Finding Counterfactual Explanations through Constraint Relaxations

Sharmi Dev Gupta
SFI Centre for Research Training in AI
School of Computer Science & IT, University College Cork, Ireland

Begüm Genç
Insight Centre for Data Analytics
School of Computer Science & IT, University College Cork

Barry O'Sullivan
SFI Centre for Research Training in AI
Insight Centre for Data Analytics
School of Computer Science & IT, University College Cork
Confirm Centre for Smart Manufacturing

Abstract

Interactive constraint systems often suffer from infeasibility (no solution) due to conflicting user constraints. A common approach to recover infeasibility 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 out some of their constraints, there exists a solution". However, one can criticise this form of explanation as not being 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 which changes can be applied to their existing constraints rather than removing them. This approach has been extensively studied in the machine learning field, but requires a more thorough investigation in the context of constraint satisfaction. We propose an iterative method based on conflict detection and maximal relaxations in over-constrained constraint satisfaction problems to help compute a counterfactual explanation.

2012 ACM Subject Classification Computing methodologies → Search methodologies

Keywords and phrases Counterfactual Explanation, Maximal Relaxation, Constraint Programming

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.0

1 Introduction

In the long-standing history of constraints, 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 [4]. When solving a scheduling problem, an explanation can provide insights to why the given problem is not feasible under the provided sets of background and foreground constraints, and removing which set of constraints can provide a relaxation to the problem such that one can find a feasible solution. However these explanations are not always produced for the user, but sometimes produced for speeding up the search or debugging for the developer.

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 from philosophy, and adapt them

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 0; pp. 0:1–0:9
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

to the AI domain to explain algorithmic decisions [19, 18]. They describe a *counterfactual explanation* as a statement that explains the minimal change to the system that results in a different outcome. By providing counterfactual explanations, it 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?" [18]. 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 an 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 optimization problems [12]. 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.

2 Related Work

Our work focuses on explanations in the constraint satisfaction branch of AI working with a multi-point relaxation system. Infeasibility in constraint systems may cause an enormous cost at an industrial level, which includes customer dissatisfaction. Hence, explanation generation has been a very active and interesting topic. The existing work on this topic has mostly focused on identification conflicts in the constraint satisfaction literature and also other relevant areas such as Boolean satisfiability [4, 14]. In this paper, we propose to adapt *counterfactual explanations* to constraint-based systems. Up to date, counterfactual explanations are mostly studied under the Explainable AI (XAI) branch of machine learning systems and attracted a lot of attention.

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 [19, 18]. In a recent survey paper on counterfactuals in XAI, Keane et al. [10] 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 [16]. Miller characterised explanations 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).

Explanation generation in constraint satisfaction is usually achieved by identification of minimal conflicts (or minimal unsatisfiable subsets), or maximal relaxations [7, 13, 17].

Despite long history of explanation generation in constraint satisfaction, counterfactual explanations is 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 [6, 5]. 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 [5]. 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 [1]. The authors define the explanation notions analogous to Ignatiev et al. [5].

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 symbolic AI perspective [2]. 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?".

To the best of our knowledge, the most relevant study to our work has recently been conducted by Korikov et al., in which the authors extend the notion of counterfactual explanations to optimisation-based decisions by using inverse optimisation [12]. They assume that the user is interested in an explanation of why a solution to an optimisation problem does not satisfy a set of additional user constraints that were not initially expressed by the user. In their work, the authors define counterfactual explanations analogous to those of Wachter et al. [18]. 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. The authors also highlight that the links between conflict-detection mechanisms in constraint satisfaction and counterfactual explanations is not clear. Subsequently, Korikov and Beck generalize their work to constraint programming and show that counterfactual explanations can be found using inverse constraint programming using a cost vector [11]. Karimi [9] along with Korikov [12] 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.

