, they demonstrated that all the necessary cuts for convergence can instead be generated using a relatively small set of interior points, motivating the use of trust regions around x^d to limit the subproblem's search set to subsets of the interior region of X. This modification allows the algorithm to find cuts more quickly and also use fewer cuts overall. The above improvements from Duan and Wang [88] and Bodur *et al.* [85] are not implemented in this thesis, but could be pursued in future work, as discussed in Chapter 7.

This thesis does implement another computational improvement to Wang's algorithm [10] identified by Bodur *et al.* [85]: an early stopping criterion (ESC) which can reduce the amount of time in the subproblem $\mathcal{SP}_{\mathcal{MILP}}$ after a valid cut (i.e. a point to be added to $\mathcal{S}^0$) has been found. In the original algorithm [10], $\mathcal{SP}_{\mathcal{MILP}}$ is solved to optimality at every iteration. However, if a feasible, but not necessarily optimal, solution x^f has been found to $\mathcal{SP}_{\mathcal{MILP}}$ which gives a better objective value than x^d, $d^{*T}x^f < d^{*T}x^d$, then a valid cut can be generated by adding x^f to $\mathcal{S}^0$. After such an x^f has been found, it may not be worth spending additional time searching for a better solution to $\mathcal{SP}_{\mathcal{MILP}}$ because there is no guarantee this will lead to a stronger cut, and x^f may already be the optimal solution, just not yet proven optimal [85]. For this reason, if a valid cut has been generated, the time spent in an iteration of the subproblem $\mathcal{SP}_{\mathcal{MILP}}$ is restricted to a time limit γ. The algorithm modified with the ESC is called *InvLP-MILP*(ESC) and defined in Algorithm 2.2.

Algorithm 2.2: *InvLP-MILP*(ESC)[85].

```
Inputs: c, D, x^d, X, γ
Output: d*
Step 1: Initialize S^0 ← ∅
Step 2: Solve MP_LCP to get optimal solution d^i
Step 3: while TRUE:
        get next feasible solution x^k to SP_MILCP ⟨d^i, X⟩
        if d^{i,T} x^k ≤ d^{i,T} x^d:
            if x^k optimal:
                if d^{i,T} x^k == d^{i,T} x^d:
                    Stop. d^i = d* is optimal to the inverse problem.
                else:
                    Update S^0 ← S^0 ∪ {x^k}
                    go to step 2
            else:
                if d^{i,T} x^k < d^{i,T} x^d:
                    if time since end of last Step 2 iteration ≥ γ:
                        Update S^0 ← S^0 ∪ {x^k}
                        go to step 2
```

Constraint-Based Partial Inverse Optimization Problems

Wang [76] is the only author to publish a general solution method for discrete partial inverse optimization problems, addressing the case when the forward problem is a MILP. In this paper, Wang solves the constraint-based variant of partial inverse optimization (2.8)-(2.11) by modifying *InvLP-MILP* to be used in a branch-and-cut framework. Wang's partial inverse optimization algorithm $B\&C_{CPIO}$ (Algorithm 2.3) [76] is not implemented in this thesis, but is reviewed for completeness.

Given a forward $\mathcal{MILP}\langle c, X\rangle$ and additional linear constraint set $Lx \leq l$, we obtain an instance of $\mathcal{CPIO}\langle c, \mathcal{D}, l, L, X\rangle$ (2.8)-(2.11). If we linearize the L_1 objective as in *InvLP-MILP*, and let $\mathcal{S}$ be the set of extreme points of $conv(X)$, we can reformulate this problem as

$$\min_{g,h,x,u} \; g + h \tag{2.17}$$

$$\text{s.t } A^T u \geq c - g + h \tag{2.18}$$

$$Ax \leq b \tag{2.19}$$

$$Lx \leq l \tag{2.20}$$

$$(c - g + h)^T x \leq (c - g + h)^T x^0 \qquad \forall x^0 \in \mathcal{S} \tag{2.21}$$

$$g, h, x, u \geq 0 \tag{2.22}$$

$$x_I \in \mathbb{N}_0^+. \tag{2.23}$$

Here, constraints (2.18) ensure d does not result in a problem with an unbounded objective, and are equivalent to constraints (2.14).

Next, we define the continuous relaxation of (2.17) - (2.23) and discuss how this relaxation is used in a branch-and-cut framework. This continuous relaxation, in which the integrality constraint (2.23) is removed and constraints (2.19) - (2.20) are replaced by (2.26), is given by

$$\min_{g,h,x,u} \; g+h \tag{2.24}$$

$$\text{s.t } A^T u \geq c - g + h \tag{2.25}$$

$$x \in \arg\min_{\tilde{x}} \{(c-g+h)^T \tilde{x} : A\tilde{x} \leq b, L\tilde{x} \leq l, \tilde{x} \geq 0)\} \tag{2.26}$$

$$(c-g+h)^T x \leq (c-g+h)^T x^0 \qquad \forall x^0 \in \mathcal{S} \tag{2.27}$$

$$g, h, x, u \geq 0. \tag{2.28}$$

This relaxation is a bi-level problem, and includes the non-linear, non-convex constraints (2.27). However, (2.24) - (2.28) can be reformulated using complimentary slackness and strong duality as:

$$\min_{g,h,x,u,y,z} \; g+h \tag{2.29}$$

$$\text{s.t } A^T u \geq c - g + h \tag{2.30}$$

$$0 \leq b - Ax \perp y \geq 0 \tag{2.31}$$

$$0 \leq l - Lx \perp z \geq 0 \tag{2.32}$$

$$0 \leq x \perp A^T y + L^T z - c - h + g \geq 0 \tag{2.33}$$

$$b^T y + l^T z \leq (c-g+h)^T x^0 \qquad \forall x^0 \in \mathcal{S} \tag{2.34}$$

$$g, h, x, u, y, z \geq 0. \tag{2.35}$$

In this reformulation, complementary slackness is first used to rewrite constraint (2.26) as constraints (2.31) - (2.33), giving a single level problem. Here $\perp$ denotes that two vectors of the same size are complementary: $a \perp b \Leftrightarrow a^T b = 0$. Then, strong duality gives $(c-g+h)^T x = b^T y + l^T z$, $y \in \mathbb{R}^n, z \in \mathbb{R}^p$, allowing the bilinear LHS of constraints (2.27) to be linearized as $b^T y + l^T z$ in (2.34). Formulation (2.29) - (2.35) is a linear program with complementarity constraints (LPCC), which can be reformulated and solved as a MILP with a big-M parameter [76, 89] or using a Benders decomposition approach [90, 91].

If $\mathcal{S}$, the set of extreme points of $conv(X)$, was known *a priori*, then it would be sufficient to

use the LPCC relaxation (2.29) - (2.35) in a branch-and-bound algorithm [76] to solve the discrete problem $\mathcal{CPIO}\langle c, \mathcal{D}, l, L, X\rangle$ [76]. Specifically, solving an LPCC relaxation (2.29) - (2.35) at each node of the search tree would be analogous to solving an LP relaxation at each node if branch-and-bound were used to solve a MILP.

However, since $\mathcal{S}$ is not known *a priori*, extreme points must instead be iteratively added to a set of known points $\mathcal{S}^0$ in a branch-and-cut process [92]. Specifically, the solution of an LPCC relaxation (2.29) - (2.35) at each node of the search tree is replaced with a subroutine SR_{CPI} (Definition 2.1), which iteratively solves the LPCC (2.29) - (2.35) with $\mathcal{S}^0$ replacing $\mathcal{S}$, and updates $\mathcal{S}^0$ until a branching decision needs to be made. The use of the subroutine SR_{CPI} is analogous to the use of an LP relaxation in a classical branch-and-bound procedure for solving MILPs. Wang's complete branch-and-cut algorithm [76] is given by Algorithm 2.3.

Definition 2.1. (SR_{CPI} [76]).

1. Solve LPCC (2.29) - (2.34) plus any constraints obtained through branching, with the set of known solutions $\mathcal{S}^0$ replacing $\mathcal{S}$. If infeasible, return "infeasible". Otherwise, get optimal solution $(g^i, h^i, x^i, u^i, y^i, z^i)$.

2. Solve $\mathcal{MILP}\langle(c - g^i + h^i), X\rangle$ to obtain x^0. If $(c - g^i + h^i)^T x^0 < (c - g^i + h^i)^T x^i$, update $\mathcal{S}^0 = \mathcal{S}^0 \cup x^0$ and return to step 1. Otherwise, return $(c - g^i + h^i, x^i)$.

Algorithm 2.3: $B\&C_{CPIO}$[76].

Inputs: $c, \mathcal{D}, l, L, X$
Output: d^*
Step 1: Initialize $\mathcal{S}^0 \leftarrow \emptyset$
Step 2: Set $d^* = null, v^* = \infty$
Step 3: Add LPCC (2.29) - (2.35), with $\mathcal{S}^0$ replacing $\mathcal{S}$, to list of problems $\mathcal{P}$
Step 4: While $\mathcal{P}$ not empty:
    1. Choose and remove an LPCC ^{i} from $\mathcal{P}$
    2. Solve the subroutine SR_{CPI} (Def. 2.1) with the selected problem
    3. If infeasible, return to Step 4. Otherwise, get solution (g, h, x, u, y, z), and set objective value to $v = g + h$
    4. If $v \geq v^*$, return to Step 4
    5. If x is integer, set $v^* \leftarrow v$, $d^* \leftarrow (c - g + h)$, and return to Step 4
    6. Branch on a variable $x_j \notin \mathbb{N}_0^+$, $j \in I$, adding two new problems to $\mathcal{P}$:
        1) LPCC ^{i} plus the constraint $x_j \leq \lfloor x_j \rfloor$
        2) LPCC ^{i} plus the constraint $x_j \geq \lceil x_j \rceil$

2.6.3 Inverse Optimization for Explanations

The last publication we review is work by Brandao *et al.*[5] [68] which uses classical inverse optimization to generate counterfactual explanations of path plans. Specifically, the authors focus on explaining a path planning problem in which, given a directed graph $G = (V, E, W)$ consisting of vertices $v'_i \in V$, directed edges $e'_j \in E$, and travel costs $w'_j \in W$ for each edge, the goal is to find the shortest path $\pi^* = (e'_1, ..., e'_n)$, $n < |V|$, from a start node v'_s to a goal node v'_g that minimizes $\sum_{k=1}^{n} w_k$. This forward problem has a linear programming formulation. After an initial shortest path π^* is found, an explainee is able to ask the contrastive question "Why is π^* the shortest path, and not some other path $\hat{\pi}$?". The authors solve a classical inverse optimization problem (2.2)-(2.3) with $\hat{\pi}$ as the target solution to find the minimally perturbed travel costs $\hat{W}$ that would have led to $\hat{\pi}$ being optimal. The counterfactual explanation is then: "$\hat{\pi}$ would have been the optimal path if $\hat{W}$ were the travel costs instead of W". Though the authors focus on explaining path plans, their techniques could be adapted to explain any problem that can be defined as an LP.

While this method is similar to the approach taken in this thesis, there are important differences. In the work of Brandao *et al.* [68], the explainee needs to specify a complete alternative solution $\hat{\pi}$, while in this thesis, the alternative solutions can be partially defined. Since the solution method of Brandao *et al.* [68], classical inverse optimization, cannot handle the more general case of partially defined solutions, an important contribution of this thesis is providing new methodologies to solve these problems.

2.7 Conclusion

This chapter reviewed past work relevant to the goal of developing a counterfactual explanation literature for discrete optimization. A high level outline of key concepts in XAI was provided and the counterfactual explanation literature in ML was reviewed. We then looked at the emerging AI planning literature on contrastive explanations and outlined past work on explanations for constrained optimization and satisfaction. Finally, we reviewed inverse combinatorial optimization.

[5] This work was published concurrently with the research done in this thesis.

Chapter 3

A General Counterfactual Explanation Problem

This chapter formulates the problem of generating a counterfactual explanation for an optimization problem using a variation of partial inverse optimization. Conditions for feasibility as well as assumptions and limitations are discussed.

3.1 Notation

Table 3.1 summarizes the main notation used in the main chapters of this thesis, relying in part on previous definitions in Section 2.6.

Table 3.1: General Notation.

Beginning of Table 3.1		
Symbol	**Definition**	**Section Defined**
x	Forward decision vector	2.6.1
c	Initial forward objective vector	2.6.1
f	Forward objective function	2.6.1
X	Forward feasible set	2.6.1
x^*	Optimal solution to initial forward problem	2.6.1
n	Number of forward decision variables	2.6.1
x^d	Target solution for inverse problem	2.6.1

Continuation of Table 3.1

Symbol	Definition	Section defined
d	Modified objective vector	2.6.1
$\mathcal{D}$	Feasible set of modified objective vectors	2.6.1
d^*	Optimal solution to an inverse or an explanation problem	2.6.1
X_ψ	Foil set	3.2
X_ψ^C	Non-foil set	3.2
ψ	Feasible set described by foil constraints	3.2
x^p	Vector of partial value assignments	2.6.1
m	Number of auxiliary/foil constraints	2.6.1
$\mathcal{M}$	Subset of decision variables involved in question	2.6.1
g	Vector of negative changes to c	2.6.2
h	Vector of positive changes to c	2.6.2
$\mathcal{S}^0$	Set of known non-target solutions or non-foils	2.6.2
γ	Early stopping criterion time limit	2.6.2
m_0	Min./Max. decision variable value for univariate explanations	4
$x^{\psi,*}$	Optimal foil for PA-NCE (Def. 5.1) problems	5.2

3.2 Nearest Counterfactual Explanations

We wish to generate a counterfactual explanation for an optimal solution x^* to a forward problem $\mathcal{FW}\langle c, f, X\rangle$. 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 $x \in X_\psi$?". As per the terminology from Section 2.3.1, X_ψ is called the foil set and each solution $x \in X_\psi$ is called a foil. To define the foil set, the explainee must provide an additional set of constraints describing a feasible set $\psi \subset \mathbb{R}^n$, with $x^* \notin \psi$. We call these additional constraints *foil constraints,* and define the foil set as $X_\psi = X \cap \psi$.

Example 3.1. (Contrastive Question). Consider a delivery scheduling problem $\mathcal{FW}\langle c, f, X\rangle$ where the variables $x \in X \subseteq \mathbb{N}_0^n$ represent the number of days before a shipment arrives, the objective coefficients $c \in \mathcal{D} \in \mathbb{N}_0^n$ are delivery priorities, and $f = c^T x$. That is, the problem is to minimize priority weighted delivery times. After seeing the optimal schedule x^*, a customer notices that for her order j, $x_j^* = 14$; that is, her delivery is scheduled in two weeks. She asks the contrastive question "Why is my order scheduled in two weeks, and not sometime next week?", a question that

can be rewritten as "Why x^* and not a solution $x \in X_\psi$?", where $X_\psi = \{x \in X : x_j \leq 7\}$. This foil set X_ψ is defined by the foil constraint $x_j \leq 7$ and $\psi = \{x \in \mathbb{N}_0^n : x_j \leq 7\}$.

We address this type of question by looking for counterfactual objective parameters $d \in \mathcal{D} \subseteq \mathbb{R}^n$ that would lead to one of the foils $x \in X_\psi$ being optimal to the modified problem $\mathcal{FW}\langle d, f, X\rangle$.[1] Here, $\mathcal{D}$ can express restrictions on the counterfactual objective parameters. Additionally, we aim to find a d which is minimally different from the initial parameters c as measured by some norm $||\cdot||$. If such a d is found, we can provide the counterfactual explanation: "A solution $x \in X_\psi$ would have been optimal if the objective parameters had been d instead of c." Formally, we define the Nearest Counterfactual Explanation (NCE) problem $\mathcal{NCE}\langle c, \mathcal{D}, f, \psi, x^*, X, ||\cdot||\rangle$ as

$$\min_{d \in \mathcal{D}} ||d - c|| \tag{3.1}$$

$$\text{s.t.} \quad \min_{x \in X_\psi} f(d, x) = \min_{x \in X} f(d, x). \tag{3.2}$$

If f and $||\cdot||$ are omitted, it is assumed that $f(d, x) = d^T x$ and $||\cdot||$ is L_1. Additionally, let the set of non-foils be denoted $X_\psi^C = X \setminus X_\psi$, a definition that will become useful in subsequent chapters. An NCE can also be formulated for maximization forward problems, in which case the minimization terms in constraint (3.2) are replaced with maximization terms. $\text{NCE}_{\min}$ and $\text{NCE}_{\max}$ can be used to specify whether the forward problem is a minimization or maximization problem, respectively, and if no direction is specified, minimization is assumed. This thesis focuses on NCEs based on discrete forward optimization problems with linear objectives and constraints.

Example 3.2. (Counterfactual Explanation). Continuing Example 3.1, assume that customer j initially paid \$30 for the lowest priority ("Standard") shipping option, and $c_j = 1$. To demonstrate how restrictions on counterfactual parameters can be useful, assume also that the customer is only interested in possible changes to their own order priority c_j, but not the order priorities of other customers. This can be expressed by setting $\mathcal{D} = \{d \in \mathbb{N}_0^n : d_i = c_i \ \forall \ i \neq j\}$. After solving the NCE (3.1) - (3.2), assume that the optimal solution is $d_j^* = 3$, $d_i^* = c_i \ \forall \ i \neq j$, where a shipping priority level of 3 corresponds to "Express Shipping" and costs \$100. The customer is given the counterfactual explanation: "Your delivery would have arrived within a week if you had paid \$100 for Express Shipping instead of \$30 for Standard Shipping."

[1] Note that the feasible set is X, not X_ψ, since the explainee wants to know why a foil was not optimal to the initial problem, not a restricted version of it.

3.3 Feasibility Conditions

Next, we can identify conditions under which an NCE is guaranteed to have a non-trivial feasible solution (i.e. $d \neq [0]^n$), considering both the general case and NCEs with linear objectives.

Theorem 3.1. An $\mathcal{NCE}\langle c, \mathcal{D}, f, \psi, x^*, X, ||\cdot||\rangle$ has a non-trivial feasible solution if and only if the following two conditions are met:

1. The foil set X_ψ includes one or more extreme points of $conv(X)$.
2. The restrictions on d do not prevent all of these extreme points from being optimal to $\mathcal{FW}\langle d, f, X\rangle$.

Proof. If the first condition is not met, then X_ψ lies entirely in the interior region of X and thus cannot contain an optimal solution. If the first condition holds, it is trivial to see that the second condition guarantees the existence of a feasible solution. □

We now turn to NCEs with linear objectives $f(c, x) = c^T x$, which are the focus of this thesis, and identify a sufficient condition guaranteeing feasibility.

Theorem 3.2. An $\mathcal{NCE}\langle c, \mathcal{D}, f, \psi, x^*, X, ||\cdot||\rangle$ with $f(c, x) = c^T x$ has a non-trivial feasible solution if both following conditions are met:

1. For all $i \in \{1, ..., n\}$, let $x_{i,\min} = \min_x\{x_i : x \in X\}$ and $x_{i,\max} = \max_x\{x_i : x \in X\}$.
   - If the forward objective is minimization (i.e. the NCE is an $\text{NCE}_{\min}$), then the foil set X_ψ must include at least one point $\tilde{x}^\psi$ such that $\exists\ j \in \{1, ..., n\}$ for which $\tilde{x}^\psi_j = x_{j,\min}$.
   - If the forward objective is maximization (i.e. the NCE is an $\text{NCE}_{\max}$), the foil set X_ψ must include at least one point $\tilde{x}^\psi$ such that $\exists\ j \in \{1, ..., n\}$ for which $\tilde{x}^\psi_j = x_{j,\max}$.
2. $\mathcal{D}^f_j \subseteq \mathcal{D}$ given
$$\mathcal{D}^f_j = \{d \in \mathbb{R}^n : 0 < d_j \leq d^{UB}_j, d_i = 0\ \forall\ i \neq j\}, \tag{3.3}$$
   where $d^{UB}_j = \max_d\{d_j : d \in \mathcal{D}\}$.

