CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Counterfactual explanation through constraint relaxation
Authors	Gupta, Sharmi Dev;O'Sullivan, Barry;Quesada, Luis
Publication date	2025-01-28
Original Citation	Gupta, S.D., O'Sullivan, B. and Quesada, L. (2024) 'Counterfactual explanation through constraint relaxation', 2024 IEEE 36th International Conference on Tools with Artificial Intelligence (ICTAI), Herndon, VA, USA, 28-30 October ,pp. 396–403. 2024 .Available at: https://doi.org/10.1109/ICTAI62512.2024.00064
Type of publication	Conference item
Link to publisher's version	https://ieeexplore.ieee.org/xpl/conhome/10849351/proceeding - 10.1109/ICTAI62512.2024.00064
Rights	© 2025, IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. - http://creativecommons.org/licenses/by/4.0/
Download date	2026-01-15 05:43:40
Item downloaded from	https://hdl.handle.net/10468/17279

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Counterfactual Explanation through Constraint Relaxation

Sharmi Dev Gupta[23], Barry O'Sullivan[123], Luis Quesada[13]
[1]Insight Centre for Data Analytics
[2]SFI Centre for Research Training in Artificial Intelligence
[3]School of Computer Science & IT, University College Cork, Cork, Ireland

***Abstract*—Interactive constraint systems often suffer from infeasibility (no solution) due to conflicting user constraints. A common approach to recover feasibility is to eliminate the constraints that cause the conflicts in the system. This approach allows the system to provide an explanation as: "if the user is willing to drop some of their constraints, there exists a solution". However, this form of explanation might not be very informative. A counterfactual explanation is a type of explanation that can provide a basis for the user to recover feasibility by helping them understand what changes can be applied to their existing constraints rather than removing them. We propose an iterative method based on conflict detection and maximal relaxations in over-constrained constraint satisfaction problems to help compute a counterfactual explanation. We have evaluated our approach using well known instances that occur in industrial applications and demonstrated the relevance of multi-point relaxations.**

***Index Terms*—Counterfactual Explanation, Maximal Relaxation, Constraint Programming.**

I. Introduction

In constraint satisfaction and constraint programming an *explanation* often strives to interpret the reasons for an infeasible scenario. This interpretation mostly depends on the identification of minimal conflicts (or minimal unsatisfiable subsets). Conflicts have been studied extensively in areas such as model-based diagnosis, Boolean satisfiability, product configuration, solving logic puzzles, interactive search, etc., where the user constraints play an important role [1]. For example, when solving a scheduling problem, an explanation can provide insights to why a schedule cannot be found given the set of tasks to be completed, the due-dates, and the machine resources available, by identifying a relaxation of the problem so that one can find a feasible schedule. Recently, the need for user-centered explanations in AI has substantially increased due to several important factors such as the black-box nature of complex AI applications, the *right to explanation of a decision* in the EU's General Data Protection Regulations, and the development of Trustworthy AI for building trust between AI and the society. To address this issue, Wachter et al. proposed to use counterfactuals and adapted them to the AI domain to explain algorithmic decisions [2], [3]. They describe a *counterfactual explanation* as a statement that explains the minimal change to the system that results in a different outcome. Providing counterfactual explanations is expected to improve the understandability of the underlying model, and support decision-making process of the user. A counterfactual explanation seeks to provide a minimal explanation to a question of the form: "Why is the outcome X and not Y?" [3]. To illustrate, consider a constraint system that aims to solve the course timetabling problem at a University. The dedicated admin staff runs the timetabling system to obtain a feasible timetable. However, a lecturer, who is used to teaching their assigned course on Mondays, asks the admin: "Why is my Course A scheduled to Friday instead of Monday? I cannot attend lectures on Fridays due to travel.". In order to accommodate this user constraint, which was not a part of the system before, the admin can add this new information to the system. However, adding the new constraint may cause an infeasible state in the system. To recover from this situation, the admin can follow a traditional conflict elimination mechanism, which involves finding a set of constraints to relax so the conflicts in the problem are removed. Alternatively, the system can provide a counterfactual explanation to the admin that explains: "If you move Course B from Monday to Tuesday, you can schedule Course A on Monday.". Note that, if the user's request does not cause infeasibility, alternative explanations can be considered such as: "Given the new constraint, an alternative schedule can be found at an extra cost of C.". Counterfactual explanations have recently been adapted to optimisation problems [4]. We discuss relevant work in more detail in the Related Work section. We then propose a new approach to finding a counterfactual explanation based on identifying conflicts and maximal relaxations, demonstrate our model on a configuration problem, and conclude with a discussion and identification of some future directions. In the remainder of this paper, we first elaborate on the state of the art (Section II). We then present our approach by first introducing the notions on which our approach depends (Section III). We demonstrate the approach in Section IV and empirically evaluate it (Section V). We present some concluding remarks in Section VI.

II. Related Work

Constraint Programming has many applications in scheduling, resource allocation, product configuration, etc. Our work focuses on explanations in constraint satisfaction. The majority of existing work on this topic has focused on the identification of conflicts amongst constraints to explain infeasibility [1], [5]–[8].

In 2017, Wachter et al. proposed to use counterfactual explanations to provide a minimal amount of information capable of altering a decision without understanding the internal logic of a model [2], [3]. In a recent survey paper on counterfactuals in XAI, Keane et al. [9] presented a detailed analysis of 100 distinct counterfactual methods and their overall evaluation, and shortcomings along with a roadmap to improvement. They highlighted that only a few approaches are supported by user evaluations. Similarly, Miller argued that in XAI, a 'good explanation' is usually defined by the researchers, but the social science dimension to this definition is not explored well [10]. Miller characterized explanations provided by humans as *contrastive*, *selected* in a biased manner, *social* (i.e. transferring knowledge), and not completely based on *probabilities* (the most likely explanation is not necessarily the best explanation).

Despite the long history of explanation generation in constraint satisfaction, counterfactual explanations is a relatively new concept. However, there exist a few relevant studies that discuss related notions such as contrastive and abductive explanations in Boolean satisfiability. As an example, Ignatiev et al. have a number of studies at the intersection of ML and SAT [11], [12]. Their work discusses different types of explanations, such as *local abductive* (answering "Why prediction X?") and *contrastive* explanations (answering "Why not?"). More specifically, the authors discuss how recent approaches for computing abductive explanations can be exploited for computing contrastive/counterfactual explanations. Their findings highlight an important property that the model-based local abductive and contrastive explanations are related by minimal hitting set relationships [12].