In this paper, our goal is to find a counterfactual explanation to a given constraint problem by using conflicts and constraint relaxation, and address the question that Korikov et al. raised related to the connection between conflicts and counterfactuals [12]. To achieve this, we use a relevant work from Ferguson and O'Sullivan as the foundation of our proposed method, in which the authors generalize conflict-based explanations to Quantified CSP framework [3]. Their approach extends the famous QuickXplain algorithm [8] 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.

3 Methodology

First, we define some important notions existing in the Constraint Programming literature on explanations, define counterfactual explanations, and discuss the relation with a counterfactual explanation and constraint relaxation. Consequently, we present our proposed model to find a counterfactual explanation and demonstrate it on a sample item configuration problem.

3.1 Preliminaries

A constraint satisfaction problem (CSP) is defined as a 3-tuple $\phi := (\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} := \{x_1, x_2, ..., x_n\}$ is a finite set of variables, $\mathcal{D} := \{D(x_1), D(x_2), ..., D(x_n)\}$ denotes the set of finite domains where the domain $D(x_i)$ is the finite set of values that variable x_i can take, and a set of constraints $\mathcal{C} := \{c_1, c_2, ..., c_m\}$. More specifically, a problem ϕ in Constraint Programming can be defined using two sets of constraints $\mathcal{B}$ representing the *background constraints* and $\mathcal{F}$ representing the *foreground constraints* (or *user requirements/constraints*) in the context of configuration problems or other interactive settings. Using this alternative representation, a problem is notated as $\phi := (\mathcal{X}, \mathcal{D}, \mathcal{C})$, where $\mathcal{C} := \mathcal{B} \cup \mathcal{F}$. In order to increase readability, we sometimes refer to a problem as $P := (\mathcal{B}, \mathcal{F})$. 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 define below a number of relevant definitions existing in the literature.

▶ **Definition 1** (Minimal Conflict [4]). *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 $\mathcal{B}$ (or if no proper subset of C is a conflict).*

▶ **Definition 2** (Maximal Relaxation [4]). *A subset R of $\mathcal{F}$ is a relaxation of $P := (\mathcal{B}, \mathcal{F})$ iff $\mathcal{B} \cup R$ has a solution. A subset R of $\mathcal{F}$ is a maximal relaxation of a problem and there is no $\{c\} \in \mathcal{F} \setminus R$ such that $\mathcal{B} \cup R \cup \{c\}$ also admits a solution.*

A problem is said to be *over-constrained* if it contains an exponential number of conflicts and an exponential number of relaxations. Based on the definition of a maximal relaxation, the complementary notion of minimal exclusion set can be defined.

▶ **Definition 3** (Minimal Exclusion Set [17]). *Given a problem $P := (\mathcal{B}, \mathcal{F})$ that is inconsistent, and a maximal relaxation $R \subseteq \mathcal{F}$, $E = \mathcal{F} \setminus R$ denotes a minimal exclusion set.*

Note that, the definitions above are defined under *two-point relaxation spaces*, which allow having the constraint in the constraint set, or not. In this paper, we work under *multi-point relaxation spaces*, which correspond to replacing a constraint with any weaker one [3, 15]. To illustrate this, consider the user constraint in Equation 1 between two variables.

$$x_1 \in \{1, 2, 3\}, x_2 \in \{3, 4\}.\{x_1 > x_2\} \tag{1}$$

$$x_1 \in \{1, 2, 3\}, x_2 \in \{3, 4\}.\{x_1 \geq x_2\} \tag{2}$$

Equation 1 is an inconsistent constraint. Assuming that all remaining constraints are consistent, one can remove this constraint from the constraint set to recover consistency in a two-point relaxation space. Alternatively, in a multi-point relaxation space, this constraint can be relaxed to Equation 2, which evaluates to True as there exist satisfying values: $x_1 = 3, x_2 = 3$. We say that Equation 1 is a *tighter* version of Equation 2, and the Equation 2 is a *relaxed* version of the former.