Proof. For any $d^f \in \mathcal{D}^f_j$, the term 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 x_j is possible below its value in $\tilde{x}^\psi$. Similarly, if the forward objective is maximization, no maximization of x_j is possible above its value in $\tilde{x}^\psi$. Therefore, $\tilde{x}^\psi$ is optimal to $\mathcal{FW}\langle d, X\rangle$ for any $d \in \mathcal{D}^f_j$, making any $d \in \mathcal{D}^f_j$ a feasible NCE solution. □

3.4 Discussion and Limitations

We now consider some assumptions about what makes an NCE a useful problem to solve in practice.

Infeasibility Though Theorems 3.1 and 3.2 provide conditions for feasibility, it may not always be easy to satisfy them, as we will see in Chapter 5. Thus, in cases when these conditions are not met, instead of receiving a counterfactual explanation, an explainee will be told "There are no alternative objective parameters under which a solution in X_ψ would have been optimal." Such information may be useful in some contexts, for example for a transportation engineer investigating whether or not a certain behaviour (e.g. route choices in a city) could be achieved by changing the objective parameters (e.g. travel costs). In other cases, however, informing an explainee that a counterfactual explanation does not exist may not be particularly helpful. In future work it may be possible to design a variation of the NCE to search for counterfactual parameters that would minimize the sub-optimality of a foil instead of searching for parameters that would make a foil optimal.

Alternative Optimal Solutions A related issue to infeasibility is the possibility that the foil set contains an alternative optimal solution $\hat{x}^*$ to $\mathcal{FW}\langle c, f, X\rangle$, and thus no change to the objective parameters is needed to make a foil optimal. For instance, the reason that x^* was selected as opposed to $\hat{x}^*$ may have been the result of tie-breaking in the optimization algorithm. Whether it is helpful to provide an explainee with information that the initial solution may just as well have been $\hat{x}^* \in X_\psi$ rather than $x^* \notin X_\psi$ may depend on how much decision-making authority is wielded by the explainee as opposed to the automated system. For example, consider an explainee with complete decision-making authority, such as a driver who is given route suggestions by an optimization system. After discovering that one of his preferred routes is just as good as the system's suggestion, the driver may proceed to take his preferred route knowing that this route is optimal. In contrast, consider an explainee with very little decision-making authority, such as a loan applicant who has no control over a bank's decision process. The applicant is unlikely to find the following information useful: "You were just as qualified to receive a loan as the candidates that were approved, but you were denied one because of an arbitrary tie-breaking process in our software."

Meaningfulness of Objective Parameters A key assumption of using an NCE for explanations is that the objective parameters c and d are meaningful, or can be made meaningful, to the explainee; in other words, the forward problem is decomposable (as defined in Section 2.2.2) with respect to these parameters. For instance, meaningful parameters might include cost in terms of money, time,

or distance. It is also possible that the explainee may not initially understand the meanings of objective coefficients, but can be made to understand them after receiving an additional explanation of the what the coefficients represent. For instance, the explanation "Your order would have been delivered this week if it was associated with priority level 3 instead of priority level 1," may not be meaningful to an explainee who does not understand what these priority levels mean. However, as in Example 3.1, this explanation can be made meaningful if the explainee is also told that priority level 1 corresponds to "standard" shipping and would cost her \$30 while priority level 3 corresponds to "express" shipping and would cost her \$100. In fact, by showing how changes in decisions could be achieved through changes in objective parameters, counterfactual explanations may empower explainees to question other parts of a decision system, such as the assignment of coefficients. For instance, the customer from Example 3.1 may argue that it is not reasonable to need to pay an additional \$70 to get her order delivered next week, and that the delivery service should re-evaluate how priorities are assigned in its system.

Differences from the Initial Solution While the foil constraints are satisfied by the optimal solution to the modified problem $\mathcal{FW}\langle d, X\rangle$, this new optimal solution may be arbitrarily different from the initial solution x^*. In practice, large changes to x^* may be problematic. For instance, if a scheduler at a factory wishes to decide whether it is worth changing an order priority to satisfy a foil constraint, it may be difficult for him to make a meaningful comparison between objectives c and d if the schedules resulting from $\mathcal{FW}\langle c, X\rangle$ and $\mathcal{FW}\langle d, X\rangle$ are very different from each other. Further consideration of this limitation is left for future work, and revisited in Chapter 7.

Privacy The goal of providing an explanation may be at odds with the goal of protecting the privacy of other people involved in the system. For example, the amount of time a patient needs to wait for his surgery may depend on the priority levels assigned to other patients, which depend on their private medical data. For this reason, it may sometimes be preferable to restrict explanations to involve only the objective parameters that do not violate anyone's privacy, even if such explanations are less informative to the explainee. These restrictions can be implemented through the definition of the set $\mathcal{D}$.

No Constraint Parameter Changes In an NCE, explanations can only be provided in terms of objective parameters, not constraint parameters. There has been far less work on inverse optimization where constraint parameters are allowed to change [93, 94], with one of the challenges of

these approaches being that they must handle multiple alternative feasible regions. Explanations that allow counterfactual changes to constraint parameters are a possible direction for future work.

3.5 Conclusion

This chapter defined the NCE (3.1) - (3.2) and discussed how it can be used to generate counterfactual explanations for forward optimization problems. Chapter 4 introduces methods to solve an NCE when the explainee is interested in a single variable and a single objective parameter, and Chapter 5 explores solution approaches for a useful subclass of multivariate explanations. Chapter 6 develops solution methods for NCEs which explain forward problems with linear objectives and constraints.

Chapter 4

Single Variable Explanations

This chapter develops solution methods for an NCE (3.1) - (3.2) for cases when an explainee is interested in changes to a single variable and single objective parameter. These methods assume that the forward objective function is linear, $f(c, x) = c^T x$, and that the decision variables in the forward problem are either binary or integral. We also assume that the contrastive question is either "Why did x^* not satisfy $x_j^* \leq m_0$?", or "Why did x^* not satisfy $x_j^* \geq m_0$?", $m_0 \in \mathbb{N}_0$. That is, it is assumed that the foil constraint is a linear constraint on a single variable. We focus on the more difficult case of integer objective parameters $c, d \in \mathbb{N}_0^n$, though the methods below could easily be modified for continuous parameters. It is shown that in the case of binary decision variables, a solution to the explanation problem can be found in closed form after solving a slightly modified version of the forward problem. For integer variables, the NCE can be solved with a binary search that involves solving a logarithmic number of modified forward problems. Unlike the subsequent chapters, this chapter does not rely on the assumption that the constraints defining X must be linear. As long as the foil constraint is linear, the explanation methods in this chapter are applicable to non-linearly constrained forward problems.

4.1 Single Variable Restrictions

There is a useful subset of explanations in which the explainee only asks about possible changes to a single decision variable x_j and is given an explanation only in terms of changes to the corresponding objective parameter c_j. The primary motivation for this restricted setting arises from forward problems in which, given $f(c, x) = c^T x$, each decision variable x_j and objective parameter c_j correspond to exactly one person j. For example, such formulations are common in medical scheduling, for

instance when x_j is used to represent the completion time of patient j's procedure, c_j represents that patient's acuity (priority level), and the goal is to minimize acuity weighted completion times.

If all the information in the objective function concerning a person j is represented with decision variable x_j and objective parameter c_j, explainee j may wish to ask questions only about the decision that concerns himself, x_j, and receive an explanation only in terms of a parameter that corresponds to himself, d_j. In other words, explainee j may be indifferent to the decisions concerning other people x_i, $i \neq j$, or possible changes to the corresponding objective parameters d_i. One reason may be that person j has no control over the parameters d_i, $i \neq j$ that correspond to other people. For instance, in Example 3.2 in which a number of customers ordered deliveries, if customer j has no control over another customer i, it is likely not useful to tell customer j that "Your order would have arrived next week if customer i had selected "standard" instead of "express" shipping." It may also be important to protect the privacy of people $i \in \{1, ..., n\}$, $i \neq j$ from explainee j, in which case the explainee may only be allowed to reason about possible changes to x_j and c_j. Finally, we can also envision useful univariate questions and explanations in cases where x_j and c_j do not correspond to a single person j; for example, customer j may place several orders, but only be interested in the possibility of changing the priority level of one order.

Let us formally define a univariate version of an NCE, denoted U-NCE. In addition to requiring that only one variable x_j be involved in the contrastive question, we assume that this question can be represented using a single linear foil constraint $x_j \leq m_0$ or $x_j \geq m_0$, $m_0 \in \mathbb{N}_0$. That is, the explainee must ask either "Why does x^* not satisfy $x_j \leq m_0$?" or "Why does x^* not satisfy $x_j \geq m_0$?". The restriction that only the parameter d_j can be involved in an explanation is expressed by requiring that $d_i = c_i$ for all $i \neq j$.

Definition 4.1. (Univariate NCE (U-NCE)). The U-NCE is an NCE $\langle c, \mathcal{D}, f, \psi, x^*, X \rangle$ in which $\mathcal{D} = \{d \in \mathbb{N}_0^n : d_i = c_i \ \forall \ i \neq j\}$, $j \in \{1, ..., n\}$, ψ is either $\psi = \{x \in \mathbb{N}^n : x_j \leq m_0\}$ or $\psi = \{x \in \mathbb{N}^n : x_j \geq m_0\}$, $m_0 \in \mathbb{N}_0$, and $f(c, x) = c^T x$.

Example 4.1. (U-NCE). The problem defined in Examples 3.1 and 3.2 is an example of a U-NCE. The question "Why is order j not scheduled to arrive next week?" is represented using $\psi = \{x \in \mathbb{N}^n : x_j \leq 7\}$. Additionally, the customer is only interested in possible changes to the shipping priority of order j, and this restriction is represented with $\mathcal{D} = \{d \in \mathbb{N}_0^n : d_i = c_i \ \forall \ i \neq j\}$.

4.2 Binary Linear Objective Problems

We now focus on cases in which the underlying forward problem has only binary decision variables, and show that U-NCEs for these problems can be solved in closed form after solving a slightly modified version of the forward problem.

4.2.1 Formulation

For this class of problems, the forward problem has a linear objective, only binary decision variables, and is called a Binary Linear Objective (BLO) problem (Definition 4.2). BLOs constitute a useful subset of combinatorial optimization problems, including, for instance, the 0-1 knapsack (KP) problem (Example 4.2), which will be used to demonstrate that using a U-NCE to explain BLOs can produce meaningful explanations (Example 4.3).

Definition 4.2. (Binary Linear Objective Problem (BLO)). A $\mathcal{BLO}\langle c, X\rangle$ is a forward problem $\mathcal{FW}\langle c, f, X\rangle$ where $X \subseteq \{0,1\}^n$, $c \in \mathbb{N}_0^n$, and $f(c,x) = c^T x$. As with all forward problems, $\mathcal{BLO}_{\min}\langle c, X\rangle$ is a minimization problem while $\mathcal{BLO}_{\max}\langle c, X\rangle$ is a maximization problem, and minimization is assumed if no direction is specified.

Example 4.2. (0-1 Knapsack Problem (KP)). In a 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 $W < \sum_{i=1}^n w_i$. The objective of the 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. 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 problem $\mathcal{KP}\langle c, X\rangle$ is $\max_x\{c^T x : x \in X\}$, $X = \{x \in \{0,1\}^n : w^T x \leq W\}$, which is NP-Complete [95]. The KP is used in many real-life applications, including selecting people to receive a limited service [96], in which case c_i represents the benefit of extending the service to person i and w_i represents the resources required to provide the service to that person. For example, if the service is a loan, c_i may be the expected interest a bank will receive from customer i over the life of the loan, w_i may be the value of i's loan, and W may be the total funds that a bank can loan out. A KP used to select people to receive a limited service is an example of the setting discussed in Section 4.1 where all the information in an objective function concerning person j is represented only by x_j and c_j.

Let a U-NCE in which the forward problem is a BLO be denoted U-NCE$^{0\text{-}1}$ when the forward optimization direction is minimization and U-NCE$^{0-1}_{\max}$ when the forward direction is maximization.

Due to the requirement in Definition 4.1 that the foil constraint must be linear and involve only one variable x_j, in a U-NCE$^{0\text{-}1}$ the foil constraint must be either $x_j = 1$ or $x_j = 0$. In the first case, the foil set would be $X_\psi = \{x \in X : x_j = 1\}$ and the non-foil set would be $X_\psi^C = \{x \in X : x_j = 0\}$. Example 4.3 shows how a U-NCE$^{0\text{-}1}$ can be used to explain a KP.

Example 4.3. (U-NCE$^{0\text{-}1}$ for a KP). Consider a $\mathcal{KP}\langle c, X\rangle$ used to determine which customers $i \in \{1, ..., n\}$ will be selected to receive a loan, as discussed in Example 4.2. Let $c_j = \$50,000$, representing the maximum amount of interest client j is expected to be able to pay safely over the life of the loan, a value limited by j's yearly income of \$40,000. The KP problem is solved to give an optimal solution x^*, in which $x_j^* = 0$, meaning that customer j denied a loan. Customer j asks "Why was I denied a loan instead of approved?", a question which can be rewritten as "Why x^* and not an $x \in X_\psi$?", where $X_\psi = \{x \in X : x_j = 1\}$. We can search for an explanation using a U-NCE$^{0-1}_{\max}$ with $\psi = \{x \in \{0,1\}^n, x_j = 1\}$ and $\mathcal{D} = \{d \in \mathbb{N}^n : d_i = c_i \; \forall \; i \neq j\}$. Assume that the optimal solution to this problem is $d_j^* = \$75{,}000$, a value that the bank estimates customer j could pay safely if her income was \$60,000 per year. The explanation given to j might be "You would have been approved a loan if you could safely pay \$75,000 in interest instead of \$50,000, which you could do if your yearly income was \$60,000 instead of \$40,000." This example also demonstrates that, as discussed in Section 3.4, it can be possible to express changes to objective parameters in a meaningful way to explainees, for instance using changes to their income, even if the objective parameters themselves, such as bank interest revenues, may not be meaningful to the explainee.

4.2.2 Solution Method

We now prove that a U-NCE$^{0\text{-}1}$ can be solved in closed form after solving a slightly modified version of the initial forward problem. Below we assume that the foil constraint is $x_j = 1$ and the forward optimization direction is maximization, but this content is easy to reformulate for the case of minimization or a foil constraint of the form $x_j = 0$. The U-NCE$^{0-1}_{\max}$ can be written as:[1]

$$\min_{d \in \mathcal{D}} ||c - d||_1 \tag{4.1}$$

$$\text{s.t.} \quad \max_{x \in X} d \cdot x = \max_{x \in X_\psi} d \cdot x \tag{4.2}$$

We will show that this problem can be solved based on the difference between the optimal objective value of the initial forward problem $\mathcal{BLO}\langle c, X\rangle$ and that of $\mathcal{BLO}\langle c, X_\psi\rangle$, a modification of the initial

[1] $d^T x$ may be written as $d \cdot x$ where it is obvious from context.

forward problem in which X is replaced by the X_ψ. Intuitively, any loss to the objective function of the forward problem incurred by forcing the solution to lie in X_ψ as opposed to X must be compensated by the change to c_j.

To develop our solution approach, we define two auxiliary functions given an arbitrary objective vector $p \in \mathbb{R}^n_+$. The first function, $\Delta^X_\psi(p)$, is the difference between the optimal objective values of the unrestricted forward problem and the forward problem where the solution must be a foil, $\mathcal{BLO}\langle p, X\rangle$ and $\mathcal{BLO}\langle p, X_\psi\rangle$, respectively:

$$\Delta^X_\psi(p) = \max_{x \in X} p \cdot x - \max_{x \in X_\psi} p \cdot x. \tag{4.3}$$

The second function, $\Delta^{X^C_\psi}_\psi(p)$, is the difference between the optimal objective values of the forward problem where the solution must be a non-foil and that where the solution must be a foil, $\mathcal{BLO}\langle p, X^C_\psi\rangle$ and $\mathcal{BLO}\langle p, X_\psi\rangle$, respectively:

$$\Delta^{X^C_\psi}_\psi(p) = \max_{x \in X^C_\psi} p \cdot x - \max_{x \in X_\psi} p \cdot x. \tag{4.4}$$

In any feasible solution to a U-NCE$^{0-1}_{\max}$, due to constraint (4.2), $\Delta^X_\psi(p) = 0$ and $\Delta^{X^C_\psi}_\psi(p) \leq 0$.

Theorem 4.1. The U-NCE$^{0-1}_{\max}$ $\langle c, \mathcal{D}, \psi, x^*, X\rangle$ with a non-empty foil set $X_\psi \neq \emptyset$ and $\psi = \{x \in X, x_j = 1\}$ has an optimal solution d^* in which $d^*_j = c_j + \Delta^X_\psi(c)$, and $d^*_i = c_i$ for all $i \neq j$.

Proof. If X_ψ contains an alternative optimal solution to $\mathcal{BLO}\langle c, X\rangle$, then $\Delta^X_\psi(c) = 0$, and $d^* = c$. Otherwise, all optimal solutions to $\mathcal{BLO}\langle c, X\rangle$ are non-foils and lie in X^C_ψ. Thus, we can write

$$\Delta^X_\psi(c) = \max_{x \in X^C_\psi} c \cdot x - \max_{x \in X_\psi} c \cdot x. \tag{4.5}$$

Since $x_j = 0$ for each $x \in X^C_\psi$, and $d_i = c_i$ for all $i \neq j$, for the last term:

$$\max_{x \in X^C_\psi} c \cdot x = \max_{x \in X^C_\psi} d \cdot x \qquad \forall d \in \mathcal{D}. \tag{4.6}$$

Similarly, for each $x \in X_\psi$, since $x_j = 1$ the contribution of the non-j components, $d \cdot x - d_j$, must be identical for every d, giving

$$\max_{x \in X_\psi} c \cdot x - c_j = \max_{x \in X_\psi} d \cdot x - d_j \qquad \forall d \in \mathcal{D}. \tag{4.7}$$

Since $\Delta_{\psi}^{X_{\psi}^{C}}(d) \leq 0$ in a feasible solution to the U-NCE$_{\max}^{0-1}$, using equations (4.5) - (4.7) gives

$$\begin{aligned}
0 &\geq \Delta_{\psi}^{X_{\psi}^{C}}(d) \\
0 &\geq \max_{x \in X_{\psi}^{C}} d \cdot x - \max_{x \in X_{\psi}} d \cdot x \\
0 &\geq \max_{x \in X_{\psi}^{C}} c \cdot x - \max_{x \in X_{\psi}} c \cdot x + c_j - d_j \\
0 &\geq \Delta_{\psi}^{X}(c) + c_j - d_j.
\end{aligned}$$

This gives $d_j \geq c_j + \Delta_{\psi}^{X}(c)$, and therefore the optimal d^* satisfies $d_j^* = c_j + \Delta_{\psi}^{X}(c)$. It is also clear that d_j^* is integer because both $c \in \mathbb{N}_0^n$ and $x \in \mathbb{N}_0^n$, and therefore $\Delta_{\psi}^{X}(c) \in \mathbb{N}_0$. □

Thus, since an optimal solution x^* to $BLO\langle c, X\rangle$ is known before the explanation begins, it is sufficient to solve $BLO\langle c, X_{\psi}\rangle$ and use Theorem 4.1 to find the optimal solution to a U-NCE$^{0\text{-}1}$.

