Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)
Survey Track

Explanation in Constraint Satisfaction: A Survey

Sharmi Dev Gupta , **Begum Genc** and **Barry O'Sullivan**
School of Computer Science & Information Technology
University College Cork, Ireland
{sharmi.devgupta|begum.genc|b.osullivan}@cs.ucc.ie

Abstract

Much of the focus on explanation in the field of artificial intelligence has been on machine learning methods and, in particular, concepts produced by advanced methods such as neural networks and deep learning. However, there has been a long history of explanation generation in the general field of constraint satisfaction, one of the AI's most ubiquitous subfields. In this paper we survey the major seminal papers on the explanation and constraints, as well as some more recent works. The survey sets out to unify many disparate lines of work in areas such as model-based diagnosis, constraint programming, Boolean satisfiability, truth maintenance systems, quantified logics, and related areas.

1 Introduction

While much of the recent focus on explanation in AI has been on machine learning, and on explanations for the techniques based on neural networks and deep learning methods, there is a considerable and long-standing literature related to explanations in the context of constraints. The objective of this survey is to briefly review the major directions of research on explanation for constraint satisfaction and bring together the many different, but linked, themes from related fields.

Most current approaches to explanation generation in constraint-based settings are based on the notion of a (set-wise) minimal set of unsatisfiable constraints, also known as a minimal conflict set of constraints. However, a minimal conflict does not necessarily give an intuitive explanation, in that many users will want to be shown which subsets of their constraints they can satisfy and which they cannot satisfy. It has also been shown that minimal conflict-based explanations can also be spurious and misleading [Friedrich, 2004].

Option	Cost
Roof rack	500
Convertible	500
CD Player	500
Leather Seats	2600

c_1	Total cost ≤ 3000
c_2	Roof rack
c_3	Convertible
c_4	CD Player
c_5	Leather Seats

Figure 1: An example explanation problem: assume there is one constraint that "*convertible cars cannot have roof racks*".

We briefly exemplify some aspects of explanations in constraints through an example in Figure 1 based on a well-known car configuration problem [Junker, 2004]. In this example there is one technical constraint that states that *convertible cars cannot have roof racks*. On the right-hand side of the figure there are the choices made by the user: the user has a maximum budget of 3000 (c_1), wants a roof rack (c_2), etc. On the left-hand side we see the costs associated with each option that the user might select.

Clearly the set of user's choices $\{c_1, \ldots, c_5\}$ cannot be satisfied simultaneously. In fact, there are subsets of the user's choices that also cannot be satisfied: $\{c_2, c_3\}, \{c_1, c_2, c_5\}, \{c_1, c_3, c_5\}$, and $\{c_1, c_4, c_5\}$. Note that any subset, if any, of these subsets of constraints is consistent; these are often referred to as *minimal conflicts*. These conflicts are somewhat useful as explanations, but they have challenges. Consider, for example, what happens if we present $\{c_1, c_2, c_5\}$ as an explanation for why the set of constraints $\{c_1, \ldots, c_5\}$ is not satisfiable. The user would be mistaken in thinking that simply eliminating this conflict by removing, say, constraint c_2 is enough to recover consistency. It is not, because $\{c_1, c_3, c_5\}$ or is also a conflict. Similarly, relaxing c_5 from $\{c_1, c_2, c_5\}$ would not have been enough because $\{c_2, c_3\}$ is a conflict. Minimal conflicts only explain why a set of constraints is inconsistent. In order to recover consistency all minimal conflicts must be eliminated by relaxing a set of constraints that form a hitting set of the conflicts.

Rather than considering conflicts, one can focus on subsets of constraints that are consistent. In this example, the sets of user constraints $\{c_1, c_2\}, \{c_1, c_5\}$ and $\{c_1, c_2, c_4\}$ are consistent, and a solution can be found for them. These sets of constraints are often referred to as *relaxations* of the problem. Sets $\{c_1, c_5\}$ and $\{c_1, c_2, c_4\}$ are interesting, since adding any additional user constraint to them will make them unsatisfiable. Such relaxations are often referred to as *maximal relaxations*. None of the conflicts are contained in the relaxation.

In the remainder of this paper, we present our review of explanation for constraint satisfaction. We can characterise explanation methods in constraint satisfaction along four dimensions. First, explanation methods can be seen as either *solver-specific* or *solver-agnostic*. Solver-specific methods are typically those that are integrated into the solver itself. For example, these methods might either be part of a mechanism to record how constraint propagation removes particular

values from the domains of variables, or they might compute reasons for a dead-end in search that can be added to the set of the constraints of the problem to avoid a future failure for the same reasons. Solver agnostic methods are typically algorithms that perform a-posteriori analysis to generate an explanation. Second, methods can be considered as being *user-focused*, for use in an interactive setting or where users add constraints to express preferences, or for use by a constraint solver to *improve search*. Third, there are a variety of *measures of quality* one can consider, e.g. finding an explanation in terms of the most important constraints, finding a small explanation such that it is of minimum cardinality, irreducible, traceable or in some other way compact. Finally, one might be interested in computing a single explanation, all possible explanations, or a selection of possible explanations. In our review we will highlight the criticals elements of these dimensions in the settings we consider.