More recently, Cooper and Marques-Silva investigate the computational complexity of finding a subset-minimal abductive or contrastive explanation of a decision taken by a classifier [13]. The authors define the explanation notions analogous to Ignatiev et al. [12]. In parallel, Cyras et al. present an extensive overview of various machine reasoning techniques employed in the domain of XAI, in which they discuss XAI techniques from the symbolic AI perspective [14]. The authors classify explanations into three categories. These are, namely, *attributive*, *contrastive*, and *actionable* explanations. Subsequently, they discuss the links between these explanation notions and the existing notions in symbolic AI by covering many different topics such as abductive logic programming, answer set programming, constraint programming, SAT, etc. They discuss that contrastive explanations can be achieved via counterfactuals and define a *counterfactual contrastive explanation* as "making or imagining different choices and analysing what could happen or could have happened". On the other hand, they define an actionable explanation as one that aims to answer "What can be done in order for a system to yield outcome o, given information i?".

Korikov et al. extend the notion of counterfactual explanations to optimisation-based decisions by using inverse optimisation [4]. In their work, the authors define counterfactual explanations analogous to those of Wachter et al. [3]. They aim to find the *nearest counterfactual explanation*, which corresponds to finding a set of changes on the features such that the new solution is as close to the previous one as possible. Subsequently, Korikov and Beck generalise their work to constraint programming and show that counterfactual explanations can be found using inverse constraint programming using a cost vector [15]. Karimi [16] along with Korikov [4] have a similar goal to generate the optimal counterfactual explanations for classifiers. Karimi however does not take into account decisions taken by explicit optimization models as opposed to Korikov.

The approach by Rebecca Eifler et al. provides an explanation framework that explains tradeoff between soft goals by identifying the conflicting soft goals and ways to resolve these conflicts [17]. Raul Mencia et al. investigate the task to explain and correct a machine scheduling problem with a limit on the makespan which makes the problem UNSAT. They compute a single arbitrary explanation and single correction efficiently [18].

The approach by Niklas Lauffer and Ufuk Topcu is close to ours [19]. They use the MARCO algorithm to generate minimal conflicts and maximal relaxations specifically for scheduling algorithms. They also modify the RCSP instances to include a given number of agents that execute tasks in parallel and introduce start and end times for each scheduling instance.

To the best of our knowledge, the most relevant study to our work has recently been conducted by Senthooran et al. who propose a problem independent approach to resolve conflicts and generate meaningful explanations. They identify the constraints causing conflicts and the smallest number of changes to be applied in order to solve the conflicts. They also conduct an evaluation of trade offs between practicality and flexibility for their approach and their meaningfulness and usefulness when used for real world applications [20]. We differ in the way we relax the problem. Their approach relies on the notion of a slack variable. Ours is defined in terms of multi-point relaxation spaces. The use of slack variables implicitly imposes a total order on the relaxations of a constraint. We allow more flexibility by letting constraints in the relaxation space be partially ordered.

In this paper, our goal is to find a counterfactual explanation to a given constraint problem by using conflicts and constraint relaxation. To achieve this, we build upon earlier work by Ferguson and O'Sullivan in which the authors generalize conflict-based explanations to Quantified CSP framework [21]. Their approach extends the famous QUICKXPLAIN algorithm [22] by allowing relaxation of constraints instead of their removal from the constraint set. We also demonstrate how this mechanism based on identification of maximal relaxations can be used to find counterfactual explanations in constraint-based systems.

III. METHODOLOGY

In this section we first define some important notions existing in the constraint programming literature on explanations. We then present the notion of counterfactual explanation, and discuss its relation to constraint relaxation. We finish the section with our proposed model to find a counterfactual explanation.

A. Conflicts, relaxations and exclusion sets

A constraint satisfaction problem (CSP) is defined as a 3-tuple $(\mathcal{V}, \mathcal{D}, \mathcal{C})$ where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is a finite set of variables, $\mathcal{D} = \{D(v_1), D(v_2), ..., D(v_n)\}$ denotes the set of finite domains where the domain $D(v_i)$ is the finite set of values that variable v_i can take, and a set of constraints $\mathcal{C} = \{c_1, c_2, ..., c_m\}$. In interactive constraint programming, the set of constraints can be divided into two sets of constraints: $\mathcal{B}$ representing the *background constraints* and $\mathcal{F}$ representing the *foreground constraints* (or *user requirements/constraints*). In order to increase readability, we sometimes omit the variables and domains and refer to a problem P as $(\mathcal{B}, \mathcal{F})$. A set of constraints is called *inconsistent* (or *unsatisfiable*) if there is no solution. In this case, the problem is said to be *infeasible*. If the problem has at least one solution, the set of constraints is said to be *consistent* (or *satisfiable*), and the related problem is referred to as *feasible*. We assume that the set of background constraints are consistent, but the user constraints may introduce infeasibility. We use Π to refer to the consistency checker. That is, $\Pi(\mathcal{C})$ is true or false depending on whether $\mathcal{C}$ is consistent or not. We define below a number of relevant definitions existing in the literature.

Definition 1 (Minimal Conflict [22]): A subset C of $\mathcal{F}$ is a conflict of a problem $P = (\mathcal{B}, \mathcal{F})$ iff $\mathcal{B} \cup C$ has no solution. A conflict C of $\mathcal{F}$ is minimal (irreducible) if each proper subset of C is consistent with the background $\mathcal{B}$ (i.e., if no proper subset of C is a conflict).

Definition 2 (Maximal Relaxation [22]): A subset $\mathcal{M}$ of $\mathcal{F}$ is a relaxation of $P = (\mathcal{B}, \mathcal{F})$ iff $\mathcal{B} \cup \mathcal{M}$ has a solution. A subset $\mathcal{M}$ of $\mathcal{F}$ is a maximal relaxation iff $\mathcal{M}$ is a relaxation and there is no $c \in \mathcal{F} \setminus \mathcal{M}$ such that $\mathcal{B} \cup \mathcal{M} \cup \{c\}$ also admits a solution.

Based on the definition of a maximal relaxation, the complementary notion of minimal exclusion set can be defined.

Definition 3 (Minimal Exclusion Set [8]): Given a problem $P = (\mathcal{B}, \mathcal{F})$ that is inconsistent, and a maximal relaxation $\mathcal{M} \subseteq \mathcal{F}$, the set $\mathcal{X} = \mathcal{F} \setminus \mathcal{M}$ is a minimal exclusion set.