4.3 Integer Variable Problems

We now turn to forward problems with integer decision variables and a linear objective, which we call Integer Linear Objective (ILO) problems (Definition 4.3). We show that in the resulting U-NCEs, the difference between the optimal objective value in X and the optimal objective value in X_{ψ} changes monotonically with d_j. After proving that the optimal value of d_j lies on a closed interval, we show that binary search can be used over this interval to solve the U-NCE.

4.3.1 Formulation

Definition 4.3. (Integer Linear Objective Problem (ILO)) An $\mathcal{ILO}\langle c, X\rangle$ is a forward problem $\mathcal{FW}\langle c, f, X\rangle$ where $X \in \mathbb{N}_0^n$, $c \in \mathbb{N}^n$, and $f(c, x) = c^T x$.

An example of an ILO is the forward problem in Example 3.1, a scheduling problem with integer delivery times and importance weights, the goal of which is minimizing weighted delivery times. Let a U-NCE in which the underlying forward problem is an ILO be denoted U-NCE$^{\mathbb{N}_0}$, and recall that the foil constraint in a U-NCE$^{\mathbb{N}_0}$ must have the form $x_j \leq m_0$ or $x_j \geq m_0$. These explanation problems are well suited for scheduling problems and questions such as "Why was task j not completed before m_0?".

4.3.2 Solution Method

Solving a U-NCE$^{\mathbb{N}_0}$ is more involved than solving a U-NCE$^{0\text{-}1}$, because we must consider multiple possible values of x_j in both the foil and non-foil sets. This section assumes that the optimization direction of the forward problem is minimization and the foil constraint has the form $x_j \leq m_0$, but the content below can be easily reformulated for the case of maximization or foil constraints of the form $x_j \geq m_0$. As in Section 4.2.2, we define an auxiliary function,[2]

$$\Delta_{\psi}^{X}(p) = \min_{x \in X_{\psi}} p \cdot x - \min_{x \in X} p \cdot x, \tag{4.8}$$

noting that in any feasible solution to an NCE, $\Delta_{\psi}^{X}(p) = 0$.

Theorem 4.2. Consider a U-NCE$^{\mathbb{N}_0}$ $\langle c, \mathcal{D}, \psi, x^*, X \rangle$ with $X_{\psi} \neq \emptyset$, $\psi = \{x \in X : x_j \leq m_0\}, m_0 \in \mathbb{N}_0$, and where no foil $x \in X_{\psi}$ is an alternative optimal solution to the initial problem $\mathcal{ILO}\langle c, X \rangle$.[3] There exists a constant $p_j^* \in \mathbb{R}_+$ such that $d \in \mathcal{D}$ results in feasible solutions to the U-NCE$^{\mathbb{N}_0}$ iff $d_j \geq p_j^*$. The optimal solution to the U-NCE$^{\mathbb{N}_0}$ is $d^* \in \mathcal{D}$, with $d_j^* = \lceil p_j^* \rceil$, $d_i^* = c_i$ for all $i \neq j$.

Theorem 4.2 will allow us to find the optimal U-NCE$^{\mathbb{N}_0}$ solution using a binary search over the values of $d_j \in \mathbb{N}_0$, at each search step checking whether a given d_j results in a feasible solution to the U-NCE$^{\mathbb{N}_0}$. Theorem 4.2 can be easily proven if Lemma 4.1 below holds.

Lemma 4.1. Given the U-NCE$^{\mathbb{N}_0}$ $\langle c, \mathcal{D}, \psi, x^*, X \rangle$ from Theorem 4.2:

1. $\Delta_{\psi}^{X}(p)$ is a monotonically non-increasing function in p_j, given $p \in \mathcal{P} = \{p \in \mathbb{R}_+^n : p_i = c_i \; \forall \; i \neq j\}$.

2. There is a $p^0 \in \mathcal{P}$ such that $\Delta_{\psi}^{X}(p^0) = 0$.

Intuitively, we will prove Lemma 4.1 by demonstrating that if p_j increases by $\delta_p \in \mathbb{R}_+$, both components of $\Delta_{\psi}^{X}(p)$ will increase, but the negative component $\min_{x \in X} p \cdot x$ will increase faster, thus showing that $\Delta_{\psi}^{X}(p)$ is monotonically non-increasing in p_j and will eventually vanish. A sketch of the steps in the following proof of Lemma 4.1 is:

1. It is shown using Lemma 4.2 that for an increase in p_j of $\delta_p \in \mathbb{R}^+$:

   - The optimal objective of $\mathcal{ILO}\langle p, X_{\psi}^{C} \rangle$, the forward problem where solutions are non-foils, increases by at least $\delta_p(m_0 + 1)$.

[2] This function differs from (4.3) because the forward optimization direction is minimization.

[3] To determine whether X_{ψ} contains an alternative optimal solution to $\mathcal{ILO}\langle c, X \rangle$, we can check if $\min_x\{c \cdot x : x \in X\} = \min_x\{c \cdot x : x \in X_{\psi}\}$. If yes, $\exists \; x \in X_{\psi}$ that is an optimal solution to $\mathcal{ILO}\langle c, X \rangle$, and therefore the optimal solution to the U-NCE$^{\mathbb{N}_0}$ is $d^* = c$.

- The optimal objective of $\mathcal{ILO}\langle p, X_\psi\rangle$, the forward problem where solutions are foils, increases by at most $\delta_p m_0$.

2. The bounds from Lemma 4.2 are used to show that foils become better solutions as p_j increases with respect to non-foils (i.e. $\Delta^X_\psi(p)$ is monotonically non-increasing in p_j).

3. A formula is derived for a p^0, which is guaranteed to exist, at which a foil is guaranteed to be optimal to the forward problem $\mathcal{ILO}\langle p^0, X\rangle$ (i.e. $\Delta^X_\psi(p^0) = 0$).

In what follows we consider two vectors $p^- \in \mathcal{P}$ and $p^+ \in \mathcal{P}$ with $p^-_j \leq p^+_j$, and define the positive difference between their jth components as $\delta_p = p^+_j - p^-_j$. We start by observing that if p^- increases by δ_p, the optimal objective value of the forward problem where the solution must be a non-foil, $\mathcal{ILO}\langle p^-, X^C_\psi\rangle$, must increase by at least $\delta_p(m_0 + 1)$, while the optimal objective value of the problem where the solution must be a foil, $\mathcal{ILO}\langle p^-, X_\psi\rangle$, can increase by at most $\delta_p m_0$.

Lemma 4.2.

1. For all $x \in X^C_\psi$, we have:

$$\min_{x\in X^C_\psi} p^+ \cdot x - \min_{x\in X^C_\psi} p^- \cdot x \geq \delta_p(m_0 + 1). \tag{4.9}$$

2. For all $x \in X_\psi$, we have:

$$\min_{x\in X_\psi} p^+ \cdot x - \min_{x\in X_\psi} p^- \cdot x \leq \delta_p m_0. \tag{4.10}$$

Proof. (Lemma 4.2). By the definition, for all $x \in X$,

$$p^+ \cdot x = p^- \cdot x + \delta_p x_j. \tag{4.11}$$

To prove statement (1) of Lemma 4.2, let $x^{C,-}$ be an optimal solution to $\mathcal{ILO}\langle p^-, X^C_\psi\rangle$ and $x^{C,+}$ be an optimal solution to $\mathcal{ILO}\langle p^+, X^C_\psi\rangle$. By optimality, $p^- \cdot x^{C,-} \leq p^- \cdot x^{C,+}$, which we use together with (4.11) to deduce

$$\min_{x\in X^C_\psi} p^+ \cdot x - \min_{x\in X^C_\psi} p^- \cdot x = p^+ \cdot x^{C,+} - p^- \cdot x^{C,-} \geq p^+ \cdot x^{C,+} - p^- \cdot x^{C,+} = \delta_p x^{C,+}_j \geq \delta_p(m_0 + 1).$$

Similarly, to prove statement (2) of Lemma 4.2, let $x^{\psi,-}$ be an optimal solution to $\mathcal{ILO}\langle p^-, X_\psi\rangle$ and $x^{\psi,+}$ be an optimal solution to $\mathcal{ILO}\langle p^+, X_\psi\rangle$. This gives

$$p^+ \cdot x^{\psi,+} \leq p^+ \cdot x^{\psi,-} = p^- \cdot x^{\psi,-} + \delta_p x^{\psi,-}_j \leq p^- \cdot x^{\psi,-} + \delta_p m_0.$$

We can use this result to write

$$\min_{x \in X_\psi} p^+ \cdot x - \min_{x \in X_\psi} p^- \cdot x = p^+ \cdot x^{\psi,+} - p^- \cdot x^{\psi,-} \leq p^- \cdot x^{\psi,-} + \delta_p m_0 - p^- \cdot x^{\psi,-} \leq \delta_p m_0.$$

□

Equipped with Lemma 4.2, we proceed to prove Lemma 4.1.

Proof. (Lemma 4.1). First, we consider statement (1) of Lemma 4.1: that $\Delta_\psi^X(p)$ is monotonically non-increasing in p_j, given $p \in \mathcal{P}$. We prove this by showing that

$$\Delta_\psi^X(p^-) \geq \Delta_\psi^X(p^+).$$

Let us look at the three following cases:

1. The case $\Delta_\psi^X(p^+) = 0$ is trivial, since by definition $\Delta_\psi^X(p^-) \geq 0$ for every $p^- \in \mathbb{R}_{0+}^n$.

2. Let $\Delta_\psi^X(p^-) = 0$. Recall that $x^{\psi,-}$ is an optimal solution to $\mathcal{ILO}\langle p^-, X_\psi \rangle$ and $x^{C,-}$ is an optimal solution to $\mathcal{ILO}\langle p^-, X_\psi^C \rangle$. Due to the foil constraint, $x_j^{\psi,-} \leq m_0 < x_j^{C,-}$, and in order for $\Delta_\psi^X(p^-) = 0$, we must have $p^- \cdot x^{\psi,-} \leq p^- \cdot x^{C,-}$, which implies

$$p^+ \cdot x^{\psi,+} = p^- \cdot x^{\psi,-} + \delta_p x_j^{\psi,-} \leq p^- \cdot x^{C,-} + \delta_p x_j^{C,-} = p^+ \cdot x^{C,+}.$$

   Since $p^+ \cdot x^{\psi,+} \leq p^+ \cdot x^{C,+}$, then $\Delta_\psi^X(p^+) = 0$.

3. Lastly, assume

$$\min\{\Delta_\psi^X(p^-), \Delta_\psi^X(p^+)\} > 0 \tag{4.12}$$

   For Δ_ψ^X to be monotonically non-increasing in p_j, the following difference of Δ's must be non-positive:

$$\Delta_\psi^X(p^+) - \Delta_\psi^X(p^-) = \min_{x \in X_\psi} p^+ \cdot x - \min_{x \in X} p^+ \cdot x - \min_{x \in X_\psi} p^- \cdot x + \min_{x \in X} p^- \cdot x. \tag{4.13}$$

   We can observe that (4.12) holds iff, for both p^- and p^+, all foils $x \in X_\psi$ are suboptimal to the respective forward problems $\mathcal{ILO}\langle p^-, X \rangle$ and $\mathcal{ILO}\langle p^+, X \rangle$, meaning that all optimal forward solutions to these problems lie in X_ψ^C. Using this condition and rearranging (4.13),

$$\Delta_\psi^X(p^+) - \Delta_\psi^X(p^-) = \min_{x \in X_\psi} p^+ \cdot x - \min_{x \in X_\psi} p^- \cdot x - \min_{x \in X_\psi^C} p^+ \cdot x + \min_{x \in X_\psi^C} p^- \cdot x.$$

Next, we insert bounds on these terms from Lemma 4.2. From Lemma 4.2 part 1, we have

$$\min_{x \in X_\psi^c} p^+ \cdot x - \min_{x \in X_\psi^c} p^- \cdot x \geq \delta_p m_0 + \delta_p.$$

Similarly, Lemma 4.2 part 2 gives

$$\min_{x \in X_\psi} p^+ \cdot x - \min_{x \in X_\psi} p^- \cdot x \leq \delta_p m_0.$$

Applying these two inequalities to 4.13, we have

$$\Delta_\psi^X(p^+) - \Delta_\psi^X(p^-) \leq \delta_p m_0 - \delta_p m_0 - \delta_p \leq 0,$$

which gives $\Delta_\psi^X(p^+) \leq \Delta_\psi^X(p^-)$ and concludes the proof of statement (1) of Lemma 4.1.

Next, we prove statement (2) of Lemma 4.1: that there is a $p^0 \in \mathcal{P}$ such that $\Delta_\psi^X(p^0) = 0$. We must show that

$$\min_{x \in X_\psi} p^0 \cdot x \leq \min_{x \in X_\psi^c} p^0 \cdot x. \tag{4.14}$$

Let us define a vector $x^{\min} \in \mathbb{N}_0^n$, where every entry other than x_j is zero and $x_j^{\min} = m_0 + 1$. For every $x \in X_\psi^C$, we observe that $x_i^{\min} \leq x_i$ for any $i \in \{1, ..., n\}$. Thus, for any $p \in \mathbb{R}_{0+}^n$ it holds that $p \cdot x^{\min} \leq \min_x\{p \cdot x : x \in X_\psi^C\}$. Therefore, (4.14) will hold if we can satisfy

$$\min_{x \in X_\psi} p^0 \cdot x \leq p^0 \cdot x^{\min} = p_j^0(m_0 + 1). \tag{4.15}$$

To this end, we pick an arbitrary $x^\psi \in X_\psi$ and recall that $x_j^\psi \leq m_0$, and observe that we will satisfy (4.15) if we can find a $p_j^0 \in \mathbb{R}_{0+}$ such that

$$p^0 \cdot x^\psi \leq m_0 p_j^0 + \sum_{i \neq j} c_i x_i^\psi \leq p_j^0(m_0 + 1).$$

We can pick

$$p_j^0 = \sum_{i \neq j} c_i x_i^\psi \tag{4.16}$$

using any foil $x \in X_\psi$. Therefore $p^0 \in \mathcal{P}$ must exist since $X_\psi \neq \emptyset$. □

Using Lemma 4.1, proving Theorem 4.2 is simple.

Proof. (Theorem 4.2). Let $p^* \in \mathcal{P} = \{p \in \mathbb{R}_+^n : p_i = c_i \ \forall \ i \neq j\}$ such that $\Delta_\psi^X(p^*) = 0$ and

$\Delta_\psi^X(p^* - \epsilon) > 0$, given an $\epsilon \in \mathcal{E} = \{\epsilon \in \mathbb{R}_+^n, \epsilon_j > 0, \epsilon_i = 0 \; \forall \; i \neq j\}$ and assuming $p_j^* - \epsilon_j \geq 0$. Recalling that:

- $\Delta_\psi^X(p)$ is monotonically non-increasing in p_j
- $\Delta_\psi^X(c) > 0$ since no foil is optimal to $\mathcal{ILO}\langle c, X\rangle$
- $\exists \; p^0$ such that $\Delta_\psi^X(p^0) = 0$

we conclude that p_j^* must be the smallest real number on the interval $(c_j, p_j^0]$ such that $\Delta_\psi^X(p^*) = 0$. Since $\Delta_\psi^X(p)$ is monotonically non-increasing in p_j, all $d \in \mathcal{D}$ with $d_j \geq p_j^*$ must be feasible U-NCE$^{\mathbb{N}_0}$ solutions because all such d's would give $\Delta_\psi^X(d) = 0$, proving the "if" direction of Theorem 4.2. Similarly, since any $d \in \mathcal{D}$ with $d_j < p_j^*$ would result in $\Delta_\psi^X(d) > 0$, all such d's would be infeasible to the U-NCE$^{\mathbb{N}_0}$, proving the "only if" direction of Theorem 4.2. Therefore, the optimal solution to the U-NCE$^{\mathbb{N}_0}$ is $d^* \in \mathcal{D}$ such that $d_j^* = \lceil p_j^* \rceil$, and we can note that d_j^* must be the smallest integer on the interval $(c_j, \lceil p_j^0 \rceil]$ for which $\Delta_\psi^X(d^*) = 0$. □

Since Δ_ψ^X is monotonically non-increasing, and d_j^* is in the interval $(c_j, \lceil p_j^0 \rceil]$, we can find d_j^* with binary search. Because $\Delta_\psi^X(c) > 0$, the smallest possible value for d_j^* is $c_j + 1$, limiting the search interval to $[c_j + 1, \lceil p_j^0 \rceil]$. We can compute p_j^0 with equation (4.16), but we must first find a feasible foil, $x \in X_\psi$. The smallest possible value of p_j^0 computed from (4.16) will be given by $x \in \text{argmin}\{c \cdot x - c_j x_j : x \in X_\psi\}$. Alternatively, it may be more convenient to use the original forward objective function so that $x \in \text{argmin}\{c \cdot x : x \in X_\psi\}$, though this may result in a higher value for p_j^0. Since $x, c \in \mathbb{N}_0$, equation (4.16) gives an integer value for p_j^0, letting us simplify the search interval to $[c_j + 1, p_j^0]$.

For each d^i that is searched, we must check whether $\Delta_\psi^X(d^i) = 0$. This can be done by solving the two modified problems $\mathcal{ILO}\langle d^i, X_\psi^C\rangle$ and $\mathcal{ILO}\langle d^i, X_\psi\rangle$ and observing that $\Delta_\psi^X(d^i) = 0$ iff

$$\min_{x \in X_\psi^C} d^i \cdot x \geq \min_{x \in X_\psi} d^i \cdot x.$$

Given that the complexity of binary search is logarithmic in the size of the search set, we can make the following statement:

Proposition 4.1. The U-NCE$^{\mathbb{N}_0}$ problem from Theorem 4.2 can be solved with at most $\log_2(p_j^0 - c_j - 1)$ solutions of $\mathcal{ILO}\langle d^i, X_\psi^C\rangle$ and $\mathcal{ILO}\langle d^i, X_\psi\rangle$.

4.4 Conclusion

This chapter developed explanation approaches for combinatorial problems with binary or integer linear objectives for cases when an explainee is interested in a single variable $x_j \in X$ and its corresponding objective parameter $c_j \in \mathcal{D}$. Such explanations are particularly well suited when there are n people represented in a forward problem and x_j and c_j correspond to exactly one person j. In this case, explainee j may have no interest in the decisions x_i or parameters c_i corresponding to other people, and it may also be important to protect the privacy of other people's information from explainee j. Our methods are valid when expainee j wants to know either "Why did x_j^* not satisfy $x_j \leq m_0$?" or "Why did x_j^* not satisfy $x_j \geq m_0$?". For problems with binary decision variables, an explanation can be generated in closed form after solving one modified forward problem $\mathcal{BLO}\langle c, X_\psi \rangle$. When the decision variables are integer, an explanation requires at most a logarithmic number of solutions to the problems $\mathcal{ILO}\langle d, X_\psi \rangle$ and $\mathcal{ILO}\langle d, X_\psi^C \rangle$.

Given these theoretical results, it was decided that numerical experiments would not constitute a substantial research contribution. Specifically, in the binary case, experiments would have simply measured the speed of solving $\mathcal{BLO}\langle c, X_\psi \rangle$, which is the initial forward problem plus a single assignment constraint. In the integer case, experiments would have measured how long it takes to solve a small (logarithmic) number of forward problems $\mathcal{ILO}\langle d, X_\psi \rangle$, and $\mathcal{ILO}\langle d, X_\psi^C \rangle$ when d is updated using the binary search strategy above. Neither of these experiments seemed likely to produce interesting methodological contributions, so research and experimental effort was instead focused on investigating multivariate explanations.