3.2 Finding a counterfactual explanation in CSP

We define a counterfactual explanation by adapting the definitions from Wachter et al. [18] and Korikov et al. [12]. 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. In other words, given a problem $P := (\mathcal{B}, \mathcal{F})$, and a user constraint $\{c\} \notin \mathcal{F}$ and $P' := (\mathcal{B}, \mathcal{F} \cup \{c\})$ is infeasible, we define a *counterfactual explanation* as a set of constraints that explain the minimal set of changes in $\mathcal{F}$ so that the problem P' with the updated constraints becomes feasible. Definition 4 formally defines a counterfactual explanation based on maximal relaxations.

▶ **Definition 4.** *Define two CSPs as* $P := (\mathcal{B}, \mathcal{F} \cup \{c\})$ *that is inconsistent and* $P' := (\mathcal{B} \cup \{c\}, \mathcal{F}')$ *that is consistent, where a constraint* $\{c\} \notin \mathcal{C}$ *denotes a counterfactual user constraint, and* $\mathcal{F}'$ *corresponds to a minimal set of changes applied to* $\mathcal{F}$ *such that* P' *becomes consistent. A counterfactual explanation, denoted by* $\mathcal{E}$*, corresponds to a minimal set of changes required on user constraints to change the state of the problem, where* $\mathcal{E} = \mathcal{F}' \setminus \mathcal{F}$.

Observe that, this system can be generalized to any infeasible problem $P := (\mathcal{B}, \mathcal{F})$ to explain how to recover feasibility without requiring any counterfactual user constraint.

Our method assumes the existence of a multi-point relaxation space defined by the knowledge engineer for each variable in the problem. The relaxation space of a feature may take different characterisations, such as a partially ordered set (poset), lattice, hierarchical ordering, etc. Using these structures pave the way to have comparable or incomparable relaxation states. A top element $\top$ and bottom element $\perp$ must be defined for each relaxation space denoting a maximally relaxed and an infeasible constraint respectively. To illustrate, Figure 1 can be considered as a multi-point relaxation space for equality or inequality constraints that deal with numerical variables. For the sake of notation, we denote comparable states as $\{\top\} \sqsubseteq \{\leq\} \sqsubseteq \{=\} \sqsubseteq \{\perp\}$, where $\{\top\} \sqsubseteq \{\leq\}$ is read as state $\{\top\}$ *dominates* state $\{\leq\}$.

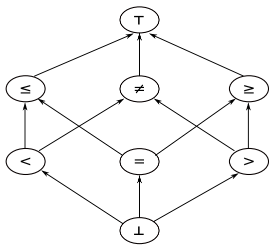

Figure 1 Sample poset of states for numerical constraints in multi-point relaxation space.

Algorithm 1 presents 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. [3]. From the set of different relaxation forms they propose, we only adapt single constraint relaxations in our work. Our proposed method follows an iterative approach for identifying maximal relaxations of 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 [7].

The CounterFactualXplain admits a CSP ϕ and the multi-point relaxation spaces of each constraint that can be relaxed, and returns a counterfactual explanation $\mathcal{E}$ (a set of constraints that needs to be changed to restore feasibility) alongside a relaxed and feasible

Algorithm 1 COUNTERFACTUALXPLAIN $(\phi, \mathcal{R})$

Input: A CSP $\phi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{C} = \mathcal{B} \cup \mathcal{F}$, a set of multi-point relaxation spaces of each user constraint $\mathcal{F} = \{c_1, \ldots, c_n\}$ as $\mathscr{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$.
Output: A counterfactual explanation $\mathcal{E}$, and a maximal relaxation C'.

```
n = |F|, E = ∅, C' = ∅
if φ is feasible then
  return no conflict
end if
if ∀i ∈ {1,...,n}|R_i| = 1 then
  return no relaxation
end if
C' := B ∪ {⊤_i | ⊤_i is top in R_i, ∀i ∈ {1,...,n}}
φ' = ⟨X, D, C'⟩
for c_i ∈ F do
  choose state r_j from maxima of R_i of c_i s.t. r_j ∉ C' and {r_j} ⊆ {c_i}
  while C' ∪ {r_j} is consistent do
    C' = C' ∪ {r_j}
    if r_j equals c_i then
      break
    end if
    r_prev := r_j
    choose maximal r_j from R_i such that {r_j} ∉ C' and {r_prev} ⊑ {r_j}
  end while
  if c_i ≠ r_prev then
    E = E ∪ {r_prev} {r_prev is a part of the explanation}
  end if
end for
return ⟨E, C'⟩
```