B. Multi-point relaxation spaces

The previous definitions are defined under *two-point relaxation spaces* in which constraints can only be either included in, or excluded from, the set of user constraints. In this paper, we work under *multi-point relaxation spaces* which defines a set of alternative ways in which a constraint can be weakened, and ultimately excluded [21], [23]. We represent a multi-point relaxation space with a directed graph. A (directed) edge from a constraint c_1 to another constraint c_2 means that c_2 is an immediate relaxation of c_1 (i.e., there is no other c_3 such that c_3 is a relaxation of c_1 and c_2 is a relaxation of c_3). That is, if $c_2 \in suc(c_1)$ or $c_1 \in pred(c_2)$ then c_2 is an immediate relaxation of c_1. In what follows we use $maxima(\mathcal{R})$ to refer to the set of constraints in the multi-point relaxation space $\mathcal{R}$ that cannot be relaxed further. That is,

$$maxima(\mathcal{R}) = \{c \in \mathcal{R} : suc(c) = \emptyset\} \quad (1)$$

Similarly, $minima(\mathcal{R})$ refers to the set of constraints in the multi-point relaxation space $\mathcal{R}$ that cannot be tightened further. That is,

$$minima(\mathcal{R}) = \{c \in \mathcal{R} : pred(c) = \emptyset\} \quad (2)$$

When $maxima(\mathcal{R})$ is a singleton set, we use $\overline{\mathcal{R}}$ to refer to the only element in $maxima(\mathcal{R})$. Similarly, when $minima(\mathcal{R})$ is a singleton set, we use $\underline{\mathcal{R}}$ to refer to the only element in $minima(\mathcal{R})$. In the instances considered in Section IV, we have that for all computation space $\mathcal{R}$ it holds that $|maxima(\mathcal{R})| = |minima(\mathcal{R})| = 1$. That is both $\overline{\mathcal{R}}$ and $\underline{\mathcal{R}}$ are well defined. $\underline{\mathcal{R}}$ corresponds to the original foreground constraint and $\overline{\mathcal{R}}$ to its total relaxation. We define $sp_{\mathcal{R}}(c_1, c_2)$ as the shortest path from c_1 to c_2 in $\mathcal{R}$[1]. When $\mathcal{R}$ is obvious from the context, we omit $\mathcal{R}$. A relaxation space implicitly defines an order among the constraints of the space. We say $c_1 \sqsubseteq c_2$ if there is a path from c_1 to c_2. If the path involves one or more edges we say $c_1 \sqsubset c_2$.

C. Relaxation Space using Allen's Algebra

Allen's interval algebra is one of the best known and widely used formalism for representing interval-based qualitative temporal information [24]. The fundamental reasoning task in Allen's interval algebra is to find a scenario that is consistent with the given information. This problem is in general NP-complete [25]. Table I shows the mathematical definition of the basic 13 relations of Allen's interval algebra. In what follows, we use $\mathcal{A}$ to refer to the set of basic relations. The equations in Table I specify the conditions under which the relations hold. For example, the 'before' relation holds between tasks x and y when $x_{end} < y_{start}$.

We use Allen's interval algebra to define the relation between various events in a scheduling problem. We compute a counterfactual explanation in a scheduling problem where the constraints are expressed in terms of Allen's interval algebra and the relaxation spaces are generated from $\mathcal{A}$.

Constraint	Symbol	Definition
$equals(x, y)$	$=$	$\{(x_{\text{start}}, y_{\text{start}}) :=, (x_{\text{end}}, y_{\text{start}}) :>, (x_{\text{end}}, y_{\text{end}}) :=\}$
$before(x, y)$	$<$	$\{(x_{\text{start}}, x_{\text{end}}) :<, (x_{\text{end}}, y_{\text{start}}) :<, (y_{\text{start}}, y_{\text{end}}) :<\}$
$meets(x, y)$	m	$\{(x_{\text{start}}, x_{\text{end}}) :<, (x_{\text{end}}, y_{\text{start}}) :=, (y_{\text{start}}, y_{\text{end}}) :<\}$
$overlaps(x, y)$	o	$\{(x_{\text{start}}, y_{\text{start}}) :<, (x_{\text{end}}, y_{\text{start}}) :>, (x_{\text{end}}, y_{\text{start}}) :<\}$
$starts(x, y)$	s	$\{(x_{\text{start}}, y_{\text{start}}) :=, (x_{\text{end}}, y_{\text{start}}) :>, (x_{\text{end}}, y_{\text{end}}) :<\}$
$during(x, y)$	d	$\{(x_{\text{start}}, y_{\text{start}}) :>, (x_{\text{start}}, x_{\text{end}}) :<, (x_{\text{end}}, y_{\text{end}}) :<\}$
$finishes(x, y)$	f	$\{(x_{\text{start}}, y_{\text{start}}) :>, (x_{\text{start}}, y_{\text{end}}) :<, (x_{\text{end}}, y_{\text{end}}) :=\}$
$overlapped_by(x, y)$	oi	$\{(x_{\text{start}}, y_{\text{start}}) :>, (x_{\text{start}}, y_{\text{end}}) :<, (x_{\text{end}}, y_{\text{end}}) :>\}$
$started_by(x, y)$	si	$\{(x_{\text{start}}, y_{\text{start}}) :=, (x_{\text{start}}, x_{\text{end}}) :<, (x_{\text{end}}, y_{\text{end}}) :>\}$
$contains(x, y)$	di	$\{(x_{\text{start}}, y_{\text{start}}) :<, (y_{\text{start}}, y_{\text{end}}) :<, (x_{\text{end}}, y_{\text{end}}) :>\}$
$finished_by(x, y)$	fi	$\{(x_{\text{start}}, y_{\text{start}}) :<, (x_{\text{end}}, y_{\text{start}}) :>, (x_{\text{end}}, y_{\text{end}}) :=\}$
$after(x, y)$	$>$	$\{(y_{\text{start}}, y_{\text{end}}) :<, (x_{\text{start}}, y_{\text{end}}) :>, (x_{\text{start}}, x_{\text{end}}) :<\}$
$met_by(x, y)$	mi	$\{(y_{\text{start}}, y_{\text{end}}) :<, (x_{\text{start}}, y_{\text{end}}) :=, (x_{\text{start}}, x_{\text{end}}) :<\}$

TABLE I: Allen's Interval Algebra

There could be a total of $2^{|\mathcal{A}|}$ possible relations that are disjunctions of the basic relations $\mathcal{A}$. Out of these $2^{|\mathcal{A}|}$, we only select 27. Two of these 27 are the top ($\overline{\mathcal{R}}$) and bottom ($\underline{\mathcal{R}}$) nodes of the relaxation space. Algorithm 1 shows the steps for the generation of a relaxation space $\mathcal{R}$ containing the relaxations.