Chapter 5

Multivariate Explanations Under Partial Assignment Restrictions

This chapter develops approaches to multi-variable explanations for a subset of NCEs with linear forward problems and discrete decision variables. Specifically, it is assumed that an explainee is interested in alternative value assignments to a subset of decision variables x_i, $i \in \mathcal{M} \subseteq \{1, ..., n\}$, and would like an explanation in terms of corresponding alternative objective parameters d_i, $i \in \mathcal{M}$. It is shown that this subset of NCEs can be solved as a classical inverse optimization problem (2.2)-(2.3) that uses the optimal solution to $\mathcal{FW}\langle c, X_\psi \rangle$ as the target solution in the inverse problem. Furthermore, this chapter considers cases when the forward problem is a constraint program and introduces inverse constraint programming (CP) as part of the solution method. Inverse CP is developed by modifying Wang's [10] cutting plane algorithm, Algorithm 2.1, to use CP instead of integer programming, and results in both pure CP and hybrid MIP-CP inverse algorithms. The performance of these algorithms is evaluated for generating explanations to two combinatorial optimization problems: the 0-1 knapsack problem and single machine scheduling with release dates. Numerical experiments show that a MIP-CP hybrid approach can outperform a pure MIP approach particularly when CP is the state of the art for the forward optimization problem.

5.1 Partial Assignment Nearest Counterfactual Explanations

We consider a subset of NCE problems where the explainee 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, $x_i^p \in \mathbb{R}^m, i \in \mathcal{M}$. To express this requirement, the foil constraints must constitute m partial assignments, giving $\psi = \{x \in \mathbb{R}^n : x_i = x_i^p \ \forall \ i \in \mathcal{M}\}$. Furthermore, it is assumed the explainee is interested in an explanation in terms of only the corresponding counterfactual objective parameters $d_i, i \in \mathcal{M}$. This requirement is expressed by setting $\mathcal{D} = \{d : d_j = c_j \ \forall \ j \in \mathcal{M}^C\}$, where $\mathcal{M}^C = \{1, ..., n\} \setminus \mathcal{M}$. Such a problem is called a Partial Assignment NCE (PA-NCE), given by Definition 5.1, and an example of which is given in Example 5.1. Observe that if $n = m$, the PA-NCE is equivalent to the classical inverse optimization problem (2.2) - (2.3). The PA-NCE can also be interpreted as a variant of assignment-based partial inverse optimization (2.4)-(2.7), where $\mathcal{D}$ is used to establish additional restrictions on changes to objective parameters.

Definition 5.1. (Partial Assignment NCE (PA-NCE)). The $\mathcal{PA} - \mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ is an $\mathcal{NCE}$ $\langle c, \mathcal{D}, \psi, x^*, X\rangle$ in which $\psi = \{x \in \mathbb{R}^n : x_i = x_i^p \ \forall \ i \in \mathcal{M}\}$ where $\mathcal{M} \subseteq \{1, ..., n\}$, $x^p \in \mathbb{R}^m$, $m = |\mathcal{M}|$, $\mathcal{D} = \{d \in \mathbb{R}_+^n : d_i = c_i \ \forall \ i \in \mathcal{M}^C\}$, and $\mathcal{M}^C = \{1, ..., n\} \setminus \mathcal{M}$.

The reasons for considering this subset of NCEs are similar to those for considering single variable explanations in Chapter 4. Mainly, an explainee may be indifferent to possible changes to some variables, x_j, and their corresponding objective parameters, d_j, $j \in \mathcal{M}^C$. For instance, consider explaining the scheduling problem described in Example 3.1, and assume that the orders $j \in \mathcal{M}^C$ come from customers of which the explainee has no knowledge of or control over. In this case, an explanation indicating how the order priorities d_j would need to change for the explainee to be satisfied with the schedule may have no value. Further, it may be important to restrict the explanation to involve only the decisions and parameters directly associated with the explainee in order to protect the privacy of others in the system.

Example 5.1. (PA-NCE). Consider again the scheduling problem $\mathcal{FW}\langle c, X\rangle$ from Example 3.1. Assume that the explainee has placed two orders, $\mathcal{M} = \{1, 2\}$ and paid \$30 for "Standard" shipping for both, so that $c_1 = c_2 = 1$. However, the explainee is disappointed to learn that in an optimal soltuion x^*, both orders are scheduled to arrive in two weeks, so that $x_1 = x_2 = 14$. They ask "Why could order 1 not arrive in two days, and order 2 not arrive in five days?". To generate an explanation, a PA-NCE is created where $\mathcal{M} = \{1, 2\}$ and $\psi = \{x \in \mathbb{N}_0 : x_1 = 2, x_2 = 5\}$. If the optimal solution to this problem is, for instance, $d_1^* = 3$ and $d_2^* = 2$, the explanation would be: "Order 1 would have arrived in two days if you paid \$100 for Express Shipping ($d_1^* = 3$) and order 2 would have arrived in five days if you paid \$50 for Priority Shipping ($d_2^* = 2$)."

5.2 Theoretical Results

This section shows that, because of its structure, the PA-NCE can be solved as a classical inverse optimization problem that uses the optimal solution to $\mathcal{FW}\langle c, X_\psi\rangle$ as the target solution. While in classical inverse optimization, there is a single, fully defined target solution $x^d \in X$ that must become optimal to $\mathcal{FW}\langle d, X\rangle$, in an NCE (3.1) - (3.2), it is sufficient for any x in the set X_ψ to become optimal to $\mathcal{FW}\langle d, X\rangle$. The possible multiplicity of X_ψ therefore adds a degree of difficulty to the NCE when compared to classical inverse optimization. Fortunately, this section shows that the structure of a PA-NCE guarantees that there exists a foil $x^{\psi,*} \in X_\psi$ will be optimal to $\mathcal{FW}\langle d, X\rangle$ for any $d \in \mathcal{D}$, implying that none of the other foils $x \in X_\psi$, $x \neq x^{\psi,*}$ need to be tracked if $x^{\psi,*}$ is known. In particular, it will be demonstrated that $x^{\psi,*}$, called the optimal foil, is given by the optimal solution to $\mathcal{FW}\langle c, X_\psi\rangle$. This observation means that a PA-NCE can be solved in two steps: first, by solving $\mathcal{FW}\langle c, X_\psi\rangle$ to find $x^{\psi,*}$, and then solving the inverse optimization problem $\mathcal{IO}\langle c, \mathcal{D}, x^d, X\rangle$ (2.2)-(2.3) with $x^d = x^{\psi,*}$.

Theorem 5.1. The $\mathcal{PA} - \mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ is equivalent to the inverse optimization problem $\mathcal{IO}\langle c, \mathcal{D}, x^d, X\rangle$ with $x^d = x^{\psi,*} \in \arg\min\{c \cdot x : x \in X_\psi\}$.

Proof. (Theorem 5.1). Both the $\mathcal{IO}\langle c, \mathcal{D}, x^d, X\rangle$ and the $\mathcal{PA} - \mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ share the same objective, $\min_{d\in\mathcal{D}} ||d - c||_1$, and the two problems differ from each other only in their constraints. For a solution d to be feasible to $\mathcal{IO}\langle c, \mathcal{D}, x^d, X\rangle$, it must satisfy constraint (2.3):

$$d \cdot x^d = \min_{x\in X} d \cdot x,$$

while for d to be feasible to $\mathcal{PA} - \mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$, it must satisfy constraint (3.2):

$$\min_{x\in X_\psi} d \cdot x = \min_{x\in X} d \cdot x.$$

These two constraints differ only in their left-hand sides, therefore to show that the two problems are equivalent, it is sufficient to demonstrate that

$$d \cdot x^d = \min_x\{d \cdot x : x \in X_\psi\}, \quad \forall\ d \in \mathcal{D}. \tag{5.1}$$

Given that x^d is optimal to $\mathcal{FW}\langle c, X_\psi\rangle$ and separating the objective into the contributions from the

variables in $\mathcal{M}$ and the variables in $\mathcal{M}^C$, then

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

The form of ψ requires that $x_i^p = x_i^d = x_i$ for all $i \in \mathcal{M}$ and $x \in X_\psi$, so the above equation can be simplified as

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

Next, due to the constraints in $\mathcal{D}$, $d_j = c_j$ for $j \in \mathcal{M}^C$, giving

$$\sum_{j \in \mathcal{M}^C} d_j x_j^d \leq \sum_{j \in \mathcal{M}^C} d_j x_j, \quad \forall\ x \in X_\psi,\ \forall\ d \in \mathcal{D}.$$

Further, because the values of x_i, $i \in \mathcal{M}$, must be x_i^p in all foils $x \in X_\psi$, the contribution of $\sum_{i \in \mathcal{M}} d_i x_i$, is equivalent in all foils given a value of d. Adding this contribution to both sides of the above inequality gives

$$\sum_{i \in \mathcal{M}} d_i x_i^d + \sum_{j \in \mathcal{M}^C} d_j x_j^d \leq \sum_{i \in \mathcal{M}} d_i x_i + \sum_{j \in \mathcal{M}^C} d_j x_j, \quad \forall\ x \in X_\psi, \quad \forall\ d \in \mathcal{D}. \tag{5.2}$$

Thus $d \cdot x^d \leq d \cdot x$ for all $x \in X_\psi$ and all $d \in \mathcal{D}$, satisfying (5.1). □

Theorem 5.1 holds for any $\mathcal{M} \subseteq \{1, ..., n\}$, including $\mathcal{M} = \{1, ..., n\}$. In this case, the explainee specifies a fully defined solution $x^p \in X$ and the foil set is a singleton, $X_\psi = \{x^p\}$. Such a problem can be solved directly as an inverse optimization problem $\mathcal{IO}\langle c, \mathcal{D}, x^d, X\rangle$ with $x^d = x^p$.

5.3 Inverse Constraint Programming

The two step approach for solving PA-NCEs from Section 5.2 can be used to explain forward problems $\mathcal{FW}\langle c, X\rangle$ that are constraint programs with linear objectives and feasible sets that can be defined with linear constraints.[1] The first step, solving $\mathcal{FW}\langle c, X_\psi\rangle$, is straightforward, since it simply requires solving the initial constraint program $\mathcal{FW}\langle c, X\rangle$ plus the foil assignment constraints. However, for the second step, a method needs to be found to solve inverse optimization problems $\mathcal{IO}\langle c, \mathcal{D}, x^d, X\rangle$ when the underlying forward problem is a constraint program. No such approaches

[1] The requirement that the constraint set is linear is in place to guarantee a finite number of extreme points, which is a condition for the finite termination of *InvLP-MILP* (Algorithm 2.1) [10], the algorithm that forms the basis of the inverse optimization methods in this chapter.

are available in the literature, motivating the development of inverse CP.

An inverse CP approach can be created by modifying Wang's [10] cutting plane algorithm for inverse MILP, *InvLP-MILP* (Algorithm 2.1), which iteratively solves an LP master problem $\mathcal{MP}_{\mathcal{LP}}$ (2.13) - (2.16) and a MILP subproblem $\mathcal{SP}_{\mathcal{MILP}}\langle d, X\rangle$ (2.12). Specifically, Wang's algorithm can be used directly if the master problem and subproblem are reformulated as constraint programs.

5.3.1 Scope

Before modifying *InvLP-MILP* to incorporate CP, the scope of the forward problems to be studied is narrowed slightly. Due to the discrete nature of CP, in this chapter, it is assumed that the objective parameters are integer, $c, d \in \mathbb{N}_0^n$. To compare CP and MILP explanation algorithms, a version of *InvLP-MILP* is defined with discrete objective parameters; the integrality of these parameters is enforced by constraint (5.5). Additionally, most constraint programs involve finite domains and therefore have bounded objectives. This chapter assumes that the forward problems are of this type and thus the duality constraints preventing unbounded objectives (2.14) in $\mathcal{MP}_{\mathcal{LP}}$ can be ignored. Section 5.4 will show that the problems used for the experiments in this chapter indeed have finite domains. Let the master problem with integer decision variables and no duality constraints be called $\mathcal{MP}_{\mathcal{MILP}}$, given by

$$\min_{g,h} \; g + h \tag{5.3}$$

$$(c - g + h)^T x^d \leq (c - g + h)^T x^0 \qquad \forall x^0 \in \mathcal{S}^0 \tag{5.4}$$

$$g, h \in \mathbb{N}_0^n, \quad (c - g + h) \in \mathcal{D}. \tag{5.5}$$

The above master problem can be solved as a MILP, so the variant of *InvLP-MILP* where $\mathcal{MP}_{\mathcal{LP}}$ is replaced with $\mathcal{MP}_{\mathcal{MILP}}$ will be called *InvMILP-MILP*. Next, *InvMILP-MILP* is modified to use CP, establishing a method to solve inverse constraint programs using both pure inverse CP and two inverse MILP-CP hybrids.

5.3.2 Pure Inverse CP

Letting $\mathcal{MP}_{\mathcal{MILP}}$ and $\mathcal{SP}_{\mathcal{MILP}}\langle d, X\rangle$ expressed as constraint programs be called $\mathcal{MP}_{\mathcal{CP}}$ and $\mathcal{SP}_{\mathcal{CP}}\langle d, X\rangle$, respectively, the pure inverse CP algorithm *InvCP-CP* (Definition 5.2) uses the cutting plane approach of *InvLP-MILP* to iteratively solve $\mathcal{MP}_{\mathcal{CP}}$ and $\mathcal{SP}_{\mathcal{CP}}\langle d, X\rangle$.

Definition 5.2. (*InvCP-CP*). Follow steps 1-3 in *InvLP-MILP* (Algorithm 2.1) using CP to solve $\mathcal{MP}_{\mathcal{CP}}$ and $\mathcal{SP}_{\mathcal{CP}}$ instead of using LP and MILP to solve $\mathcal{MP}_{\mathcal{MILP}}$ and $\mathcal{SP}_{\mathcal{MILP}}$, respectively.

5.3.3 Hybrid Inverse CP

Since the master problem (5.3) - (5.5) can be expressed using either MILP or CP, an inverse MILP-CP hybrid, called *InvMILP-CP*, can be created by following the steps of *InvLP-MILP* and using MILP to solve $\mathcal{MP}_{\mathcal{MILP}}$ and CP to solve $\mathcal{SP}_{\mathcal{CP}}\langle d, X\rangle$. This hybrid takes advantage of MILP algorithms being well suited to problems with simple linear structures such as (5.3) - (5.5), while allowing the forward (sub) problem to be expressed and solved with CP.

A second hybrid is motivated by the fact that the forward problems in this chapter's experiments can be formulated as both MILPs and constraint programs, allowing $\mathcal{SP}_{\mathcal{MILP}}\langle d, X\rangle$ and $\mathcal{SP}_{\mathcal{CP}}\langle d, X\rangle$ to be interchangeable. When this is the case, a second hybrid, *InvCP-MILP*, can be defined, which follows the steps of *InvLP-MILP* to solve $\mathcal{MP}_{\mathcal{CP}}$ using CP and $\mathcal{SP}_{\mathcal{MILP}}\langle d, X\rangle$ using MILP.

5.3.4 Algorithm Summary

Four inverse algorithms have been defined, which can be used interchangeably for inverse problems in which the master problem and subproblem can be formulated with both CP and MILP:

- Pure inverse MILP, *InvMILP-MILP*
- Pure inverse CP, *InvCP-CP*
- Two hybrids, *InvMILP-CP* and *InvCP-MILP*.

Additionally, in the experiments in Section 5.6, the addition of the early stopping criteria (ESC) [85] defined in Algorithm 2.2 will be evaluated for the algorithms using MILP master problems. These algorithms will be denoted *InvMILP-MILP*(ESC) and *InvMILP-CP*(ESC) respectively.

5.4 Models

The two-step PA-NCE solution approach is numerically evaluated for two forward problems: the 0-1 knapsack problem (KP, Example 4.2) and single machine scheduling with release dates, $1|r_j|\sum w_j C_j$. The KP was selected because it is NP-complete [95], has a simple structure, and is easy to understand. 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 [97], scheduling problem. As shown below, both problems can be represented as both MILPs and constraint programs.

5.4.1 0-1 Knapsack Problem

The KP is defined in Example 4.2.

MILP Model

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

$$\min_x \{c^T x : w^T x \leq W, x \in \{0,1\}^n\}. \tag{5.6}$$

CP Model

The KP CP model uses a packing global constraint, specifically *binPackingCapa* [98]. 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 KP is

$$\max\ c \cdot x \tag{5.7}$$

$$\text{s.t. } \mathit{binPackingCapa}(\{\langle 0, \infty \rangle, \langle 1, W \rangle\}, \{\langle x_i, w_i \rangle | i \in \{1, ..., n\}\}) \tag{5.8}$$

$$x \in \{0,1\}^n. \tag{5.9}$$

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.

5.4.2 Single Machine Scheduling with Release Dates, $1|r_j|\sum w_j C_j$

In the $1|r_j|\sum w_j C_j$ problem, 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[2] $c_i \in \mathbb{N}$, and a release date $r_i \in \mathbb{N}$. The objective is to minimize the weighted

[2]This problem is typically defined with w representing the job weights. To keep notation consistent, w is replaced with c, though the problem will be referred to by its typical name, $1|r_j|\sum w_j C_j$.

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).

Time-Indexed MIP Model

Though several MIP formulations exist for $1|r_j|\sum w_j C_j$, a time-indexed formulation is used due to its strong performance over a variety of instances [99]. 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. Given a time horizon T, which is an upper bound on latest completion time of any job (a formula for T is given in Section 5.5), the model is

$$\min \sum_{i=1}^{n} \sum_{t=0}^{T-q_i} c_i(t+q_i)x_{i,t} \tag{5.10}$$

$$\text{s.t.} \sum_{t=0}^{T-q_i} x_{i,t} = 1 \qquad \forall\, i \in \{1,...,n\} \tag{5.11}$$

$$\sum_{i=1}^{n} \sum_{s=\max(0,t-q_i+1)}^{t} x_{i,s} \leq 1 \qquad \forall\, t = 0,1,...,T-1 \tag{5.12}$$

$$\sum_{t=0}^{r_i-1} x_{i,t} = 0 \qquad \forall\, i \in \{1,...,n\} \tag{5.13}$$

$$x_{i,t} \in \{0,1\} \qquad \forall\, i \in \{1,...,n\},\ \forall\, t \in 1,...,T-1. \tag{5.14}$$

Constraints (5.11) force each job to start exactly once. Constraints (5.12) ensure no two jobs are processed at the same time, and constraints (5.13) enforce the release dates.

CP Model

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

$$\min \sum_{i=1}^{n} c_i e_i \tag{5.15}$$

$$\text{s.t.}\ \ NoOverlap(\{I_1,...,I_n\}) \qquad \forall\, i \in \{1,...,n\} \tag{5.16}$$

$$s_i \geq r_i \qquad \forall\, i \in \{1,...,n\} \tag{5.17}$$

$$I_i = intervalVar(q_i, [s_i, e_i]) \qquad \forall\, i \in \{1,...,n\}. \tag{5.18}$$

Constraint (5.16) is the *NoOverlap* global constraint that forces jobs to be processed one at a time. Constraints (5.17) ensure that jobs do not start before they are released.

5.4.3 Bounded Objectives

The objectives of both the knapsack and scheduling problems are guaranteed to be bounded for any feasible parameter vector d. The domain of the KP decision vector $x \in \{0,1\}^n$ is a finite set and guarantees the objective function is bounded from above by $\sum_{i=1}^{n} c_i$ and from below by 0. For $1|r_j|\sum w_j C_j$, it is well known that a lower bound on the objective is given by the solution to the problem without release dates, $1||\sum w_j C_j$ [100], for which the shortest processing time (SPT) dispatch heuristic gives an optimal solution. An upper bound on the $1|r_j|\sum w_j C_j$ objective is given by $\sum_{i=1}^{n} Tc_i$. Since both objectives are guaranteed to remain bounded for any feasible value of d, it is safe to remove the duality constraints (2.14) from the master problem when constructing inverse problems.