version of the constraint set of ϕ. If ϕ is feasible, then the algorithm returns 'no conflict'. Similarly, if there is no relaxation space defined for foreground constraints, the algorithm returns 'no relaxation'. For any other problem, the algorithm creates a copy CSP ϕ' with the original set of variables and domains, but uses a constraint set C' that initially contains only the top elements of each relaxation space for each constraint in $\mathcal{F}$. Then, the procedure iteratively attempts to *tighten* the maximal relaxation of each constraint until either the original user constraint is reached or an inconsistent set of constraints is formed. In this context, *tightening* a constraint c corresponds to adding a dominated state of c to the existing set of constraints. In the case of having incomparable states in the relaxation space, when tightening a constraint, first a path from the top element to the original constraint is found. Next, each path is explored from the most relaxed state to the tighter ones on the path.

4 Demonstration

Consider a small problem from the item configuration domain, in which a user wants to purchase a laptop. Assume each laptop has five features: brand, screen size, memory, battery life, and price. Table 1 lists all available laptops in the solution space. Also assume that the knowledge engineer defines the relaxation spaces as directions for the numerical values (screen size, memory, battery life, and price) for this problem, and the brand relaxation space consists of incomparable states. There are two directions for the numerical values: MIB ("more is better") and LIB ("less is better"), and all brands are equally distant to each other. The users can express their preferences on the direction of numerical features.

Table 1 The set of all available laptops.

Brand	Size (inches)	Memory (MB)	Life (hr)	Price
Lenovo	15.4	1024.0	2.2	1499.99
Sony	11.1	1024.0	11.0	2349.99
Lenovo	15.0	512.0	10.0	2616.99
HP	15.0	512.0	4.5	785.99
Lenovo	14.0	512.0	4.5	1899

For demonstration purposes, assume the user initially expresses her preferred values for some of these features. In Table 2, c_1, c_2, c_3, c_4 correspond to the initial constraints of the user. She is interested in finding a 'Lenovo' laptop with screen size of at least 15 inches, memory of at least 512 MB, and battery life of at least 10 hours. As solution, item {Lenovo, 15.0 inches, 512.0 MB, 10 hr, \$2616.99} is returned to the user. However, the user is not satisfied with the solution as she realises that the recommended item exceeds her budget. Therefore, she adds an extra constraint to the system by asking the question: "Why does the laptop recommended to me costs more than \$2000? I need an alternative that costs at most \$2000.". This user constraint is captured as c_5 in Table 2. Note that, we are interested in a solution that may not satisfy some user constraints but satisfies the counterfactual constraint. Therefore, we move the counterfactual constraint to the background constraints to avoid its relaxation by the CounterFactualXplain algorithm.

Table 2 The list of initial set of user constraints (c_1, c_2, c_3, c_4) and the counterfactual constraint (c_5). The user preferences of directions are MIB ("more is better") and LIB ("less is better").

c_i	Property	User Constraint	Preference
c_1	Brand	Lenovo	–
c_2	Size (inches)	15.0	MIB
c_3	Memory (MB)	512.0	MIB
c_4	Life (hr)	10.0	MIB
c_5	Price	2000	LIB

As our relaxation spaces are defined as directions, we use an ordered list representation. Table 3 presents the relaxation spaces for all constraints, where features are ordered with respect to the user's preference of direction. If the user does not have a preference, we assume the direction is the default direction provided by the knowledge engineer.