[1]There can be more than one shortest path from one constraint to another in a space. We assume an order among the edges in the space and break the ties accordingly.

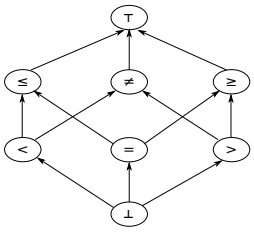

Fig. 1: PO: a partial order on the numerical relations used in the definitions of $\mathcal{A}$. An edge from a relation r_i to a relation r_j means that r_j is a relaxation of r_i.

Algorithm 1 SPACEGEN($\mathcal{A}, PO$)

Require: Allen's Interval Algebra relations $\mathcal{A}$, Partial order on numerical relations (PO) as shown in Figure 1.
Ensure: Relaxation space $\mathcal{R}$

$\mathcal{R} = \emptyset$
$\forall r \in \mathcal{A}$, $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\overline{\mathcal{R}_i}, r), (r, \underline{\mathcal{R}_i})\}$
while $\mathcal{A} \neq \emptyset$ **do**
  $d = \emptyset$
  for all combinations of pairs $(r_i, r_j) \in \mathcal{A}$ **do**
    $rels_i \leftarrow$ ONESTEPRELAXATION(r_i, PO)
    $rels_j \leftarrow$ ONESTEPRELAXATION(r_j, PO)
    for $r' \in rels_i$ **do**
      if $r' \in rels_j$ **then**
        $d[r_i \vee r_j] = r'$
        $\mathcal{R} \leftarrow \mathcal{R} \cup \{(r_i \vee r_j, r_i), (r_i \vee r_j, r_j), (\overline{\mathcal{R}_i}, r_i \vee r_j), (r_i \vee r_j, \underline{\mathcal{R}_i})\}$
        break
  $\mathcal{A} = d$
return $\mathcal{R}$

When we generate the relaxations for all relations in $\mathcal{A}$, we use *one step relaxation* to limit the number of possible relaxations generated for each relation. We introduce *one step relaxation* as the modified mathematical definition of $\mathcal{A}$ after relaxing (only once) one of the inequality operators. Algorithm 2 shows the detailed logic for generating *one step relaxation* for any given relation. Table II shows one such example: constraints $before(x, y)$ and $meets(x, y)$ are relaxed and their relaxations are compared. Since they have a common relaxation, both $before(x, y)$ and $meets(x, y)$ can be connected through an edge to a common node $beforeORmeets$ (in this case) which represents the constraints relation.

All relations in $\mathcal{A}$ go through this process until we form the next level of nodes representing the common relaxations,

Algorithm 2 ONESTEPRELAXATION(r_m, PO)

Require: Relation to be relaxed r_m, Partial order on numerical relations PO
Ensure: Set of all one-step relaxations rel_m

$rel_m = \emptyset$
for $p \in r_m$ **do**
  for $(r1, r2) \in PO$ **do**
    if $r_m[p] = r1$ **then**
      $r'_m \leftarrow \emptyset$
      for $p' \in r_m$ **do**
        $r'_m[p'] \leftarrow r2$ **if** $p' = p$ **else** $r_m[p']$
      $rel_m \leftarrow rel_m \cup \{r'_m\}$
return rel_m

Constraint	Definition	Relaxations
$before(x, y)$	$\{(x_{\text{start}}, x_{\text{end}}) :<, (x_{\text{end}}, y_{\text{start}}) :<, (y_{\text{start}}, y_{\text{end}}) :<\}$	$\{(x_{\text{start}}, x_{\text{end}}) :\leq, (x_{\text{end}}, y_{\text{start}}) :<, (y_{\text{start}}, y_{\text{end}}) :<\}$, $\{(x_{\text{start}}, x_{\text{end}}) :<, (x_{\text{end}}, y_{\text{start}}) :\leq, (y_{\text{start}}, y_{\text{end}}) :<\}$, $\{(x_{\text{start}}, x_{\text{end}}) :<, (x_{\text{end}}, y_{\text{start}}) :<, (y_{\text{start}}, y_{\text{end}}) :\leq\}$
$meets(x, y)$	$\{(x_{\text{start}}, x_{\text{end}}) :<, (x_{\text{end}}, y_{\text{start}}) :=, (y_{\text{start}}, y_{\text{end}}) :<\}$	$\{(x_{\text{start}}, x_{\text{end}}) :\leq, (x_{\text{end}}, y_{\text{start}}) :=, (y_{\text{start}}, y_{\text{end}}) :<\}$, $\{(x_{\text{start}}, x_{\text{end}}) :<, (x_{\text{end}}, y_{\text{start}}) :\leq, (y_{\text{start}}, y_{\text{end}}) :<\}$, $\{(x_{\text{start}}, x_{\text{end}}) :<, (x_{\text{end}}, y_{\text{start}}) :=, (y_{\text{start}}, y_{\text{end}}) :\leq\}$

TABLE II: Multiple one step relaxations for $before(x, y)$ and $meets(x, y)$ and the common relaxation $beforeORmeets(x, y)$ between them

the one step relaxations that are not common between any relations are discarded. The relations, that do not have any common relaxations with any other relation are connected to $\overline{\mathcal{R}}$. Once we connect all relations in $\mathcal{A}$ to either the common node or $\overline{\mathcal{R}}$, we repeat the above process for this newly formed level of nodes. This process is continued until there are no more common relaxations between the nodes, in which case all the nodes without any outgoing edges are connected to $\overline{\mathcal{R}}$. Figure 2 shows the final relaxation space generated using $\mathcal{A}$.

D. Finding a counterfactual explanation

We define a counterfactual explanation by adapting the definitions from Wachter et al. [3] and Korikov et al. [4]. We aim to find an explanation to the user with minimal changes to her constraints and inform her on how to recover from an infeasible state.

Definition 4 (Counterfactual Explanation): Given a problem $P = (\mathcal{B}, \mathcal{F})$ that is inconsistent, a maximal relaxation $\mathcal{M} \subseteq \mathcal{F}$, and the corresponding minimal exclusion set $\mathcal{X} = \mathcal{F} \setminus \mathcal{M}$, a counterfactual explanation, $\mathcal{E}$, for P is a relaxation of the constraints in $\mathcal{X}$ such that $\mathcal{B} \cup \mathcal{M} \cup \mathcal{E}$ is consistent. A minimal counterfactual explanation is one such that none of its constraints can be tightened further without introducing an inconsistency.