5.5 Experimental Setup

The goal of the following experiments is to evaluate the PA-NCE based explanation approach for the two combinatorial optimzation problems described above using the two-step solution method described in Section 5.2. The experiments involve solving an initial forward problem, generating a contrastive question, and solving the resulting PA-NCE, focusing on the latter.

5.5.1 Problem Instance Generation

To generate PA-NCE instances, a forward problem instance is created and solved, and then a set of foil constraints is generated to represent a contrastive question.

KP Instances

The KP problem instance sizes tested were $n \in \{20, 30, 40\}$, the profit c_i and weight w_i values were both drawn independently from the random uniform distribution $[1, R]$ with $R = 1000$, and the knapsack capacity was $W = \max\{\lfloor P\sum_{i=1}^{n} w_i \rfloor, R\}$, with $P = 0.5$. Each instance $\mathcal{KP}\langle c, X\rangle$ was solved to produce an optimal solution, x^*.

To generate the contrastive question, $m \in \{5, 10, 15\}$ items were randomly selected from $\{1, ..., n\}$ to create the set $\mathcal{M}$. Each proposed assignment $x_i^p, i \in \mathcal{M}$, was set to the opposite value of x_i^*,

that is, 0 if $x_i^* = 1$ and 1 if $x_i^* = 0$. The foil set X_ψ was defined by adding the assignment constraints $x_i = x_i^p,\ \forall i \in \mathcal{M}$, to the constraints defining X, and restrictions on changes to the objective parameters were expressed with $\mathcal{D} = \{d \in \mathbb{N}_0^n : d_i = c_i\ \forall\ i \in \mathcal{M}^C\}$. There were 20 problem instances tested for each combination of (n, m). This instance generation procedure sometimes resulted in infeasible PA-NCEs if the item assignments in $\mathcal{M}$ forced the knapsack to exceed its capacity. In these cases, a new random set $\mathcal{M}$ was generated until a non-empty foil set X_ψ was found, but these cases were rare.

Single Machine Scheduling Instances

For the $1|r_j|\sum w_j C_j$ problem, forward instances of size $n = \{5, 10, 15\}$ were generated with the random uniform distributions $q_i \in [10, 100]$, $c_i \in [1, 10]$, and $r_i \in [0, \lfloor \alpha Q \rfloor]$, where $\alpha = 0.4$ and $Q = \sum_{i=1}^{n} q_i$. The time horizon T was calculated as $T = \lfloor \alpha Q \rfloor + Q$. Twenty instances were generated for each value of (n, m), with values of m given in Section 5.6.

Generating a feasible set of foil constraints to assign start times to the jobs $i \in \mathcal{M}$ was non-trivial for this problem due to the possibility of infeasible PA-NCEs. In an optimal solution for a given complete sequence of jobs, all jobs are left-shifted subject to the release date 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.

Thus, to generate instances more likely to have feasible explanations, a different approach is followed, although the infeasibility of some PA-NCEs remains an issue (see Section 5.6.4). A random permutation $(a_i)_{i \in \mathcal{M}}$ is created for a randomly chosen subset of m jobs, $\mathcal{M}$. The original problem, with added constraints requiring the jobs in $\mathcal{M}$ to follow the permutation $(a_i)_{i \in \mathcal{M}}$, is then solved to optimality. Finally, the job completion times for the contrastive question, $x_i^p, i \in \mathcal{M}$, are set to be the completion times from this solution.

Specifically, the constraints added to the CP forward problem were

$$endBeforeStart(I_j, I_i) \qquad \forall\ i, j \in \mathcal{M}, a_i > a_j, \tag{5.19}$$

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 the MILP problem, the constraints added were

$$\sum_{t=0}^{T-q_j} tx_{j,t} < \sum_{t=0}^{T-q_i} tx_{i,t} \qquad \forall\ i, j \in \mathcal{M}, a_i > a_j. \tag{5.20}$$

After obtaining the values $x_i^p, i \in \mathcal{M}$, needed to define the foil set, the set of valid counterfactual parameters is defined as $\mathcal{D} = \{d \in \mathbb{N}^n : d_i = c_i\ \forall\ i \in \mathcal{M}^C\}$, giving a complete PA-NCE definition.

5.5.2 Solving PA-NCEs

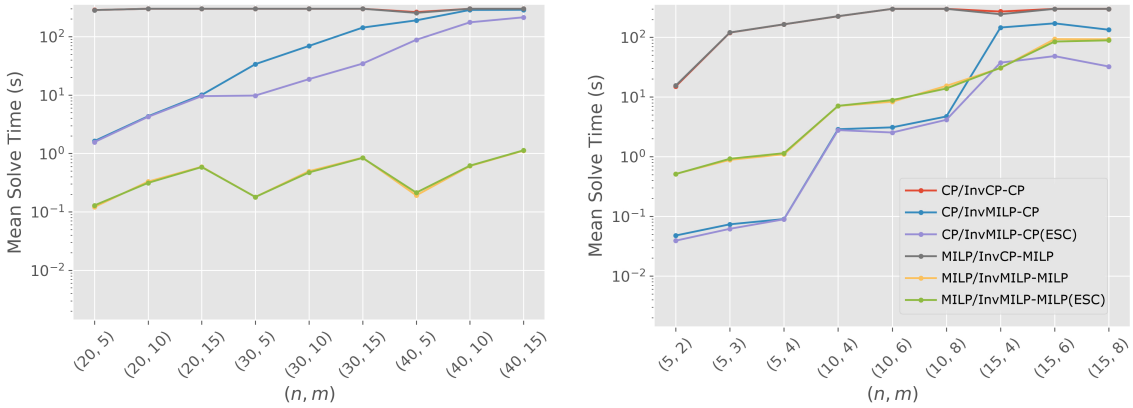
The PA-NCE instances, $\mathcal{PA} - \mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$, are solved using the two-step approach from Section 5.2: first finding the optimal foil $x^{\psi,*}$ by solving $\mathcal{FW}\langle c, X_\psi\rangle$, and then solving the inverse problem problem $\mathcal{IO}\langle c, \mathcal{D}, x^{\psi,*}, X\rangle$, using $x^{\psi,*}$ as the target solution.

Since the forward problems can be solved with both CP and MILP, and there are several inverse algorithms available in Section 5.3, there are multiple ways to solve each PA-NCE instance. Two groups of two-step PA-NCE algorithms were tested for each instance: in the first group, all forward problems (including the subproblem in the inverse algorithms) were solved with CP, and in the second group, all forward problems were solved with MILP. For each of these two algorithm groups, three inverse algorithms were tested:

1. Using CP for the master problem
2. Using MILP for the master problem
3. Using MILP for the master problem and applying the ESC (Algorithm 2.2) [85] to the subproblem.

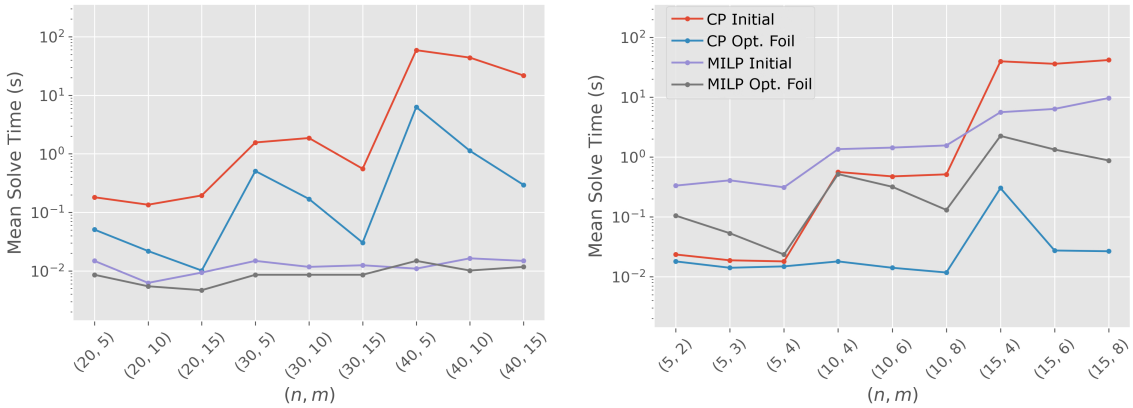
The two-step PA-NCE are named by first specifying the technique (CP or MIP) used to solve the optimal foil problem $\mathcal{FW}\langle c, X_\psi\rangle$, and then specifying the inverse algorithm. For instance, *CP/InvMILP-CP* is the algorithm that uses CP to find the optimal foil (step one) and *InvMILP-CP* to solve the inverse problem (step two). Six two-step PA-NCE algorithms are tested in total for each instance. In all PA-NCE algorithms, the optimal forward solution x^* from the instance generation stage is used in the inverse algorithm to initialize the set of known solutions as $\mathcal{S}^0 = \{x^*\}$.

Finally, the performance of CP and MILP for solving the initial forward problem $\mathcal{FW}\langle c, X\rangle$ is also tracked. While this computation is part of instance generation and not explanation generation, it is a useful proxy for the solver performance in the subproblem of the inverse algorithms, $\mathcal{SP}\langle d, X\rangle$, since the subproblem differs from the initial problem only in its objective.



(a) KP (b) Single Machine Scheduling

Figure 5.1: PA-NCE Mean Solve Times.



(a) KP (b) Single Machine Scheduling

Figure 5.2: Mean Solve Times for Initial Forward and Optimal Foil Problems.

5.5.3 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 (see Section 2.6.2), 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.

5.6 Experimental Results

Mean solution times are shown in Figure 5.1 for the two-stage PA-NCE algorithms and in Figure 5.2 for the optimal foil problem $\mathcal{FW}\langle c, X_\psi \rangle$ (stage 1) and the initial forward problem $\mathcal{FW}\langle c, X \rangle$. Box plots illustrating the distributions of PA-NCE solution times are shown in Figures 5.3 and 5.4.

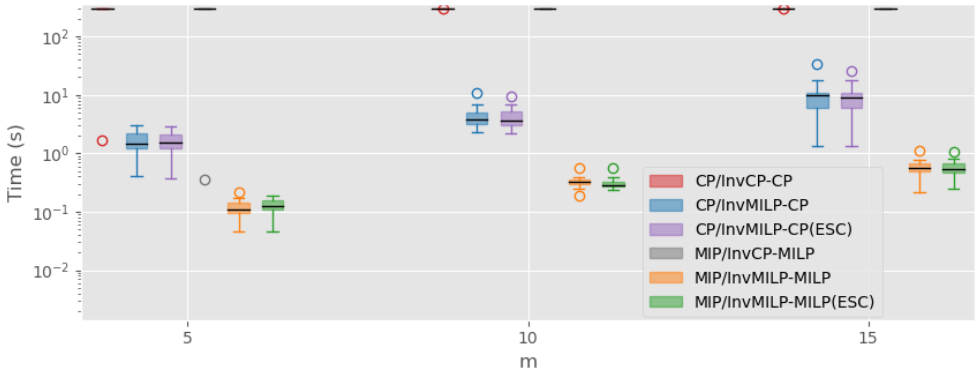
5.6.1 Strongest PA-NCE Algorithms

Knapsack Problem For PA-NCEs based on the KP, the *MILP/InvMILP-MILP* and *MILP/InvMILP-MILP*(ESC) algorithms are by far the most effective. Their efficiency is driven by strong MILP performance on the forward problem, demonstrated in Figure 5.2a for the $\mathcal{KP}\langle c, X \rangle$, which is not surprising since MILP solvers are typically very good for knapsack constraints.

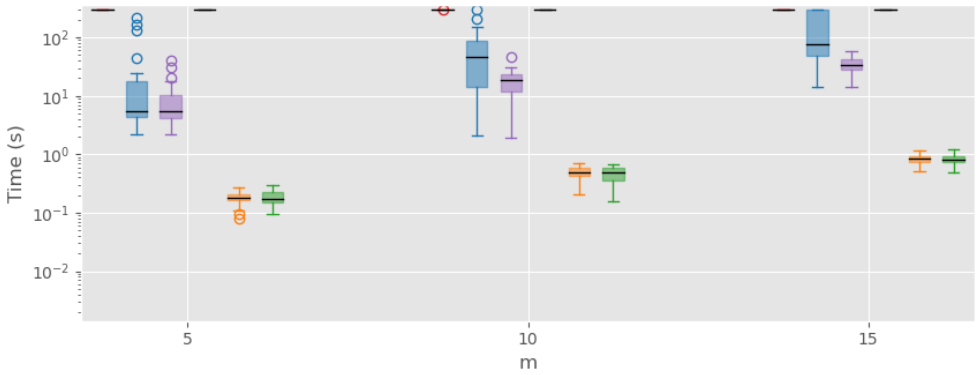
Scheduling Problem For PA-NCEs based on the single-machine scheduling problem, the best performing algorithm overall is *CP/InvMILP-CP*(ESC). For instances with $n \leq 10$, the success of this algorithm is driven by the superiority of CP over MILP for the forward problem (Figure 5.2b). For instances with $n = 15$, though MILP is more efficient than CP for the forward problem, *CP/InvMILP-CP*(ESC) remains the best performing PA-NCE algorithm due to the beneficial effects of the ESC for *InvMILP-CP* (Figure 5.1b).

5.6.2 Early Stopping Criteria

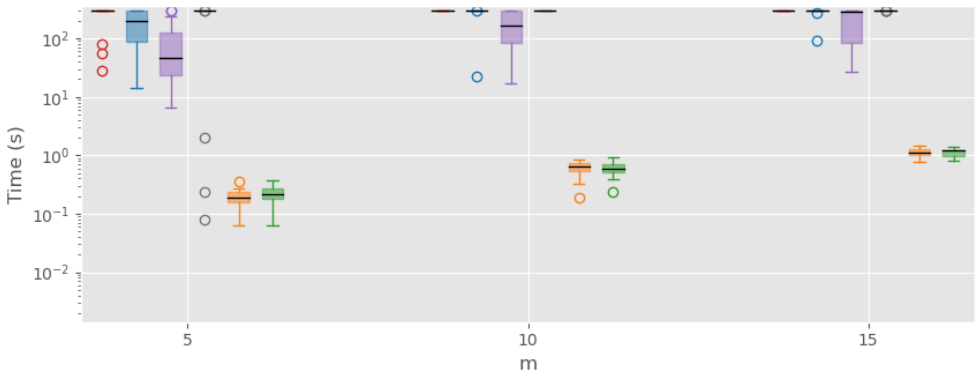
For both problems, the early stopping criteria was clearly beneficial for *InvMILP-CP*, when n was sufficiently large. It had no effect for problems with small n since the time to solve the forward problem was less than γ. Interestingly, the ESC did not produce improvements the forward MILP based algorithm, *InvMILP-MILP*, even though Bodur *et al.* [85] showed that the ESC was beneficial for *InvLP-MILP*. One possible reason for this disparity is that the inverse problem instances studied in this chapter may be too small for the ESC to have an effect on inverse algorithms that use MILP for forward problems. Regardless, Figure 5.1 demonstrates that the ESC may effect inverse algorithms differently depending on whether the subproblems are solved with CP or MILP. One possible reason why the ESC is more beneficial to *InvMILP-CP* than *InvMILP-MILP* in this chapter's experiments may be that CP algorithms encounter suboptimal feasible solutions more frequently than MILP, resulting in the ESC being triggered more frequently with CP. Additionally, after an optimal solution has been found, CP often requires much longer to prove the optimality of the solution than MILP. Therefore, avoiding the proof stage with the ESC may be more beneficial for CP than for MILP.



(a) n = 20

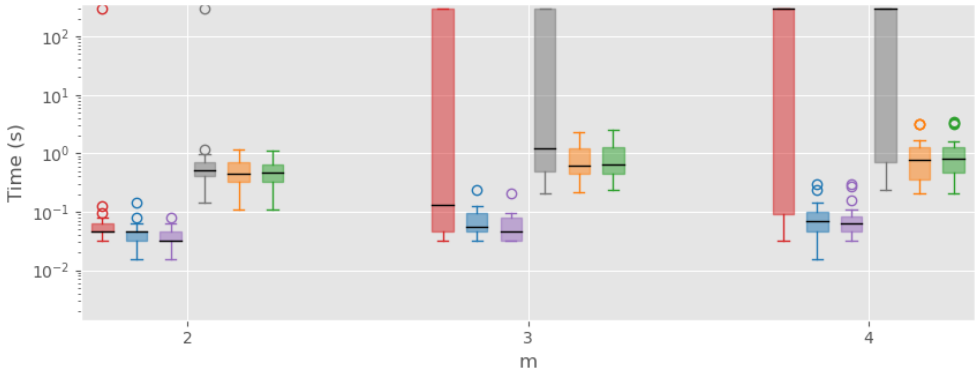


(b) n = 30

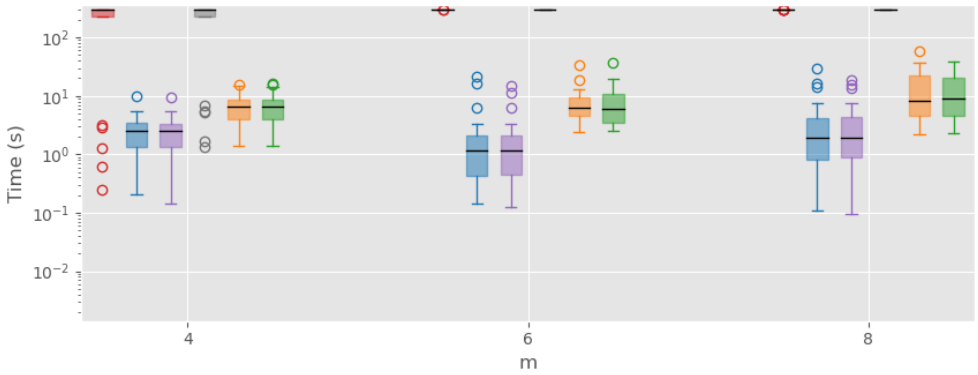


(c) n = 40

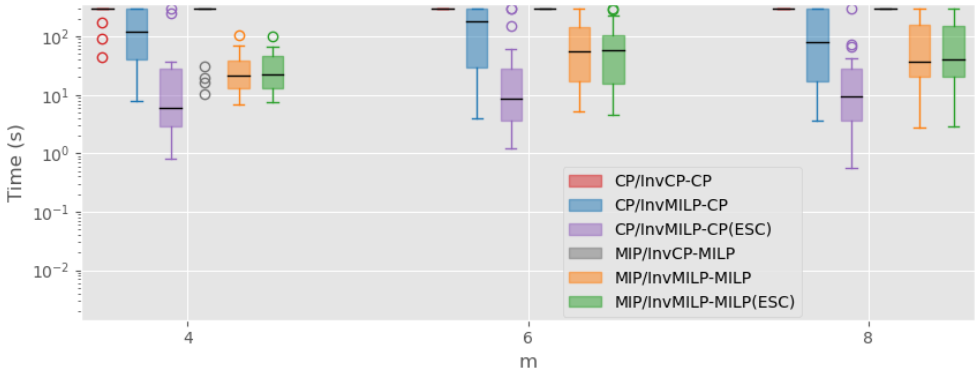
Figure 5.3: KP PA-NCE Solve Time Distributions



(a) n = 5



(b) n = 10



(c) n = 15

Figure 5.4: Single Machine Scheduling PA-NCE Solve Time Distributions

Table 5.1: Number of 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						MILP					
Inv Alg		CP-CP		MILP-CP		MILP-CP (ESC)		CP-MILP		MILP-MILP		MILP-MILP (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

5.6.3 CP Master Problems

CP performs drastically worse than MILP when it is used to solve the master problem: *CP/InvCP-CP* and *MILP/InvCP-MILP* both reach the time limit on most instances (Figure 5.2a and 5.2b). MILP likely has a considerable advantage over CP because the master problem has a simple linear structure for which MILP is very well suited. However, in future work, CP may be useful in the master problem to model more complex explanation problems, for instance by expressing more complicated constraints on the allowable objective parameters in $\mathcal{D}$.