Table 3 Relaxation spaces defined for each feature of our data set.

c_i	Relaxation space of c_i $(\mathcal{R}_i)$
c_1	$\top \subseteq$ {HP, Lenovo, Sony} $\subseteq \bot$
c_2	$\top \subseteq 11.1 \subseteq 14.0 \subseteq 15.0 \subseteq 15.4 \subseteq \bot$
c_3	$\top \subseteq 512 \subseteq 1024 \subseteq \bot$
c_4	$\top \subseteq 2.2 \subseteq 4.5 \subseteq 10.0 \subseteq 11.0 \subseteq \bot$
c_5	$\top \subseteq 2616.99 \subseteq 2349.99 \subseteq 1899 \subseteq 1499.99 \subseteq 785.99 \subseteq \bot$

Table 4 lists all the steps performed by our CounterFactualXplain algorithm to find a counterfactual explanation and a maximal relaxation to the given problem with the set of constraints $\mathcal{B}' = \mathcal{B} \cup \{c_5\}$ and $\mathcal{F} = \{c_1, c_2, c_3, c_4\}$. Note that $\mathcal{P}' := \mathcal{B}' \cup \mathcal{F}$ is inconsistent. The algorithm initializes the set of constraints $C' = \{\top_1, \top_2, \top_3, \top_4\}$. Let S_i denote a subset of constraints to represent the elements in C' at each iteration. Initially, $S_0 = C'$, and the subsequent subsets are identified by the iteration number in the table and are accumulated as $S_i = S_{i-1} \cup \{r_j\}$, where r_j denotes the next tightening performed on the constraints.

Table 4 The list of all iterations to find a counterfactual explanation to constraints in Table 2.

i	Subset (S_i)	S_i consistent?	$\mathcal{E}$
1	$S_1 = S_0 \cup \{c_1 = \text{'Lenovo'}\}$	true	$\{\}$
2	$S_2 = S_1 \cup \{c_2 \geq 11.1\}$	true	$\{\}$
3	$S_3 = S_2 \cup \{c_2 \geq 14.0\}$	true	$\{\}$
4	$S_4 = S_3 \cup \{c_2 \geq 15.0\}$	true	$\{\}$
5	$S_5 = S_4 \cup \{c_3 \geq 512\}$	true	$\{\}$
6	$S_6 = S_5 \cup \{c_4 \geq 2.2\}$	true	$\{\}$
7	$S_7 = S_6 \cup \{c_4 \geq 4.5\}$	false	$\{c_4 \geq 2.2\}$

In Table 4, the first iteration tightens c_1 to 'Lenovo', which corresponds to the initial user constraint c_1, and the set of constraints corresponding to this iteration S_1 is consistent. Therefore, in the next iterations (from 2 to 4 inclusive), the constraint tightening is performed for the next constraint c_2. As it is possible to tighten the c_2 until the original user constraint, the fifth iteration, tightens the next constraint, i.e. c_3. Similarly, iterations 6 and 7 performs tightening on c_4, where the seventh iteration with $c_4 \geq 4.5$ makes the set of constraints inconsistent. Therefore, the tightest version of this constraint that is consistent is added to the explanation. Finally, the algorithm returns the maximal relaxation $C' = S_6$, and the counterfactual explanation $\mathcal{E} = \{c_4 \geq 2.2\}$. The user-interface can inform the user with an explanation that is similar to: "If you change your constraint on battery life from 10 hr to 2.2 hr, you can find at least one solution that satisfies your remaining constraints". The relaxed CSP ϕ' contains a single solution: {Lenovo, 15.4 inches, 1024.0 MB, 2.2 hr, $1499.99}.

It is important to note here, one can argue that the item {Lenovo, 14.0 inches, 512 MB, 4.5 hr, $1899} is closer to the initial solution than the solution found by our approach by applying another metric. Our aim in this paper is to find a set of changes that can be applied to the system to change the outcome (feasibility state) of the system. At this stage, we discuss only preliminary research findings, and the relation between system-based minimal changes vs. solution-based minimal changes needs to be studied further.