Algorithm 3 COUNTERFACTUALXPLAIN(P, $\mathscr{R}$)

Require: A problem $P = (\mathcal{B}, \mathcal{F})$, where $\mathcal{F} = \{c_1, \ldots, c_n\}$, and a set $\mathscr{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ of multi-point relaxation spaces of each user constraint in $\mathcal{F}$.
Ensure: A counterfactual explanation.

if $\Pi(\mathcal{B} \cup \mathcal{F})$ **then return** no conflict
if $\forall i \in \{1, \ldots, n\} | \mathcal{R}_i| = 1$ **then return** no relaxation
$\mathcal{M} \leftarrow \emptyset$
for $c_i \in \mathcal{F} : \Pi(\mathcal{B} \cup \mathcal{M} \cup \{c_i\})$ **do** $\mathcal{M} \leftarrow \mathcal{M} \cup \{c_i\}$
$\mathcal{X} \leftarrow \mathcal{F} \setminus \mathcal{M}$
Let $\mathcal{R}_1 \ldots \mathcal{R}_m$ be the spaces associated with $\mathcal{X}$ and set r_i to $\overline{\mathcal{R}_i}$
while $\exists_{i \in 1 \ldots m}. |\mathcal{R}_i| > 1$ **do**
  while $\exists_{i \in 1 \ldots m}. pred(r_i) \neq \emptyset \wedge \Pi(\mathcal{B} \cup \mathcal{M} \cup \{r_1 \ldots r_m\})$ **do**
    choose an i s.t. $pred(r_i) \neq \emptyset$
    $r'_i \leftarrow r_i$
    choose an r_i from $pred(r_i)$
  if $\Pi(\mathcal{B} \cup \mathcal{M} \cup \{r_1 \ldots r_m\})$ **then return** $\{r_1 \ldots r_m\}$
  $\mathcal{R}_i \leftarrow \{r \mid r \in \mathcal{R}_i, r \not\sqsubseteq r_i\}$
  $r_i \leftarrow r'_i$
  foreach $\mathcal{R}_j$ **do** $\mathcal{R}_j \leftarrow \{r \mid r \in \mathcal{R}_j, r \sqsubseteq r_j\}$
return $\{r_1 \ldots r_m\}$

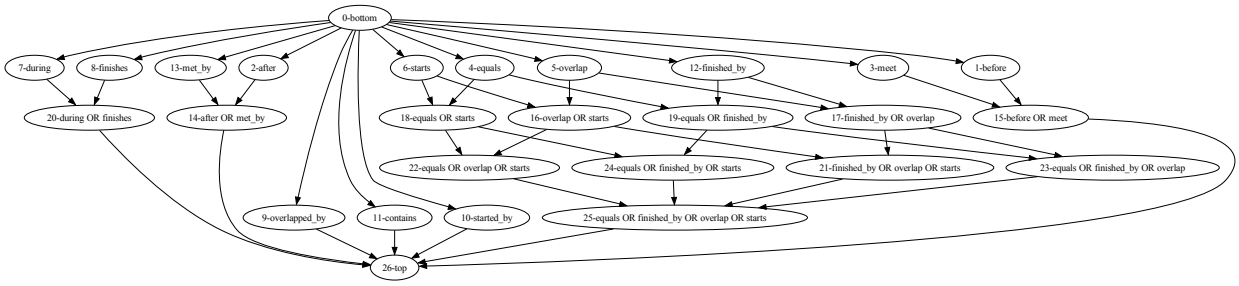

Fig. 2: Relaxation Space

E. CounterFactualXplain

We present in Algorithm 3 the details of our proposed method COUNTERFACTUALXPLAIN. This approach is an adaptation of the QUANTIFIEDXPLAIN algorithm that was proposed to solve Quantified CSPs following a set of different relaxation forms including single constraint relaxation, relaxation of existentially/universally quantified domain, quantifier relaxation, etc. [21]. From the set of different relaxation forms they propose, we only adapt single constraint relaxations in our work. Our proposed method currently follows an iterative approach for identifying the counterfactual explanation for the problem. Note that, if the relaxation spaces are two-point (binary), then the algorithm becomes a version of Junker's REPLAYXPLAIN algorithm that is an iterative approach to find a minimal conflict [6]. The COUNTERFACTUALXPLAIN admits a problem $P = (\mathcal{B}, \mathcal{F})$ and the multi-point relaxation spaces of each constraint that can be relaxed $\mathcal{R}$, and returns a counterfactual explanation $\mathcal{E}$ (a set of constraints that needs to be changed to restore feasibility). If the given problem P is feasible, then the algorithm returns 'no conflict'. Similarly, if there is no relaxation space defined for the foreground constraints, the algorithm returns 'no relaxation'. For infeasible problems, the algorithm first identifies two sets of constraints: a maximal relaxation $\mathcal{M}$ and minimal exclusion set $\mathcal{X}$ (Line 4 and Line 5). This is achieved by adding each user constraint iteratively to the background constraints and checking for consistency. It is important to note that the complexity of the consistency checker directly influences the complexity of the overall algorithm. Between Lines 6 to 15 the procedure iteratively attempts to *tighten* the maximal relaxation of each constraint present in the minimal exclusion set $\mathcal{X}$ until either the original user constraint is reached or an inconsistent set of constraints is formed. As mentioned before, we see spaces as directed graphs, so *tightening* a constraint corresponds to replacing it with a predecessor. Finally, the algorithm returns a set of the maximally tightened relaxations of the minimal exclusion set.

F. Properties of CounterFactualXplain

We focus now on how the tightening of the total relaxation of $\mathcal{X}$ is achieved, and elaborate on some properties of our algorithm. One requirement of our algorithm is that, for each constraint c_i that can be relaxed, $\overline{\mathcal{R}_i}$ fully relaxes c_i for the given $\mathcal{R}_i$. At Line 6, we set the relaxation of $\mathcal{X}$ to its total relaxation. We then have two nested while loops. The outer while loop iterates while it is possible to tighten the relaxation of $\mathcal{X}$. The inner while loop iterates while the current state of relaxation (i.e., $\mathcal{B} \cup \mathcal{M} \cup \{r_1 \dots r_n\}$) is consistent and it is still possible to tighten the relaxation of $\mathcal{X}$. At each iteration of this inner while loop we choose a constraint that can be tightened (Line 9), save its current state (Line 10), and replace it with a predecessor in the space (Line 11). After leaving the inner while loop we first check the reason why we left the loop. if the relaxation of $\mathcal{X}$ could not be tightened further, we return the current relaxation (Line 12). If an inconsistency was detected, we update $\mathcal{R}_i$ by removing all those constraints that are tighter than or equal to r_i (Line 13), restore the previous r_i (Line 14), and for all space $\mathcal{R}_j$ we remove all those constraints that are weaker or incomparable to r_j (Line 15). Notice that as we are removing the incomparable constraints during the filtering, after the filtering we have that r_j is the most relaxed constraint in $\mathcal{R}_j$ for every space $\mathcal{R}_j$. The selection of the constraint to be tightened is done nondeterministically (see Line 9). We also choose from the possible alternatives to tighten a constraint in a nondeterministic way (see Line 11). The first choice ensures that no constraint is favoured. Notice that at each iteration of the inner loop we may choose a different constraint. The second choice ensures that the whole relaxation space of a constraint is considered, thus allowing the possibility of returning any valid maximal relaxation of $\mathcal{X}$.