5.6.4 Instance Breakdown

Tables 5.1 and 5.2 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. Unfortunately, there were a large number of infeasible PA-NCEs for the larger scheduling instances, even though the instance generation approach (Section 5.5.1) aimed at reducing this number. This result warrants more investigation into methods for dealing with infeasible NCE instances, a topic discussed further in Chapter 6. In contrast to single machine scheduling, no PA-NCE instances were infeasible for KP.

5.7 Conclusion

This chapter developed solution approaches to a subset of NCEs in which the contrastive question describes alternative value assignments to a subset of decision variables. The resulting counterfactual explanations are valid when the explainee is interested specifically in possible changes to the objective

Table 5.2: 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									MILP								
Inv Alg		CP-CP			MILP-CP			MILP-CP (ESC)			CP-MILP			MILP-MILP			MILP-MILP (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

parameters corresponding to that variable subset. It was shown that for these explanation problems, a single foil is guaranteed to be optimal with respect to all other foils for all values of $d \in \mathcal{D}$, alleviating the difficulties in needing to track multiple foils. This observation led to a two-step method to generate explanations that involved first solving the forward problem $\mathcal{FW}\langle c, X_\psi\rangle$ and then using the result as the target solution in an inverse optimization problem. The case of explaining forward constraint programs was also considered, leading to the development of inverse CP for the second stage of the solution approach. Wang's cutting plane algorithm, *InvLP-MILP* [10], was modified for this purpose, resulting in a pure inverse CP algorithm and two inverse MILP-CP hybrids. Finally, numerical experiments showed that explanations can be generated in a reasonable amount of time, and that an inverse MILP-CP hybrid can outperform alternatives when CP is state-of-the-art for the underlying forward problem.

Chapter 6

Solving General Nearest Counterfactual Explanation Problems

6.1 Introduction

In this chapter, an algorithm is presented for solving Nearest Counterfactual Explanation (NCE) that can explain forward problems with linear objectives and constraints. The restrictions of the previous two chapters are lifted: specifically, no objective parameters are required to remain fixed at their initial values in an explanation, and the foil constraints are no longer required to take the form of either a partial assignment (Chapter 5) or a linear constraint on a single variable (Chapter 4). In this chapter, though the constraints defining X must be linear, the foil constraints can take the form of any linear or quadratic constraint set. The new algorithm further modifies Wang's cutting plane method [10], introducing quadratic terms into the constraints of the master problem (2.13) - (2.16) to address the possibility of non-singleton foil sets. To implement this new algorithm, this chapter takes advantage of recent advances in optimization solvers: specifically a new feature in Gurobi 9.0+ which allows non-convex quadratic expressions to be modeled directly. The new algorithm is tested on two forward problems, the 0-1 KP and single machine scheduling problems. It is demonstrated that, if the foil set is non-empty, the sufficient conditions for the feasibility of an NCE derived in Chapter 3 can be applied effectively.

6.2 Solution Methodology

This section presents an algorithm, *NCXplain* (for Nearest Counterfactual eXplanation), Algorithm 6.1, for solving NCEs where the forward problems have linear objectives and constraints and the foil constraints are either linear or quadratic. It is developed by modifying the cutting plane approach of *InvLP-MILP* (Algorithm 2.1) [10], specifically, by modifying the master problem (2.13) - (2.16) and the condition used for cut generation and termination. While this chapter focuses on mixed-integer linear programming (MILP) forward problems, a near-term extension is to consider cases when the forward problems are constraint programs. Thus, as in Chapter 5, it is assumed that forward problems have finite feasible sets, meaning the dual constraints (2.14) are not necessary because there is no possibility of unbounded objectives (see Section 5.3.1).

6.2.1 The *NCXplain* Algorithm

Letting $\mathcal{S}$ be the set of all extreme points of $conv(X)$, and decision vector $x \in X_\psi$ be a foil, the $\mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ (3.1)-(3.2) can be expressed as the quadratic problem:

$$\min_{d,x} \ ||d - c||_1 \tag{6.1}$$

$$\text{s.t. } d \cdot x \leq d \cdot x^0 \qquad \forall\ x^0 \in \mathcal{S} \tag{6.2}$$

$$x \in X_\psi \tag{6.3}$$

$$d \in \mathcal{D}. \tag{6.4}$$

Constraints (6.2) force a foil to give a forward objective no worse than any extreme point of $conv(X)$, and have a left-hand side which is bilinear (thus non-convex) in variables d and x.

As in *InvLP-MILP*, the objective can be linearized using $d = c - g + h$, where $g, h \in \mathbb{R}^n$. Then, relaxing constraints (6.2) by replacing $\mathcal{S}$ with a set of known extreme points $\mathcal{S}^0 \subseteq \mathcal{S}$ gives the master problem for *NCXplain*, denoted $\mathcal{MP}_{\mathcal{NCE}}$:

$$\min_{g,h,x} \ g + h \tag{6.5}$$

$$\text{s.t. } (c - g + h) \cdot x \leq (c - g + h) \cdot x^0 \qquad \forall\ x^0 \in \mathcal{S}^0 \tag{6.6}$$

$$x \in X_\psi \tag{6.7}$$

$$(c - g + h) \in \mathcal{D}. \tag{6.8}$$

Observe that since $\mathcal{S}^0 \subseteq \mathcal{S}$, the $\mathcal{MP}_{\mathcal{NCE}}$ is either a relaxation of the NCE (6.1)-(6.4) (if $\mathcal{S}^0 \subset \mathcal{S}$) or equivalent to the NCE (if $\mathcal{S}^0 = \mathcal{S}$). In either case, if a solution (d^*, x^*), where $d^* = c - g^* + h^*$, is optimal to $\mathcal{MP}_{\mathcal{NCE}}$ and d^* satisfies constraint (3.2),

$$\min_{x \in X_\psi} d^* \cdot x = \min_{x \in X} d^* \cdot x$$

(i.e. d^* is feasible to the NCE), then d^* must be optimal to the NCE. Furthermore, if $\mathcal{MP}_{\mathcal{NCE}}$ is found to be infeasible, the NCE must also be infeasible.

Given a solution (d^i, x^i) which is optimal to $\mathcal{MP}_{\mathcal{NCE}}$ at iteration i of *NCXplain*, whether d^i satisfies (3.2) can be checked using the optimal solution $x^{0,i}$ to the subproblem $\mathcal{SP}\langle d^i, X\rangle = \mathcal{FW}\langle d^i, X\rangle$. If d^i is not shown to be feasible to the NCE at iteration i, then $x^{0,i}$ is added to $\mathcal{S}^0$, producing a new cut in the master problem.

The complete *NCXplain* algorithm is given by Algorithm 6.1. Both the master problem and the subproblem can be expressed directly in Gurboi 9.0+ due to recent advances which allow bilinear constraints such as (6.6) to be modelled directly.

Algorithm 6.1: *NCXplain*.

Inputs: $\mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$
Output: d^*
Step 1: Initialize $\mathcal{S}^0 \leftarrow x^*$.
Step 2: Solve $\mathcal{MP}_{\mathcal{NCE}}$.
    If infeasible:
        Return *INFEASIBLE*.
    Else:
        Get solution (d^i, x^i), with $d^i = (c - g^i + h^i)$.
Step 3: Solve $\mathcal{SP}\langle d^i, X\rangle$ to get solution $x^{0,i}$.
    If $d^i \cdot x^{0,i} = d^i \cdot x^i$ (Case 1):
        Stop and return $d^* = d^i$.
    Else if $d^i \cdot x^{0,i} < d^i \cdot x^i$ and $x^{0,i} \in X_\psi$ (Case 2):
        Stop and return $d^* = d^i$.
    Else (Case 3):
        Update $\mathcal{S}^0 = \mathcal{S}^0 \cup \{x^{0,i}\}$ and return to Step 2.

Theorem 6.1. Given an instance of $\mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$, the *NCXplain* algorithm will, in a finite number of iterations, either solve the NCE to optimality or prove the NCE is infeasible.

Proof. (Theorem 6.1). $\mathcal{MP}_{\mathcal{NCE}}$ is a relaxation of the NCE (6.1)-(6.4) when $\mathcal{S}^0 \subset \mathcal{S}$ and equivalent to it when $\mathcal{S}^0 = \mathcal{S}$. Thus, in a given iteration if $\mathcal{MP}_{\mathcal{NCE}}$ is infeasible, then the NCE must also be infeasible, and if a solution d^* is optimal to $\mathcal{MP}_{\mathcal{NCE}}$ and feasible to the NCE, then d^* must also be optimal to the NCE.

Each iteration of the subproblem $\mathcal{SP}\ \langle d^i, X\rangle$ (Step 3) either terminates the algorithm or adds a new extreme point to $\mathcal{S}^0$, and since the set $\mathcal{S}$ of extreme points of $conv(X)$ is finite, an iteration of $\mathcal{MP}_{\mathcal{NCE}}$ must eventually be reached when $\mathcal{S}^0 = \mathcal{S}$ if the algorithm does not terminate in a prior iteration. To prove Theorem 6.1, it remains to be shown that:

1. *NCXplain* only terminates in Step 3 if d^i from the preceding $\mathcal{MP}_{\mathcal{NCE}}$ solution, (d^i, x^i), is feasible to the NCE.

2. If an iteration of $\mathcal{MP}_{\mathcal{NCE}}$ is reached where $\mathcal{S}^0 = \mathcal{S}$ and the $\mathcal{MP}_{\mathcal{NCE}}$ is not infeasible, then *NCXplain* will terminate in the next iteration of Step 3.

To prove (1), consider the stopping conditions in Step 3, given an $\mathcal{MP}_{\mathcal{NCE}}$ solution (d^i, x^i) and an optimal solution $x^{0,i}$ to $\mathcal{SP}\ \langle d^i, X\rangle$. If $d^i \cdot x^{0,i} = d^i \cdot x^i$ (Case 1), then the foil x^i is optimal to $\mathcal{FW}\langle d^i, X\rangle$, so d^i is feasible to the NCE. Otherwise, if $d^i \cdot x^{0,i} < d^i \cdot x^i$ and $x^{0,i} \in X_\psi$ (Case 2), then $x^{0,i}$, the optimal solution to $\mathcal{FW}\langle d^i, X\rangle$, satisfies the foil constraints and d^i is feasible to the NCE.

To prove (2), observe that an optimal solution $x^{0,i}$ to the subproblem $\mathcal{SP}\ \langle d^i, X\rangle$ for any $d^i \in \mathcal{D}$ must be an extreme point of $conv(X)$ and thus a member of $\mathcal{S}$. Therefore, if $\mathcal{S}^0 = \mathcal{S}$ in an iteration of $\mathcal{MP}_{\mathcal{NCE}}$, the solution (d^i, x^i) must satisfy $d^i \cdot x^i \leq d^i \cdot x^{0,i}$. Since x^i cannot give a better subproblem objective than $x^{0,i}$, then $d^i \cdot x^{0,i} = d^i \cdot x^i$ and *NCXplain* must terminate. □

Because the number of extreme points of $conv(X)$ must be finite to guarantee the finite termination of *NCXplain*, the scope of this chapter is limited to forward problems with linear constraints. However, the foil constraints which define X_ψ can be any set of linear or quadratic constraints.

6.3 Experimental Method

Similarly to Chapter 5, experiments testing *NCXplain* are performed in three steps, focusing on the last:

1. Solving an instance of a forward problem $\mathcal{FW}\langle c, X\rangle$ to get x^*.

2. Simulating a contrastive question "Why x^* and not an $x \in X_\psi$?" to create an instance of an $\mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$.

3. Solving $\mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ with *NCXplain*.

6.3.1 Forward Problems

The same two forward problems are used for experiments as in Chapter 5: the 0-1 Knapsack Problem (KP, Example 4.2) and single machine scheduling with release dates, $1|r_j|\sum w_j C_j$ (Section 5.4.2). The MILP formulations (5.6) and (5.10) - (5.14) were used to represent the two forward problems, respectively.[1]

6.3.2 Contrastive Questions

Knapsack Problem

For the KP experiments, given a subset of knapsack items $\mathcal{S}_\psi \subset \{1, ..., n\}$, $\mathcal{S}_\psi \neq \emptyset$, the contrastive question asks why some minimal number of items from $\mathcal{S}_\psi$ were not included in the knapsack. Specifically, given a parameter $\beta_\psi \in (0, 1]$, and letting $m = |\mathcal{S}_\psi|$, the contrastive question is "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

$$X_\psi = \{x \in X : \sum_{j \in \mathcal{S}_\psi} x_j \geq \beta_\psi m\}. \tag{6.9}$$

Recall from Section 3.2 that X_ψ must be defined so that $x^* \notin X_\psi$.

Example 6.1. Consider a KP being solved to determine which people will receive access to a limited service, where w_j and c_j, respectively, represent the cost of providing the service and the expected benefit for person j. Letting $\mathcal{S}_\psi$ represent a group of 100 people that has been identified as disadvantaged ($m = 100$), a contrastive question of the form above with $\beta_\psi = 0.75$ is "Why were at least 75 people from the disadvantaged group not selected to receive the service?". The nearest counterfactual explanation would then represent the minimal total change to the profits of all the people in the system so that the optimal solution lies in X_ψ.

Scheduling Problem

For the scheduling problem, the constrastive question asks why m jobs $\mathcal{M} \subseteq \{1, ..., n\}$ were not scheduled earlier. Specifically, letting $t^* \in [0, T]^n$ represent the start times in the initial solution x^*, a vector $t^\psi \in [0, T]^m$ was created with t^ψ_j representing the maximal counterfactual start time of job $j \in \mathcal{M}$ such that $r_j \leq t^\psi_j < t^*_{\mathcal{M}_j}$. Then, the contrastive question asked "Why was each job $j \in \mathcal{M}$

[1] See Chapter 7 for comments on the possibility of using constraint programming formulations in future work.

not completed by $(t_j^\psi + q_j)$, respectively?". This question is represented with the foil set

$$X_\psi = \{x \in X : \sum_{t=0}^{T-q_j} t x_{j,t} \leq t_j^\psi \quad \forall\, j \in \mathcal{M}\}. \tag{6.10}$$

Example 6.2. Consider the $1|r_j|\sum w_j C_j$ problem being solved to schedule jobs $\mathcal{J} = \{1, ..., 10\}$ on a machine at a factory, with $c \in [1, 10]^{10}$ representing importance levels assigned to the jobs. In the initial completion times $(t^* + q)$, measured in hours after the schedule starts, jobs $\mathcal{M} = \{2, 3\}$ are scheduled to complete at times $[100, 200]$, respectively. A manager asks, "Why not complete job 2 in under 50 hours and job 3 in under 100 hours?". This question is represented with a foil set of the form (6.10) with $t_2^\psi = 50 - q_2$, $t_3^\psi = 100 - q_3$. Assuming the manager's query results in a feasible NCE,[2] the explanation reveals the minimal modifications to the job importance levels that would result in job 2 completing in under 50 hours and job 3 completing in under 100 hours.

6.3.3 Counterfactual Objectives

The only parameter in the $\mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ that remains to be defined is the set of feasible counterfactual objectives $\mathcal{D}$. This set is defined as

$$\mathcal{D} = \{d \in \mathbb{N}_0^n : 0 \leq d_i \leq c_i^{UB}\ \forall i \in \{1, ..., n\}\}, \tag{6.11}$$

where $c_i^{UB} \in \mathbb{N}$ is the maximum value possible for parameter c_i in a forward problem instance (see Section 6.4.1 for c_i value distributions).

6.3.4 NCE Feasibility

Sufficient conditions for the feasibility of NCE instances are now described. In contrast to Chapter 5, it is possible to use these conditions to prevent NCE instances from being infeasible. The first condition is that $X_\psi \neq \emptyset$. That is, the explainee must describe a set of alternative solutions that includes at least one feasible solution. Next, the sufficient conditions for NCE feasibility described by Theorem 3.2 (repeated below) are revisited.

[2] Sections 6.3.4 and 6.4 discuss how infeasible NCE instances can be avoided.

Theorem 3.2. An $\mathcal{NCE}\langle c, \mathcal{D}, \psi, x^*, X\rangle$ has a non-trivial feasible solution if both following conditions are met:

1. For all $i \in \{1, ..., n\}$, let $x_{i,\min} = \min_x\{x_i : x \in X\}$ and $x_{i,\max} = \max_x\{x_i : x \in X\}$.
   - If the forward objective is minimization (i.e. the NCE is an $\text{NCE}_{\min}$), then the foil set X_ψ must include at least one point $\tilde{x}^\psi$ such that $\exists\ j \in \{1, ..., n\}$ for which $\tilde{x}^\psi_j = x_{j,\min}$.
   - If the forward objective is maximization (i.e. the NCE is an $\text{NCE}_{\max}$), the foil set X_ψ must include at least one point $\tilde{x}^\psi$ such that $\exists\ j \in \{1, ..., n\}$ for which $\tilde{x}^\psi_j = x_{j,\max}$.
2. $\mathcal{D}^f_j \subseteq \mathcal{D}$ given
$$\mathcal{D}^f_j = \{d \in \mathbb{R}^n : 0 < d_j \leq d^{UB}_j, d_i = 0\ \forall\ i \neq j\},$$
   where $d^{UB}_j = \max_d\{d_j : d \in \mathcal{D}\}$.

If $X_\psi \neq \emptyset$, it is shown that any NCE in the knapsack and scheduling experiments will meet the conditions in Theorem 3.2, and thus have a non-trivial feasible solution. Looking at the second condition of Theorem 3.2, it is simple to see that $\mathcal{D}^f_j \subseteq \mathcal{D}$ for any $j \in \{1, ..., n\}$ when $\mathcal{D}$ is given by (6.11) and $\mathcal{D}^f_j$ by (3.3). Thus, it remains to be shown that the first condition of the theorem is also satisfied by the NCE described in Sections 6.3.1 - 6.3.3 with non-empty foil sets.

For an NCE based on a maximization forward objective such as the 0-1 KP objective, the first condition of Theorem 3.2 requires that there exists a feasible foil $\tilde{x}^\psi$ with at least one component $\tilde{x}^\psi_j$ equal to its maximal feasible value in X. In the 0-1 KP the maximal value of any variable is 1, and given the foil constraints in (6.9), any foil $x \in X_\psi$ must assign a value of 1 to at least one component x_j. Therefore, both conditions of Theorem 3.2 are satisfied by the KP based NCEs in the experiments.

Similarly, for an NCE based on a minimization forward objective such as the $1|r_j|\sum w_jC_j$ objective, the first condition of Theorem 3.2 is that there exists a feasible foil $\tilde{x}^\psi$ with at least one component $\tilde{x}^\psi_j$ equal to its minimal feasible value in X. For the scheduling problem, taking any schedule $x \in X_\psi$ (6.10) and left-shifting it causes the first job in the schedule, which will be called job j, to start at its release date r_j. Since r_j is the minimal value of x_j for any schedule $x \in X$, the scheduling based NCEs also satisfy both conditions of Theorem 3.2.