2 Formal Background

A constraint satisfaction problem is defined as a triple $P := \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$. The set of $\mathcal{X}$ denotes a finite set of variables, $\mathcal{D}$ is a finite set of domains of each variable, where domain of a variable $x_i \in \mathcal{X}$ denoted by $D(x_i)$ is defined as a set of finite values that x_i can take, and $\mathcal{C}$ is a finite set of constraints.

2.1 Foreground versus Background Constraints

In some contexts, e.g. interactive search, product configuration, etc., it is useful to consider a partition of constraint types. A problem $\mathcal{P}$ can be represented by two sets of constraints $\mathcal{P} := (\mathcal{B}, \mathcal{F})$, where $\mathcal{B}$ represents the *background constraints* (or hard constraints), i.e. the constraints that cannot be relaxed, and $\mathcal{F}$ the *foreground constraints* (or soft constraints). The set $\mathcal{F}$ can also be referred as customer requirements in the context of configuration problems, or user constraints in other interactive settings. A set of constraints is *inconsistent* (or *unsatisfiable*) if there is no solution; otherwise it is *consistent* (or *satisfiable*). A set of constraints C is referred as *over-constrained* if they cannot be satisfied at the same time.

We present the basic definitions related to explanations in CSPs before presenting some related definitions taken from the various papers we reviewed.

Definition 1 (Relaxation [Junker, 2004])**.** *A subset R of $\mathcal{F}$ is a relaxation of $P := (\mathcal{B}, \mathcal{F})$ iff $\mathcal{B} \cup R$ has a solution.*

A natural and commonly used notion of quality for relaxations is that they are maximal with respect to set inclusion.

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

The complement of a maximal relaxation, those constraints that are excluded to recover consistency is referred to as a *minimal exclusion set* [O'Sullivan *et al.*, 2007]. Capturing how inconsistency arises, we have the notion of a conflict set of constraints, and its natural notion of quality that is set-wise minimal.

Definition 3 (Conflict [Junker, 2004])**.** *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.*

Definition 4 (Minimal Conflict [Junker, 2004])**.** *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).*

Each of the maximal relaxations (minimal conflicts) are set-wise incomparable. The number of maximal relaxations (minimal conflicts) is exponential in the cardinality of $\mathcal{F}$.

There exists a duality between minimal conflicts and maximal relaxations. The minimal exclusion sets, the complements of each maximal relaxation, form a hitting set of the minimal conflicts in the problem since by excluding these constraints, all conflicts are removed. This is a property that can be algorithmically exploited when enumerating all possible minimal conflicts [Bailey and Stuckey, 2005; Reiter, 1987].

2.2 Monolithic Formulas and Sets of Constraints

In some applications no distinction is made between a background and foreground sets of constraints. Instead the set of constraints (or the formula in the case of Boolean satisfiability) is regarded as a single conjunction of individual constraints (clauses). In the associated literature the terminology is similar to that presented above.

Definition 5 (Unsatisfiable Core [Lynce and Marques-Silva, 2004])**.** *Given a formula ϕ, UC is an unsatisfiable core for ϕ iff UC is a formula ϕ_c s.t. ϕ_c is unsatisfiable and $\phi_c \subseteq \phi$.*

Definition 6 (Minimal Unsatisfiable Core (MUC) [Lynce and Marques-Silva, 2004])**.** *A UC for a formula ϕ is a minimal unsatisfiable core iff removing any clause $\omega \in UC$ from UC implies that $UC \setminus \{\omega\}$ is not an unsatisfiable core.*

When the core is restricted to clauses of the original formula, we have the following related notion, which is equivalent to minimal conflict in the constraint case.

Definition 7 (Minimal Unsatisfiable Subset (MUS) [Liffiton and Sakallah, 2008])**.** *A subset $U \subseteq \mathcal{F}$ is an MUS if U is unsatisfiable and $\forall C \in U, U \setminus \{C\}$ is satisfiable.*

Definition 8 (Maximal Satisfiable Subset (MSS) [Liffiton and Sakallah, 2008])**.** *A subset $S \subseteq \mathcal{F}$ is an MSS if S is satisfiable and $\forall C \in (\mathcal{F} \setminus S), S \cup \{C\}$ is unsatisfiable.*

Note that, a MUS is also referred as *minimal unsatisfiable subformula*. The analogous notion to *minimal exclusion set* discussed above is the *minimal correction subset*.

Definition 9 (Minimal Correction Subset (MCS) [Liffiton and Sakallah, 2008])**.** *A subset $M \subseteq \mathcal{F}$ is an MCS if $\mathcal{F} \setminus M$ is satisfiable and $\forall C \in M, \mathcal{F} \setminus (M \setminus \{C\})$ is unsatisfiable.*

In the field of model-based diagnosis one can find similar notions to conflicts, such as *diagnosis* and *minimal diagnosis* [Felfernig *et al.*, 2012]. In this context, the concept of minimal diagnosis is analogous to that of an MCS. Finding an MCS corresponds to finding a minimal hitting set of all minimal conflict sets of the problem.