Proposition 1: If $\{r_1 \dots r_m\}$ is a maximal relaxation of $\mathcal{X}$ it is a possible output of COUNTERFACTUALXPLAIN.

Proof: We show that this proposition holds by constructing a sequence of (choose statement) decisions that leads us to $\{r_1 \dots r_m\}$. The initial state of the relaxation of $\mathcal{X}$ is $\{\overline{\mathcal{R}_1} \dots \overline{\mathcal{R}_m}\}$. One (not necessarily unique) sequence that leads us to $\{r_1 \dots r_m\}$ is one where we consider the foreground constraints in order and tighten each constraint following the constraints in $sp_{\mathcal{R}_i}(\overline{\mathcal{R}_i}, r_i)$. That is we tighten the i-th foreground constraint before tightening the $i+1$-th foreground constraint, thus ending with a sequence whose length is equal to the sum of the length of the shortest paths.

Proposition 2: COUNTERFACTUALXPLAIN terminates.

Proof: The termination of the execution is ensured by the fact that at each iteration of the outer while loop we prune at least one constraint: the one that caused the inconsistency (r_i). That is, in the worst case, the number of iteration of the outer

while loop is bounded by the total number of constraints in the spaces. The number of ancestors of a constraint (i.e., the constraints that are tighter than the given constraint) is finite so the number of iterations of the inner while loop is finite too. As a constraint is only visited once the number of consistency checks is also bounded by the total number of constraints in the spaces.

The time complexity of COUNTERFACTUALXPLAIN depends on the complexity of the checks and therefore it is dependent on the number of consistency checks. The worst case scenario is when at each iteration we remove one element from the relaxation space of a foreground constraint so the total number of checks is bounded by the total number of constraints in the spaces. As each constraint is considered once, the number of checks is linear with respect to the number of constraints in the relaxation spaces.

IV. DEMONSTRATION

This section illustrates COUNTERFACTUALXPLAIN using one of the instances considered in the empirical evaluation. We use the relaxation space presented in Figure 2.

A. Problem

We use a modified version of the Resource Constrained Project Scheduling Problem (RCPSP). We took instance $j3022_2$ from the instance set in the paper by Ouellet et al. [26]. In the RCPSP problem, the goal is to schedule n number of tasks, where each task is using multiple resources and each resource has a capacity. The amount of resources a task consumes varies by resource and can be zero. There are precedence constraints between tasks and and there is also a given makespan. In this paper we consider a generalisation of the problem where the constraint between two tasks can be of the form $rel(t_1, t_2)$ where rel can be any of the relations in our generated relaxation space (Figure 2). We can formally present our problem as a CSP as follows:

1) Constants:

- rc is an array representing the resource capacities.
- $Tasks$ is the set of all tasks.
- d is an array representing the task durations.
- rr is a 2D array representing the resource requirements for each task and resource.
- $makespan$ is a parameter representing the makespan.

2) Variables:

- $starts$ is an array of integer variables representing the start times of tasks in the set $Tasks$. Each value in the domain represents a time point.

3) Constraints:

a) Background constraints:

- $cumulative(starts, d, [rr[r, t] \mid t \in Tasks], rc[r])$
- $\forall t \in Tasks : starts_t + d_t \leq makespan$

b) Foreground constraints:

- A set of constraints of the form $rel(t_1, t_2)$, where rel can be any of the relations in our generated relaxation space (Figure 2)

B. Instance

We modify $j3022_2$ by replacing the precedence relation between the tasks with a randomly selected relation from $\mathcal{A}$. The set of foreground constraints we have after modifying this instance are presented in Table III, where i is the ID of the foreground constraints and the Relation column represents one of the relations in $\mathcal{A}$. All other constraints of instance $j3022_2$ stay the same.

i	Task1	Task2	Relation	i	Task1	Task2	Relation
1	1	10	$overlap$	9	5	12	$finishes$
2	2	4	$before$	10	6	11	$overlapped_by$
3	2	6	$contains$	11	6	15	met_by
4	2	14	$started_by$	12	7	12	$equals$
5	3	5	$equals$	13	8	10	$starts$
6	3	7	$equals$	14	10	13	$contains$
7	3	8	$after$	15	11	12	$equals$
8	5	9	$finished_by$	16	14	15	$during$

TABLE III: Foreground Constraints

Table IV shows the final outputs of the algorithm after application on the instance above. Column i refers to the constraint ID of the foreground constraint, which is in $\mathcal{X}$. Task1 and Task2 represent the tasks. Column Foreground Constraint refers to the original relation. Column Relaxed constraint refers to the relaxation of the original relation. The constraints 3,4,5,9,12 and 16 are fully relaxed as their relaxation is $\overline{\mathcal{R}_i}$. The rest are partially relaxed.

i	Task1	Task2	Foreground constraint	Relaxed constraint
1	1	10	$overlap$	$overlap\ OR\ starts$
3	2	6	$contains$	$\overline{\mathcal{R}_3}$
4	2	14	$started_by$	$\overline{\mathcal{R}_4}$
5	3	5	$equals$	$\overline{\mathcal{R}_5}$
6	3	7	$equals$	$equals\ OR\ starts$
8	5	9	$finished_by$	$equals\ OR\ finished_by\ OR\ starts$
9	5	12	$finishes$	$\overline{\mathcal{R}_9}$
12	7	12	$equals$	$\overline{\mathcal{R}_{12}}$
13	8	10	$starts$	$overlap\ OR\ starts$
15	11	12	$equals$	$equals\ OR\ overlap\ OR\ starts$
16	14	15	$during$	$\overline{\mathcal{R}_{16}}$

TABLE IV: Counterfactual Explanation

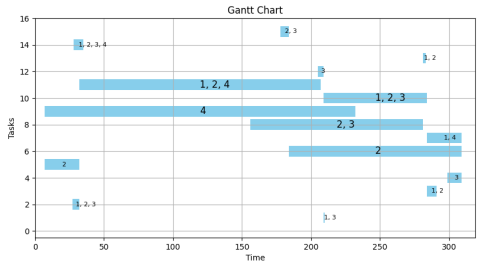

Fig. 3: Distribution of tasks

Figure 3 is a Gantt chart showing how tasks has been scheduled and the resources utilised by each task. All the tasks end by the time the makespan is reached. Figure 4 shows the resource utilisation throughout the makespan.