Intuitively, if $d \in \mathcal{D}^f_j$ is a feasible NCE solution, then in the case of the 0-1 KP this d implies that there is no benefit from including any items in the knapsack other than item j. Similarly, in the case of the $1|r_j|\sum w_jC_j$ problem, such a d implies that there is no benefit from the earlier completion of any jobs other than job j.

6.4 Experimental Data

This section describes instance generation details for the forward problem and NCE instances, as well as computational details.

6.4.1 Forward Instances

Knapsack Problem

0-1 KP instances of sizes $n \in \{250, 500, 1000\}$ were generated using the same distributions for item profits, weights, and knapsack capacity as in Section 5.5.1. These instance sizes result in $[500, 1000, 2000]$ quadratic terms in constraints (6.6) of $\mathcal{MP}_{\mathcal{NCE}}$, respectively.[3]

Scheduling Problem

Instances of the $1|r_j|\sum w_j C_j$ problem 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 $T = \lfloor \alpha Q \rfloor + Q$. It can be shown that the tested values of n correspond to average values of approximately [225, 530, 965] binary decision variables in the forward problem and [450, 1060, 1930] quadratic terms in constraints (6.6) in $\mathcal{MP}_{\mathcal{NCE}}$,[4] which are similar values to those in the 0-1 KP experiments.

6.4.2 Contrastive Question Instances

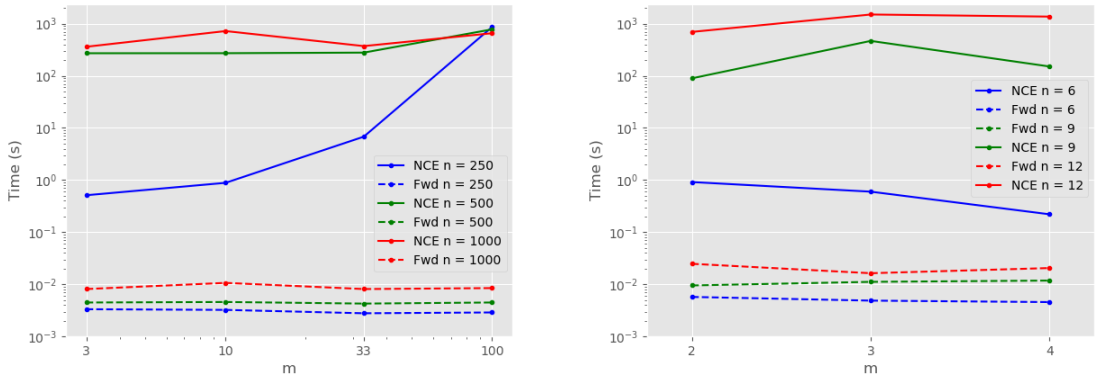
The process for generating the contrastive questions is described below. After a question was generated, it was checked whether the resulting foil set contained at least one feasible solution. If the foil set was found to be empty, the data was re-randomized until a non-empty foil set was found, though such cases were rare.

Knapsack Problem

The contrastive questions for the 0-1 KP were generated with $\beta = 0.75$ by randomly selecting m items to form the set $\mathcal{S}_\psi$, such that x^* did not satisfy the foil constraint (6.9).

[3] There are two quadratic terms in constraints (6.6) for each variable x_i: $g_i x_i$ and $h_i x_i$.

[4] The MILP model for $1|r_j|\sum w_j C_j$ uses $T - q_i$ binary variables for each job $i \in \{1, ..., n\}$, and the average value of T across all instances for a given n is approximately $T_\mu \approx (1 + \alpha)Q_\mu$, where Q_μ is the average sum of processing times. Since the average processing time is $p_\mu = 5.5$, then $Q_\mu = q_\mu n = 5.5n$, and $T_\mu \approx (1.3)(5.5n) = 7.15n$. Thus, the average number of forward decision variables is $n_{\mu,DV} \approx n(T_\mu - q_\mu) \approx 7.15n^2 - 5.5n$.



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

Figure 6.1: Mean Solve Times for NCE and Forward Problems

Scheduling Problem

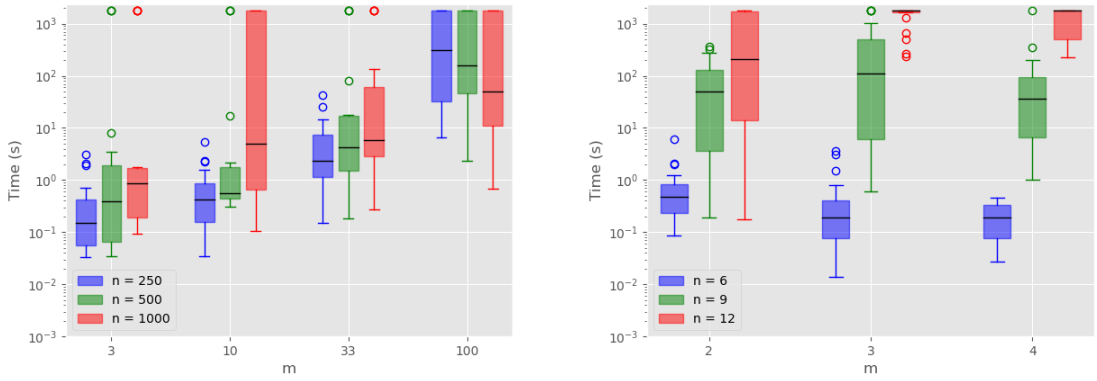
The contrastive questions for the $1|r_j|\sum w_j C_j$ problem were created by randomly selecting m jobs to form the set $\mathcal{M}$. For each job $j \in \mathcal{M}$, the maximal counterfactual start time t_j^{ψ} used in (6.10) was randomly selected from the interval $[t_j^{\psi,LB}, t_j^{\psi,UB}]$, where

$$t_j^{\psi,UB} = t^* - 1,$$
$$t_j^{\psi,LB} = \lceil r_j + \theta(t_j^* - 1 - r_j) \rceil,$$

and θ was a parameter set to 0.5.

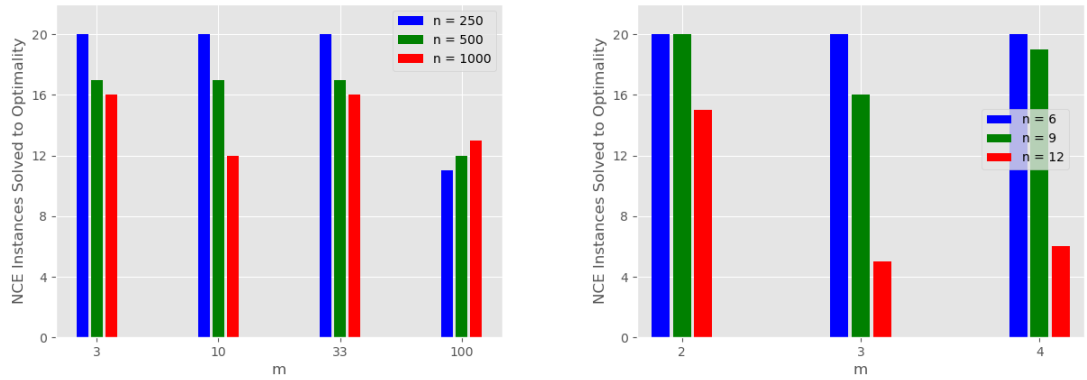
6.4.3 Computational Details

Twenty instances were tested for each value of (n, m). All algorithms were implemented in Gurobi 9.5 and tested on a single core of a 2.6 GHz Intel Core i7-10750H CPU. A time limit of 30 minutes was used for the NCE algorithm, and if an NCE instance was not solved before this time limit, its runtime was recorded as 30 minutes. Thus, the aggregated NCE solution times should be interpreted as lower bounds on the true solution times, unless all instances of a given size were solved by the time limit, in which case the reported times are the true solution times.



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

Figure 6.2: NCE Solve Time Distributions



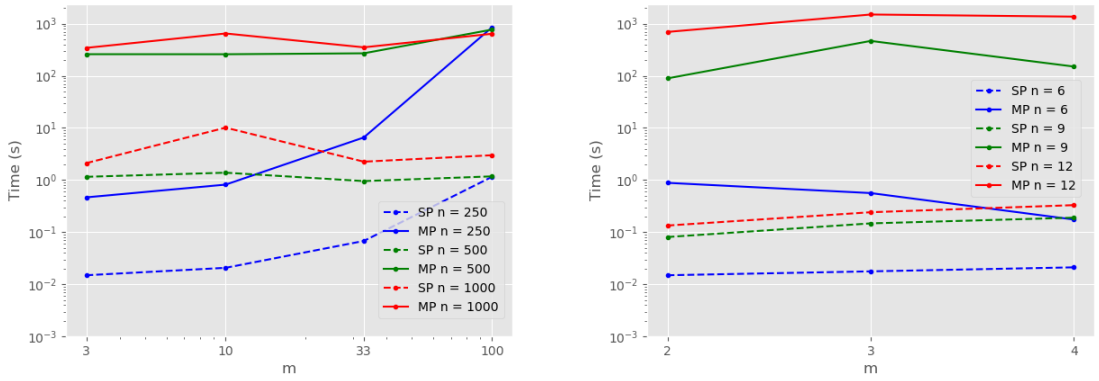
(a) 0-1 KP (b) Single Machine Scheduling

Figure 6.3: Number of NCE Instances Solved to Optimality

6.5 Results

Figure 6.1 illustrates the mean solution times for the NCE instances using *NCXplain* as well as the mean MILP solution times for the forward problem instances being explained, and the distributions of the NCE solution times are represented with box plots in Figure 6.2. Figure 6.3 shows the number of NCE instances that were solved to optimality; if an instance was not solved to optimality, it is because *NCXplain* reached the time limit. These two figures show that for forward instances with approximately 500 binary decision variables or less ($n \leq 500$ for KP, $n \leq 9$ for Scheduling),[5] most nearest counterfactual explanations could be found in under 30 minutes. These instances sizes involve up to approximately 1000 quadratic terms in constraints (6.6) in $\mathcal{MP}_{\mathcal{NCE}}$.

[5]Section 6.4.1 discusses the number of binary decision variables in the $1|r_j|\sum w_j C_j$ problem.



(a) 0-1 KP

(b) Single Machine Scheduling

Figure 6.4: Cumulative Time in NCE Master Problem vs Subproblem

These results also show that contrastive questions which are fairly expressive can be addressed effectively by *NCXplain*. For instance, in the 0-1 KP experiments that simulated the contrastive question "Why were at least 75% of a custom set $\mathcal{S}_\psi$ of 33 items not included?", 88% of NCE instances were solved to optimality within 30 minutes, and when the custom set included 100 items, 60% of NCE instances were solved optimally within 30 minutes. The use of *NCXplain* allows contrastive questions to be considerably more expressive than the questions in Chapter 5 which must be represented with partial assignments or in Chapter 4 which must be represented with a single linear constraint.

Figure 6.4 breaks down the mean time to solve an NCE instance into the cumulative time in the master problem ($\mathcal{MP}_{\mathcal{NCE}}$) and the cumulative time in the subproblem ($\mathcal{SP}$). Clearly, much more time is spent in the master problem than in the subproblem, with *NCXplain* spending at least 90% of its time in the master problem and up to 99.95% for larger instances. Thus, future improvements to *NCXplain* should focus on reducing the total time spent solving the master problem. Since it is much faster to add a new point to $\mathcal{S}^0$ (by solving $\mathcal{SP}$) than to solve $\mathcal{MP}_{\mathcal{NCE}}$, it may be worth pursuing a modification to *NCXplain* which can add multiple points to $\mathcal{S}^0$ for one iteration of $\mathcal{MP}_{\mathcal{NCE}}$, and thus reduce the number of iterations of $\mathcal{MP}_{\mathcal{NCE}}$ required. It is also worth performing a polyhedral study for *NCXplain* similar to that done by Bodur *et al.* [85] for the *InvLP-MILP* algorithm (see Section 2.6.2).

No NCE instances were found to be infeasible, consistent with Section 6.3.4 which showed that the experimental setup guarantees the existence of an explanation for all problem instances given that a non-empty foil set is provided. The ability to avoid infeasible NCE instances when the

conditions in Section 6.3.4 are met is an advantage of the *NCXplain* explanation approach over the PA-NCE explanation approach of Chapter 5, where experiments resulted in large numbers of infeasible NCEs for some instance sizes.

Finally, Figure 6.1a shows that for KP instances of size $n = 250$, increasing the value of m resulted in more difficult NCE instances. Increasing the value of m increases the number of decision variables involved in the foil constraints (6.3), but further investigation is needed to understand how changes in m effect NCE difficulty before any generalizations can be made.

6.6 Conclusion

This chapter introduced the *NCXplain* algorithm, which is able to solve NCE problems based on forward objectives and constraints which are linear, and foil constraints which are linear or quadratic. Specifically, this algorithm takes advantage of new features in Gurobi 9.0+, using a cutting plane approach that involves bilinear constraints (6.3) in the master problem. The process for explaining forward problem solutions using a contrastive question and *NCXplain* was demonstrated with simulations based on the 0-1 KP and $1|r_j|\sum w_j C_j$ problems. Furthermore, given a non-empty foil set, it was shown how meeting a set of conditions guaranteed the feasibility of NCE instances and the existence of an explanation. The numerical experiments showed that, for tested instances with 500 or fewer forward decision variables, a nearest counterfactual explanation could usually be computed in under 30 minutes. Most importantly, it was shown that *NCXplain* can handle significantly more expressive contrastive questions and places fewer restrictions on the objective parameters used in an explanation than the methods in the previous two chapters.

Chapter 7

Conclusions and Future Work

This chapter summarizes the contributions of this thesis and discusses directions for future work.

7.1 Summary of Contributions

This thesis applies the technique of counterfactual explanations to optimal discrete optimization decisions. Specifically, after an explainee asks why the optimal decision was not different in some way, the explanation takes the form of the minimal change to the objective parameters such that an alternative solution which is different in the way specified becomes optimal.

The problem of finding an explanation is formalized in Chapter 3 as the Nearest Counterfactual Explanation (NCE) problem (3.1)-(3.2), which can be interpreted as a variant of inverse optimization (Section 2.6.1). The goal of an NCE is to explain an optimal soltuion x^* to a forward problem $\mathcal{FW}\langle c, X\rangle$. The NCE allows an explainee to express a contrastive question "Why was x^* not different?" by implicitly defining a set of alternative solutions called the foil set using a custom constraint set (the foil constraints). The explanation that the NCE searches for is as minimal modification to the objective parameters such that a solution in the foil set becomes optimal to the forward problem with the modified objective. It is noted that the NCE is a generalization of classical inverse optimization; in the latter, a single target solution must become optimal, while in the former, it is sufficient for *any* solution in the foil set to become optimal. Next, solution methods are presented for two useful subsets of the NCE (Chapters 4 and 5) and then for general NCEs based on linear forward objectives and constraints (Chapter 6).

The first type of NCE that a solution method is developed for involves a contrastive question that asks why a single variable x_j^* did not satisfy a linear foil constraint $x_j^* \leq m_0$ and gives an

explanation in terms of changes to the objective parameter c_j (Chapter 4). The primary application for these univariate NCEs (Definition 4.1) is explaining optimization problems where an explainee j is represented with exactly one decision variable x_j and one objective parameter c_j, and desires an explanation that concerns only themselves. When the forward problem involves only binary decision variables, it is shown that explanations can be computed in closed form after the solution of a single modified forward problem, $\mathcal{FW}\langle c, X_\psi\rangle$. When the forward problem contains only integer variables, it is shown that an explanation can be found after at most a logarithmic number of solutions to a pair of modified forward problems.

The next subset of NCEs considered, called Partial Assignment NCEs (PA-NCEs, Definition 5.1), assumes that an explainee asks why the variables in a subset of decision variables were not assigned to specified values, and wants an explanation in terms of hypothetical changes to the objective parameters associated with the variables in this subset (Chapter 5). PA-NCEs are intended to be especially well-suited to cases when the explainee does not have complete information or control over all the objective parameters in a problem. It is shown that in a PA-NCE, a single foil $x^\psi \in X_\psi$ is guaranteed to remain optimal with respect to all other foils for any counterfactual objective vector. Then, it is demonstrated that the PA-NCE can be solved with a two-step method by solving the forward problem $\mathcal{FW}\langle c, X_\psi\rangle$ to find the optimal foil x^ψ, and then solving a classical inverse optimization problem with x^ψ as the target solution.

For the inverse optimization step, in addition to implementing a version of Wang's cutting plane algorithm, *InvLP-MILP* (Algorithm 2.1) [10], Chapter 5 also introduces several variations of this algorithm that use constraint programming (CP). The resulting CP based inverse algorithms allow explanations to be found for forward problems that are constraint programs. Numerical experiments are performed testing the PA-NCE explanation approach for the 0-1 Knapsack and Single Machine Scheduling with Release Dates forward problems. In addition to validating the PA-NCE explanation approach, these experiments show that an inverse MILP-CP hybrid can outperform alternative inverse algorithms when CP is state of the art for the underlying forward problem. The experiments also revealed that one of the challenges of the PA-NCE explanation approach is that it can sometimes be difficult to ask contrastive questions that result in feasible PA-NCE instances.

Finally, Chapter 6 introduces an algorithm, *NCXplain* (Algorithm 6.1), for solving NCE problems with forward objectives and constraints which are linear, and foil constraints which are linear or quadratic. *NCXplain* enables explainees to ask considerably more expressive questions than the methods in Chapters 4 and 5, and does not require extra restrictions on which objective parameters can be used in the explanation. The algorithm follows a similar cutting plane strategy to Wang's

inverse MILP algorithm [10], but uses a master problem with bilinear quadratic constraints to account for the feasibility of the NCE when *any* solution in the foil set becomes optimal to the forward problem. Simulations are performed using the same two forward problems as in Chapter 5, testing the *NCXplain* approach and also demonstrating how a simple set of conditions are sufficient to produce feasible NCE instances.

7.2 Future Work

Several directions for further research are now described, broadly divisible into two categories: 1) extensions based on inverse optimization concepts, and 2) extensions inspired by the ML counterfactual explanation literature (reviewed in Section 2.3.3).

7.2.1 Inverse Optimization Based Extensions

Finding feasible NCE solutions before proving optimality. One of the limitations of the PA-NCE and *NCXplain* approaches is that no feasible NCE solutions are produced before the algorithms terminate. Duan and Wang [88] observed a similar issue with the *InvLP-MILP* algorithm, which fails to produce intermediate inverse solutions. This observation led them to extend *InvLP-MILP* with a heuristic that parallelizes cut generation and also iteratively computes feasible solutions as upper bounds for the inverse MILP. In the PA-NCE approach, Duan and Wang's extended algorithm [88] could directly replace the *InvLP-MILP* base for the inverse algorithms, allowing feasible explanations to be provided before the inverse algorithm terminates. It is also worth investigating whether a similar heuristic could be implemented for *NCXplain*, noting that if the conditions in Theorem 3.2 are satisfied, at least one feasible NCE solution is already known prior to starting the algorithm.

Improving CP master problem performance. The CP implementation of the inverse master problem (Section 5.3) performed very poorly compared to the MILP implementation (see Figure 5.1), and it is worth investigating the root cause of this poor performance. Since CP algorithms rely on intermediate feasible solutions, there is a possibility that implementing Duan and Wang's heuristic [88] would improve CP performance because feasible inverse solutions would be provided for the master problem.