5 Discussion and Future Work

We propose 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. As future work, we are planning to study the relationship between minimal changes on the set of constraints and its effects on the set of solutions, as well as conduct a user study the utility of our explanations.

6 Acknowledgments

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant 16/RC/3918, 12/RC/2289-P2, and 18/CRT/6223, which are co-funded under the European Regional Development Fund. This research was also partially supported by TAILOR, HumanE AI Network, BRAINE, and StairwAI projects funded by EU Horizon 2020 under Grant Agreements 952215, 952026, 876967, and 101017142.

References

1 Martin C Cooper and João Marques-Silva. On the tractability of explaining decisions of classifiers. In *Proceedings of CP 2021*, 2021.

2 Kristijonas Cyras, Ramamurthy Badrinath, Swarup Kumar Mohalik, Anusha Mujumdar, Alexandros Nikou, Alessandro Previti, Vaishnavi Sundararajan, and Aneta Vulgarakis Feljan. Machine reasoning explainability, 2020. arXiv:2009.00418.

3 Alex Ferguson and Barry O'Sullivan. Quantified constraint satisfaction problems: From relaxations to explanations. In *IJCAI*, pages 74–79, 2007.

4 Sharmi Dev Gupta, Begum Genc, and Barry O'Sullivan. Explanation in constraint satisfaction: A survey. In *Proceedings of IJCAI 2021*, pages 4400–4407, 2021.

5 Alexey Ignatiev, Nina Narodytska, Nicholas Asher, and Joao Marques-Silva. On relating'why?'and'why not?'explanations. *arXiv preprint arXiv:2012.11067*, 2020.

6 Alexey Ignatiev, Filipe Pereira, Nina Narodytska, and João Marques-Silva. A sat-based approach to learn explainable decision sets. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Proceedings of IJCAR 2018*, pages 627–645, 2018.

7 Ulrich Junker. Quickxplain: Conflict detection for arbitrary constraint propagation algorithms. In *IJCAI'01 Workshop on Modelling and Solving problems with Constraints*, 2001.

8 Ulrich Junker. QuickXplain: preferred explanations and relaxations for over-constrained problems. In *Proceedings of AAAI 2004*, pages 167–172, 2004.

9 Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *Proceedings of the International Conference on AI and Statistics*, pages 895–905, 26–28 Aug 2020.

10 Mark T. Keane, Eoin M. Kenny, Eoin Delaney, and Barry 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*, pages 4466–4474, 2021.

11 Anton Korikov and J Christopher 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.

12 Anton Korikov, Alexander Shleyfman, and J. Christopher Beck. Counterfactual explanations for optimization-based decisions in the context of the GDPR. In *Proceedings of IJCAI 2021*, pages 4097–4103, 2021.

13 Mark H. Liffiton and Karem A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reason.*, 40(1):1–33, 2008. doi:10.1007/s10817-007-9084-z.

14 João Marques-Silva and Carlos Mencía. Reasoning about inconsistent formulas. In Christian Bessiere, editor, *Proceedings of IJCAI 2020*, pages 4899–4906, 2020.

15 Deepak Mehta, Barry O'Sullivan, and Luis Quesada. Extending the notion of preferred explanations for quantified constraint satisfaction problems. In *Proc ICTAC*, pages 309–327, 2015.

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

17 Barry O'Sullivan, Alexandre Papadopoulos, Boi Faltings, and Pearl Pu. Representative explanations for over-constrained problems. In *Proceedings of AAAI*, pages 323–328, 2007.

18 S Wachter, BDM Mittelstadt, and C Russell. Counterfactual explanations without opening the black box: automated decisions and the gdpr. *Harvard Journal of Law and Technology*, 31(2):841–887, 2018.

19 Sandra Wachter, Brent D. Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *CoRR*, abs/1711.00399, 2017.