We can use Figure 5 to gain insight into why the relaxation computed is maximal. Notice, for instance, that reversing the relation between Tasks 1 and 10 to $overlap$ would lead to an increase (in at least one unit) of the makespan. This follows from the fact that $overlap$ forces the start of Task 10 to be

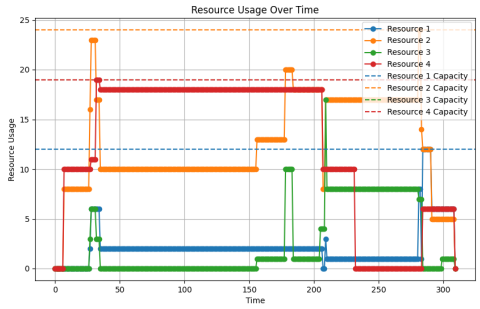

Fig. 4: Resource Utilisation Chart

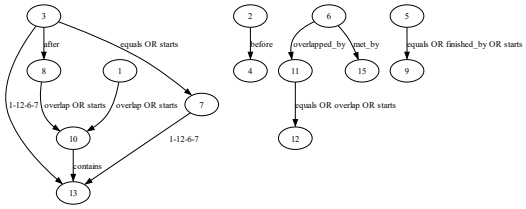

Fig. 5: Relation graph of the relaxation. Vertices represent tasks. Edges are associated with the relation between tasks. There are two set of relations: one set corresponds to the relaxation of the foreground constraints. The other set refers to tasks that need to be executed sequentially due to the capacity of the resources. For instance, the edge from Task 3 to Task 13 refers to one this cases. The label (1-12-6-7) means that the tasks share Resource 1, which has capacity 12. Task 3 uses 6 units and Task 13 uses 7 units.

greater than the start of Task 1, which also means that those tasks that cannot start before the end Task 10 (e.g., Task 7) are also affected.

V. Empirical Evaluations

We use 465 instances from the $ksd15_d$ datasets to empirically evaluate the performance of the algorithm. The suc relation in the 15 foreground constraints is first replaced by the $before$ relation from $\mathcal{A}$. We also create a second set of datasets by replacing the successor relation with any of the relations in $\mathcal{A}$. That is, $suc(t_1, t_2)$ is replace with $before(t_1, t_2)$ for the first instance set $g1$, and for the second instance set $g2$, $suc(t_1, t_2)$ is replaced with $r(t_1, t_2)$ where r is chosen randomly from $\mathcal{A}$. Algorithm 4 shows the modification process to generate an instance in the second set of instances.

The results of the evaluation of Algorithm 3 are summarised in the boxplots [27] of Figures 6, 7 and 8. The boxplots, which were computed using the DataFrame.boxplot function of Seaborn [28], show median, inter-quartile range (IQR) bounds of +/- 1.5*IQR beyond the box, and outliers. The experiments were carried out on an Intel(R) Xeon(R) CPU E5-2650 v2 machine operating at 2.60GHz with 128GB of DDR3 memory using Ubuntu 22.04 LTS. We implemented our approach in Python 3.10 using Minizinc 2.8.3 as modeling language and Chuffed 0.13.1 as underlying solver.

Algorithm 4 InstanceGen($\mathcal{B}, \mathcal{F}$)

Require: $\mathcal{F} = \{c_1, \ldots, c_n\}$ where $c_i = suc(t_1, t_2)$ and Allen's Interval Algebra relations $\mathcal{A}$
Ensure: modified RCSP instance $(\mathcal{B}, \mathcal{F}')$ where $\mathcal{F}' = \{c'_1, \ldots, c'_n\}$
1: $F'' = \{c''_1, \ldots, c''_n\}$ where $c''_i = \top$
2: **while** $\Pi(\mathcal{B} \cup F'')$ **do**
3: **for** $suc(t_1, t_2) \in \mathcal{F}$ **do**
4: choose an r from $\mathcal{A}$
5: $c''_i \leftarrow r(t_1, t_2)$
6: **if** $\neg\Pi(\mathcal{B} \cup F'')$ **then**
7: $\mathcal{F}' \leftarrow F''$
8: **return** $(\mathcal{B}, \mathcal{F}')$

Figure 6 shows the number of checks required by Algorithm 3. The number of checks increases with the number of constraints for both groups, but the role of the number of constraints is stronger in the second group.

Figure 7 shows the runtime in seconds. While there is a correlation with the number of checks, the small number of checks observed overall indicates that the runtime is dominated by the complexity of the underlying decision problem. Figure 8 shows the partial relaxations returned for the instances. We observe that very few instances get partially relaxed in $g1$. However, for all the instances in $g2$, there is at least one constraint that is partially relaxed.

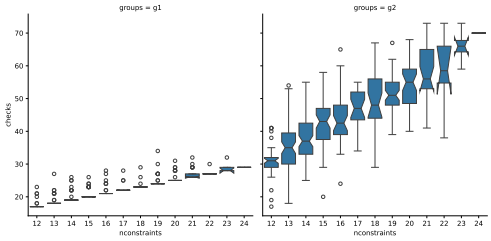

Fig. 6: Checks

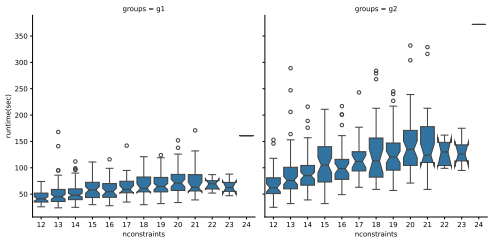

Fig. 7: Runtime

VI. Conclusion and Future Works

The aim of our approach is to find a set of changes that can be applied to the system to change the outcome (feasibility state) of the system. We proposed a novel explanation type for constraint based systems by using the counterfactual explanation framework and identifying a maximal relaxation of the constraint set. This framework aims to find a minimal set of changes for a set of user constraints using multi point relaxation spaces.