Minimizing Foil Sub-optimality This thesis assumes the explainee is only interested in understanding why no foils were *optimal*. Thus, if an NCE is infeasible, no explanation can be given and

the explainee is told "A foil will never be optimal given the possible modifications to the objective." However, if no foil can be made optimal, the explainee may wish to know which objective vector $d \in \mathcal{D}$ minimizes the sub-optimality of the foil set. Techniques addressing similar problems have been developed in inverse optimization, in which the sub-optimality of the target solution is minimized if the target solution cannot be made optimal (e.g. [101, 102, 103]). It is worth investigating if these approaches can be adapted to minimize the sub-optimality of a foil set instead of a single target solution. If such techniques are developed, if an NCE is infeasible, an explainee could be told "Though no foil can be made optimal, if the objective was d instead of c, a foil would be as close to optimal as possible."

Explanations Using Constraint Parameters While explanations of optimization decisions in this thesis are expressed only in terms of counterfactual objective parameters, an explainee may also want an explanation in terms of hypothetical changes to constraint parameters. A challenge of computing such constraint-based explanations is that the feasible set X may change with the constraint parameters. However, Chan and Kaw [93] have recently developed approaches for solving inverse linear optimization problems where constraint parameters are allowed to change. A direction for future work is to investigate whether similar approaches could be designed to address variations of the NCE where the constraint parameters are allowed to change.

7.2.2 Extensions Inspired by Explanations in Machine Learning

Diversity One of the ideas proposed by Wachter *et al.* [8] is that, sometimes, an explainee may desire not a single counterfactual explanation but a diverse set of explanations. If a method is developed to produce intermediate, feasible NCE solutions before optimality is proven (see Section 7.2.1), a simple way to generate multiple explanations would be to show the user some of these intermediate NCE solutions. In the work of Russel [57] on counterfactual explanations for classifiers, multiple iterations of explanation generation are performed, with a constraint added to each subsequent iteration preventing previously encountered explanations from being returned. A similar approach could be taken for NCE based explanations, where, after finding an optimal explanation, the algorithm would not terminate but rather continue to look for alternative explanations. In another work on counterfactual explanations for classifiers, Mothilal *et al.* [58] search for a set of alternative explanations such that the diversity of the set is maximized while the change to the initial parameters, aggregated across the set, is minimized. It may be possible to design a variation of the NCE that implements a similar idea, searching for a set of explanations while balancing the

objectives of maximizing diversity and minimizing perturbation to the initial parameters.

Sparsity Another idea proposed by Wachter *et al.* [8] is that sparsity can be beneficial for counterfactual explanations because it is easier to understand a few large changes to few parameters rather than many small changes to many parameters. In future work, it is worth investigating if a term that rewards sparsity, such as the L_0 norm, could be added to the NCE objective.

Actionability Finally, another idea explored in the machine learning literature [59, 60] is that, in certain settings, counterfactual explanations should prioritize actionability. That is, explanations should prioritize parameters which are easier for the explainee to change, and avoid parameters which the explainee has no control over. Chapters 4 and 5 already demonstrated how NCE-based explanations can be prevented from featuring parameters the explainee has no control over by using additional constraints on $\mathcal{D}$. However, an idea for future work is to replace the simple L_1 norm in the NCE objective (3.1) with a weighted L_1 norm, with each norm weight representing the difficulty of changing a parameter.

Minimizing Difference from Initial Decision One of the limitations discussed in Section 3.4 is that the optimal solution to $\mathcal{FW}\langle d, X\rangle$ could be arbitrarily far from x^*, making it difficult to modify the objective vector in practice because large changes to the initial solution could be problematic. A direction for future work is to study a variant of the NCE where a term minimizing the change to x^* is added to the objective. For instance, the term $||x^* - x||_1$ might be added to the master problem of *NCXplain* (6.5)-(6.8). A similar loss term is present in the objective (2.1) studied by Wachter *et al.* [8] for classifier explanations.

7.2.3 Conclusion

This thesis constitutes the first application of counterfactual explanations for optimal solutions of discrete optimization problems. Such explanations support counterfactual reasoning, a fundamental human reasoning strategy [56], across these complex problems. By allowing explainees to find counterfactual objectives which lead to counterfactual solutions, these explanations improve a person's ability to understand the effect of objective parameters on discrete optimization problems. Furthermore, this knowledge may enable a person to act to change these parameters, thus improving human-AI interaction in discrete optimization contexts. In addition, explainees may be empowered to contest decisions based on objectives that they believe were assigned unfairly. Finally, this thesis

also lays the groundwork for several directions for future work. These directions include technical improvements to enhance algorithmic performance, but also more ambitious ideas such as generating multiple diverse explanations or explanations based on minimizing a measure of foil suboptimality.

Bibliography

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

[2] 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.

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

[4] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, *et al.*, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)," in *International conference on machine learning*, pp. 2668–2677, PMLR, 2018.

[5] Z. C. Lipton, "The mythos of model interpretability," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[6] S. Wachter, B. Mittelstadt, and L. Floridi, "Why a right to explanation of automated decision-making does not exist in the general data protection regulation," *International Data Privacy Law*, vol. 7, no. 2, pp. 76–99, 2017.

[7] F. Doshi-Velez and M. Kortz, "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] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *Harv. JL & Tech.*, vol. 31, p. 841, 2017.

[9] S. Verma, J. Dickerson, and K. Hines, "Counterfactual explanations for machine learning: A review." NeurIPS Workshop on ML Retrospectives, Surveys and Meta-Analyses, 2020.

[10] L. Wang, "Cutting plane algorithms for the inverse mixed integer linear programming problem," *Operations Research Letters*, vol. 37, no. 2, pp. 114–116, 2009.

[11] B. Kim, C. Rudin, and J. A. Shah, "The bayesian case model: A generative approach for case-based reasoning and prototype classification," in *Advances in neural information processing systems*, pp. 1952–1960, 2014.

[12] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pp. 80–89, IEEE, 2018.

[13] E. Freuder, "Explaining ourselves: human-aware constraint reasoning," in *AAAI*, 2017.

[14] U. Junker, "Preferred explanations and relaxations for over-constrained problems," in *AAAI*, 2004.

[15] O. Guieu and J. W. Chinneck, "Analyzing infeasible mixed-integer and integer linear programs," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 63–77, 1999.

[16] J. Van Loon, "Irreducibly inconsistent systems of linear inequalities," *European Journal of Operational Research*, vol. 8, no. 3, pp. 283–288, 1981.

[17] M. H. Liffiton and K. A. Sakallah, "Algorithms for computing minimal unsatisfiable subsets of constraints," *Journal of Automated Reasoning*, vol. 40, no. 1, pp. 1–33, 2008.

[18] A. Felfernig, M. Schubert, and C. Zehentner, "An efficient diagnosis algorithm for inconsistent constraint sets," *AI EDAM*, vol. 26, no. 1, pp. 53–62, 2012.

[19] J. Marques-Silva, F. Heras, M. Janota, A. Previti, and A. Belov, "On computing minimal correction subsets," in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[20] 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.

[21] E. Gamba, B. Bogaerts, and T. Guns, "Efficiently explaining CSPs with unsatisfiable subset optimization," *arXiv preprint arXiv:2105.11763*, 2021.

[22] A. Yelamanchili, J. Agrawal, S. Chien, J. Biehl, A. Connell, U. Guduri, J. Hazelrig, I. Ip, K. Maxwell, K. Steadman, *et al.*, "Ground-based automated scheduling for the mars 2020 rover," 2020.

[23] I. Senthooran, M. Klapperstueck, G. Belov, T. Czauderna, K. Leo, M. Wallace, M. Wybrow, and M. G. de la Banda, "Human-Centred Feasibility Restoration," in *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)* (L. D. Michel, ed.), vol. 210 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 49:1–49:18, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

[24] R. Borgo, M. Cashmore, and D. Magazzeni, "Towards providing explanations for AI planner decisions." Presented at the IJCAI/ECAI 2018 Workshop on Explainable Artificial Intelligence (XAI). Stockholm, July 2018.

[25] B. Krarup, M. Cashmore, D. Magazzeni, and T. Miller, "Model-based contrastive explanations for explainable planning." Presented at the ICAPS 2019 Workshop on Explainable Planning, 2019.

[26] M. Cashmore, A. Collins, B. Krarup, S. Krivic, D. Magazzeni, and D. Smith, "Towards explainable ai planning as a service." Presented at ICAPS Workshop on Explainable AI Planning (XAIP), 2019.

[27] 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*, 2020.

[28] R. Eifler, M. Steinmetz, A. Torralba, and J. Hoffmann, "Plan-space explanation via plan-property dependencies: Faster algorithms & more powerful properties," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 4091–4097, 2021.

[29] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "The emerging landscape of explainable automated planning & decision making.," in *IJCAI*, pp. 4803–4811, 2020.

[30] T. C. Chan, R. Mahmood, and I. Y. Zhu, "Inverse optimization: Theory and applications," *arXiv preprint arXiv:2109.03920*, 2021.

[31] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *CoRR*, vol. abs/1702.08608, 2017.

[32] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (xai)," *IEEE access*, vol. 6, pp. 52138–52160, 2018.

[33] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM CSUR*, vol. 51, no. 5, pp. 1–42, 2018.

[34] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Communications of the ACM*, vol. 63, no. 1, pp. 68–77, 2019.

[35] B. Mittelstadt, C. Russell, and S. Wachter, "Explaining explanations in AI," in *Proceedings of the conference on fairness, accountability, and transparency*, pp. 279–288, 2019.

[36] F. Rajabiyazdi and G. A. Jamieson, "A review of transparency (seeing-into) models," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 302–308, IEEE, 2020.

[37] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777, 2017.

[38] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?" Explaining the predictions of any classifier," in *ACM SIGKDD*, pp. 1135–1144, 2016.

[39] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pp. 0210–0215, IEEE, 2018.

[40] R. Tibshirani, "Regression shrinkage and selection via the lasso: a retrospective," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 3, pp. 273–282, 2011.

[41] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[42] M. H. Sqalli and E. C. Freuder, "Inference-based constraint satisfaction supports explanation," in *AAAI/IAAI, Vol. 1*, pp. 318–325, 1996.

[43] Y. Lou, R. Caruana, and J. Gehrke, "Intelligible models for classification and regression," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 150–158, 2012.

[44] J. Hoffmann and D. Magazzeni, "Explainable ai planning (XAIP): overview and the case of contrastive explanation," *Reasoning Web. Explainable Artificial Intelligence*, pp. 277–282, 2019.

[45] S. Krening, B. Harrison, K. M. Feigh, C. L. Isbell, M. Riedl, and A. Thomaz, "Learning from explanations using sentiment and advice in RL," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 1, pp. 44–55, 2016.

[46] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender systems*, pp. 165–172, 2013.

[47] A. Newell, H. A. Simon, *et al.*, *Human problem solving*, vol. 104. Prentice-hall Englewood Cliffs, NJ, 1972.

[48] G. A. Klein, "Do decision biases explain too much," *Human Factors Society Bulletin*, vol. 32, no. 5, pp. 1–3, 1989.

[49] M. S. Cohen, J. T. Freeman, and S. Wolf, "Metarecognition in time-stressed decision making: Recognizing, critiquing, and correcting," *Human factors*, vol. 38, no. 2, pp. 206–219, 1996.

[50] R. Caruana, H. Kangarloo, J. D. Dionisio, U. Sinha, and D. Johnson, "Case-based explanation of non-case-based learning methods.," in *Proceedings of the AMIA Symposium*, p. 212, American Medical Informatics Association, 1999.

[51] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[52] F. Doshi-Velez, B. Wallace, and R. Adams, "Graph-sparse LDA: a topic model with structured sparsity," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.

[53] B. Rous, "Major update to acm's computing classification system," *Communications of the ACM*, vol. 55, no. 11, pp. 12–12, 2012.

[54] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati, "Plan explanations as model reconciliation: Moving beyond explanation as soliloquy," *arXiv preprint arXiv:1701.08317*, 2017.

[55] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[56] K. Epstude and N. J. Roese, "The functional theory of counterfactual thinking," *Personality and social psychology review*, vol. 12, no. 2, pp. 168–192, 2008.

[57] C. Russell, "Efficient search for diverse coherent explanations," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 20–28, 2019.

[58] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 607–617, 2020.

[59] B. Ustun, A. Spangher, and Y. Liu, "Actionable recourse in linear classification," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 10–19, 2019.

[60] A.-H. Karimi, B. Schölkopf, and I. Valera, "Algorithmic recourse: from counterfactual explanations to interventions," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 353–362, 2021.

[61] A. Artelt and B. Hammer, "On the computation of counterfactual explanations–a survey," *arXiv preprint arXiv:1911.07749*, 2019.

[62] A. V. Looveren and J. Klaise, "Interpretable counterfactual explanations guided by prototypes," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 650–665, Springer, 2021.

[63] T. Le, S. Wang, and D. Lee, "Grace: Generating concise and informative contrastive sample to explain neural network model's prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 238–248, 2020.

[64] D. Mahajan, C. Tan, and A. Sharma, "Preserving causal constraints in counterfactual explanations for machine learning classifiers," *arXiv preprint arXiv:1912.03277*, 2019.

[65] I. Stepin, J. M. Alonso, A. Catala, and M. Pereira-Fariña, "A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence," *IEEE Access*, vol. 9, pp. 11974–12001, 2021.

[66] D. E. Smith, "Choosing objectives in over-subscription planning.," in *ICAPS*, vol. 4, p. 393, 2004.

[67] R. Eifler, M. Brandao, A. J. Coles, J. Frank, and J. Hoffmann, "Plan-property dependencies are useful: A user study." ICAPS 2021 Workshop on Explainable AI Planning, 2021.

[68] M. Brandao, A. Coles, and D. Magazzeni, "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.

[69] F. Rossi and A. Sperduti, "Acquiring both constraint and solution preferences in interactive constraint systems," *Constraints*, vol. 9, no. 4, pp. 311–332, 2004.

[70] M. H. Liffiton and A. Malik, "Enumerating infeasibility: Finding multiple muses quickly," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 160–175, Springer, 2013.

[71] J. Marques-Silva and A. Previti, "On computing preferred muses and mcses," in *International Conference on Theory and Applications of Satisfiability Testing*, pp. 58–74, Springer, 2014.

[72] D. Mehta, B. O'Sullivan, and L. Quesada, "Extending the notion of preferred explanations for quantified constraint satisfaction problems," in *International Colloquium on Theoretical Aspects of Computing*, pp. 309–327, Springer, 2015.

[73] D. E. Joslin and D. P. Clements, "Squeaky wheel optimization," *Journal of Artificial Intelligence Research*, vol. 10, pp. 353–373, 1999.

[74] Z. Ruttkay, "Fuzzy constraint satisfaction," in *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, pp. 1263–1268, IEEE, 1994.

[75] C. Heuberger, "Inverse combinatorial optimization: A survey on problems, methods, and results," *Journal of combinatorial optimization*, vol. 8, no. 3, pp. 329–361, 2004.

[76] L. Wang, "Branch-and-bound algorithms for the partial inverse mixed integer linear programming problem," *Journal of Global Optimization*, vol. 55, no. 3, pp. 491–506, 2013.

[77] 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.

[78] M. Demange and J. Monnot, "An introduction to inverse combinatorial problems," *Paradigms of Combinatorial Optimization: Problems and New Approaches*, pp. 547–586, 2014.

[79] X. Li, X. Shu, H. Huang, and J. Bai, "Capacitated partial inverse maximum spanning tree under the weighted hamming distance," *Journal of Combinatorial Optimization*, vol. 38, no. 4, pp. 1005–1018, 2019.

[80] K. Wei and V. Vaze, "Modeling crew itineraries and delays in the national air transportation system," *Transportation Science*, vol. 52, no. 5, pp. 1276–1296, 2018.

[81] A. Korikov and J. C. Beck, "Counterfactual Explanations via Inverse Constraint Programming," in *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)* (L. D. Michel, ed.), vol. 210 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 35:1–35:16, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

[82] J. Zhang and Z. Liu, "Calculating some inverse linear programming problems," *Journal of Computational and Applied Mathematics*, vol. 72, no. 2, pp. 261–273, 1996.

[83] R. K. Ahuja and J. B. Orlin, "Inverse optimization," *Operations Research*, vol. 49, no. 5, pp. 771–783, 2001.

[84] G. Iyengar and W. Kang, "Inverse conic programming with applications," *Operations Research Letters*, vol. 33, no. 3, pp. 319–330, 2005.

[85] M. Bodur, T. C. Chan, and I. Y. Zhu, "Inverse mixed integer optimization: Polyhedral insights and trust region methods," *INFORMS Journal on Computing*, 2022.

[86] A. J. Schaefer, "Inverse integer programming," *Optimization Letters*, vol. 3, no. 4, pp. 483–489, 2009.

[87] J. B. Lamperski and A. J. Schaefer, "A polyhedral characterization of the inverse-feasible region of a mixed-integer program," *Operations Research Letters*, vol. 43, no. 6, pp. 575–578, 2015.

[88] Z. Duan and L. Wang, "Heuristic algorithms for the inverse mixed integer linear programming problem," *Journal of Global Optimization*, vol. 51, no. 3, pp. 463–471, 2011.

[89] C. Audet, P. Hansen, B. Jaumard, and G. Savard, "Links between linear bilevel and mixed 0–1 programming problems," *Journal of optimization theory and applications*, vol. 93, no. 2, pp. 273–300, 1997.

[90] J. Hu, J. E. Mitchell, J.-S. Pang, K. P. Bennett, and G. Kunapuli, "On the global solution of linear programs with linear complementarity constraints," *SIAM Journal on Optimization*, vol. 19, no. 1, pp. 445–471, 2008.

[91] J. Hu, J. E. Mitchell, J.-S. Pang, and B. Yu, "On linear programs with linear complementarity constraints," *Journal of Global Optimization*, vol. 53, no. 1, pp. 29–51, 2012.

[92] J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook of applied optimization*, vol. 1, pp. 65–77, 2002.

[93] T. C. Y. Chan and N. Kaw, "Inverse optimization for the recovery of constraint parameters," *European Journal of Operational Research*, vol. 282, no. 2, pp. 415–427, 2020.

[94] K. Ghobadi and H. Mahmoudzadeh, "Inferring linear feasible regions using inverse optimization," *European Journal of Operational Research*, vol. 290, no. 3, pp. 829–843, 2021.

[95] D. Pisinger, H. Kellerer, and U. Pferschy, "Knapsack problems," *Handbook of Combinatorial Optimization*, p. 299, 2013.

[96] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer Science & Business Media, 2013.

[97] J. K. Lenstra, A. R. Kan, and P. Brucker, "Complexity of machine scheduling problems," in *Annals of discrete mathematics*, vol. 1, pp. 343–362, Elsevier, 1977.

[98] "Global Constraint Catalogue." https://sofdem.github.io/gccat/. Accessed: 2021-08-5.

[99] A. B. Keha, K. Khowala, and J. W. Fowler, "Mixed integer programming formulations for single machine scheduling problems," *Computers & Industrial Engineering*, vol. 56, no. 1, pp. 357–367, 2009.

[100] M. Pinedo, *Scheduling*, vol. 29. Springer, 2012.

[101] D. Bertsimas, V. Gupta, and I. C. Paschalidis, "Data-driven estimation in equilibrium using inverse optimization," *Mathematical Programming*, vol. 153, no. 2, pp. 595–633, 2015.

[102] A. Aswani, Z.-J. Shen, and A. Siddiq, "Inverse optimization with noisy data," *Operations Research*, vol. 66, no. 3, pp. 870–892, 2018.

[103] P. Mohajerin Esfahani, S. Shafieezadeh-Abadeh, G. A. Hanasusanto, and D. Kuhn, "Data-driven inverse optimization with imperfect information," *Mathematical Programming*, vol. 167, no. 1, pp. 191–234, 2018.