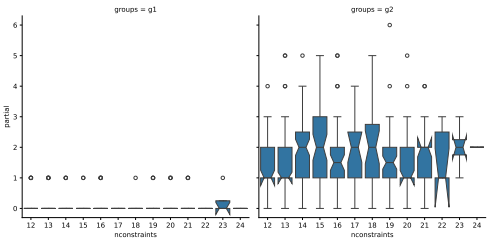

Fig. 8: Partial Relaxations

While the notion of multipoint relaxation space has been already introduced [21], in this paper we express this definition in terms of graph theory concepts. We believe this approach not only makes the concept more intuitive but also allows us to present the algorithm for computing counterfactual explanations more clearly. The presentation of the algorithm for computing counterfactual explanations is complemented with the elaboration on the correctness of the algorithm.

We used a well-known set of over-constrained RCPSP instances [26] and proposed a way to modify them using relations from Allen's algebra. Our experiments suggest that our approach scales well with respect to the number of foreground constraints, and that we can find cases where we can benefit from having partial relaxations.

An obvious direction for future work will be to undertake user-experiments using real-world problems to ensure the useability of our approach. However, since our work is inspired by the well known QuickXplain algorithm, which has been widely used in real world systems and applications, we are not concerned that the results of such a user-study would highlight computational issues.

In this paper we have focused on maximality only. However, a natural extension of our approach is to consider preferences on the foreground constraints. The preferences will certainly play a role in the selection of the foreground constraints that we choose not to relax (i.e., the constraints that we place in $\mathcal{M}$) and also on how we choose among the constraints in $\mathcal{X}$.

VII. Acknowledgement

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 18/CRT/6223 and 12/RC/2289-P2, the latter co-funded under the European Regional Development Fund. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] S. D. Gupta, B. Genc, and B. O'Sullivan, "Explanation in constraint satisfaction: A survey," in *Proceedings of IJCAI 2021*, 2021, pp. 4400–4407.

[2] S. Wachter, B. D. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *CoRR*, vol. abs/1711.00399, 2017.

[3] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: automated decisions and the gdpr," *Harvard Journal of Law and Technology*, vol. 31, no. 2, pp. 841–887, 2018.

[4] A. Korikov, A. Shleyfman, and J. C. Beck, "Counterfactual explanations for optimization-based decisions in the context of the GDPR," in *Proceedings of IJCAI 2021*, 2021, pp. 4097–4103.

[5] J. Marques-Silva and C. Mencía, "Reasoning about inconsistent formulas," in *Proceedings of IJCAI 2020*, C. Bessiere, Ed., 2020, pp. 4899–4906.

[6] U. Junker, "Quickxplain: Conflict detection for arbitrary constraint propagation algorithms," in *IJCAI'01 Workshop on Modelling and Solving problems with Constraints*, 2001.

[7] M. H. Liffiton and K. A. Sakallah, "Algorithms for computing minimal unsatisfiable subsets of constraints," *J. Autom. Reason.*, vol. 40, no. 1, pp. 1–33, 2008. [Online]. Available: https://doi.org/10.1007/s10817-007-9084-z

[8] B. O'Sullivan, A. Papadopoulos, B. Faltings, and P. Pu, "Representative explanations for over-constrained problems," in *Proceedings of AAAI*, 2007, pp. 323–328.

[9] M. T. Keane, E. M. Kenny, E. Delaney, and B. Smyth, "If only we had better counterfactual explanations: Five key deficits to rectify in the evaluation of counterfactual XAI techniques," in *Proceedings of IJCAI 2021*, 2021, pp. 4466–4474.

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

[11] A. Ignatiev, F. Pereira, N. Narodytska, and J. Marques-Silva, "A sat-based approach to learn explainable decision sets," in *Proceedings of IJCAR 2018*, D. Galmiche, S. Schulz, and R. Sebastiani, Eds., 2018, pp. 627–645.

[12] A. Ignatiev, N. Narodytska, N. Asher, and J. Marques-Silva, "On relating'why?'and'why not?'explanations," *arXiv preprint arXiv:2012.11067*, 2020.

[13] M. C. Cooper and J. Marques-Silva, "On the tractability of explaining decisions of classifiers," in *Proceedings of CP 2021*, 2021.

[14] K. Cyras, R. Badrinath, S. K. Mohalik, A. Mujumdar, A. Nikou, A. Previti, V. Sundararajan, and A. V. Feljan, "Machine reasoning explainability," 2020.

[15] 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)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[16] A.-H. Karimi, G. Barthe, B. Balle, and I. Valera, "Model-agnostic counterfactual explanations for consequential decisions," in *Proceedings of the International Conference on AI and Statistics*, 26–28 Aug 2020, pp. 895–905.

[17] R. Eifler, J. Hoffmann, and J. Frank, "Explaining soft-goal conflicts through constraint relaxations," in *31st International Joint Conference on Artificial Intelligence*, 2022.

[18] R. Mencía Cascallana, C. Mencía Cascallana, J. Marqués Silva *et al.*, "Efficient reasoning about infeasible one machine sequencing," in *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, 2023.

[19] N. Lauffer and U. Topcu, "Human-understandable explanations of infeasibility for resource-constrained scheduling problems," in *ICAPS 2019 Workshop XAIP*, 2019.

[20] I. Senthooran, M. Klapperstueck, G. Belov, T. Czauderna, K. Leo, M. Wallace, M. Wybrow, and M. Garcia de la Banda, "Human-centred feasibility restoration in practice," *Constraints*, vol. 28, no. 2, pp. 203–243, 2023.

[21] A. Ferguson and B. O'Sullivan, "Quantified constraint satisfaction problems: From relaxations to explanations." in *IJCAI*, 2007, pp. 74–79.

[22] U. Junker, "QuickXplain: preferred explanations and relaxations for over-constrained problems," in *Proceedings of AAAI 2004*, 2004, pp. 167–172.

[23] D. Mehta, B. O'Sullivan, and L. Quesada, "Extending the notion of preferred explanations for quantified constraint satisfaction problems," in *Proc ICTAC*, 2015, pp. 309–327.

[24] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[25] D. N. Pham, J. Thornton, and A. Sattar, "Modelling and solving temporal reasoning as propositional satisfiability," *Artificial Intelligence*, vol. 172, no. 15, pp. 1752–1782, 2008.

[26] Y. Ouellet and C.-G. Quimper, "The softcumulative constraint with quadratic penalty," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, pp. 3813–3820.

[27] R. Mcgill, J. W. Tukey, and W. A. Larsen, "Variations of box plots," *The American Statistician*, vol. 32, no. 1, pp. 12–16, 1978.

[28] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: https://doi.org/10.21105/joss.03021