The literature in this area is vast and is not possible to include all papers in the short survey we present here. We refer readers to an excellent recent survey that provides an in-depth analysis of inconsistent formulas, including MUS and MCS computation in non-monotonic formulas [Marques-Silva and Mencía, 2020].

3 Solver-Agnostic Approaches

In this section, we review solver-agnostic, sometimes referred to as non-intrusive, approaches to explanation generation for CSPs that can easily integrate with any arbitrary solver, i.e. the system does not make any assumptions about the solver.

3.1 QUICKXPLAIN

Truth maintenance systems are able to compute explanations for those problem solvers that create explicit justifications for each inference step. As this requirement limits the scope of applicability of truth maintenance systems, there was an interest in solver-agnostic methods that could work for arbitrary constraint solvers. Bakker et al. [1993] proposed a sequential method for computing a minimal conflict, but it required a linear number of consistency checks and was too expensive for many problems.

It was long believed that the conflict detection problem could not be solved recursively by successively decomposing the constraint set into smaller subsets. Indeed, it is not possible to discard the other subsets when trying to find elements of a conflict within a selected subset as this conflict might involve elements from several of the subsets. Instead of discarding these other subsets Junker [2001] proposed to move them to the background when looking for conflict elements in a selected subset. This permitted a new recursive conflict detection strategy that can scale well and is able to handle complex real-world problems. However, the worst-case query performance of the proposed algorithm is not better than Bakker's algorithm [Marques-Silva and Mencía, 2020].

In subsequent work, Junker [2004] showed that this divide-and-conquer strategy is able to find a lexicographically preferred and thus minimal (and irreducible) conflict and introduced a consolidated version of the QUICKXPLAIN algorithm. This work influenced many other researchers and won the AAAI Classic Paper award in 2020.

Whereas QUICKXPLAIN focused on the computation of conflicts, it formulates the decomposition property for both conflicts and relaxations. Consider two constraints sets $\mathcal{F}_1$ and $\mathcal{F}_2$ such that all elements of the first set are preferred to those of the second set with respect to a total importance order defined on all constraints:

1. If R_1 is a lexicographically-preferred relaxation of $(\mathcal{B}, \mathcal{F}_1)$ and R_2 is a lexicographically-preferred relaxation of $(\mathcal{B} \cup R_1, \mathcal{F}_2)$, then $R_1 \cup R_2$ is a lexicographically-preferred relaxation of $(\mathcal{B}, \mathcal{F}_1 \cup \mathcal{F}_2)$.
2. If C_2 is a lexicographically-preferred conflict of $(\mathcal{B} \cup \mathcal{F}_1, \mathcal{F}_2)$ and C_1 is a lexicographically-preferred conflict of $(\mathcal{B} \cup C_2, \mathcal{F}_1)$, then $C_1 \cup C_2$ is a lexicographically-preferred conflict of $(\mathcal{B}, \mathcal{F}_1 \cup \mathcal{F}_2)$.

Many related perspectives have appeared. For example, around the same time in model-based diagnosis, Mauss and Tatar [2002] proposed a similar conflict detection algorithm without taking into account preferences. Their approach recursively divides the constraint set according to the structure of an inconsistency proof and therefore cannot guarantee that lex-preferred conflicts are computed. In Boolean satisfiability, Lynce and Marques-Silva [2004] focused on finding minimum unsatisfiable cores to certify the solver using unsatisfiability proofs. They focused on finding the unsatisfiable cores with a focus on smallest (in terms of the number of clauses) unsatisfiable cores of a given instance. All unsatisfiable formulas have at least one minimum unsatisfiable core, and the authors highlight that there may be a significant difference between the length of a minimal unsatisfiable core and a minimum one. However, their technique is not scalable enough due to the scale of the search space. Subsequently, Ignatiev et al. [2015] proposed a more effective approach, namely FORQES, to tackle the scalability issue. They also noted that finding the smallest unsatisfiable core is hard for the second level of the polynomial hierarchy.

Hemery et al. [2006] studied how to extract minimal conflict sets from constraint networks as a refinement of the work of Bakker et al. [1993]. Their proposed conflict-based approach is based on performing complete runs of backtracking search by using a conflict-directed variable ordering heuristic. They iteratively restart the search by keeping the constraint weightings until no smaller core can be found. They show that by using dichotomic search to identify an unsatisfiable set, their approach is bounded by $O(log_e \times k_e)$, where e is the number of constraints and k_e is the number of constraints of the extracted MUC. When the extracted MUC is small, it performs better than QUICKXPLAIN in the worst case.

There have been many directions of work inspired by the QUICKXPLAIN algorithm. O'Callaghan et al. [2005] introduced and studied corrective explanations. The goal of providing a corrective explanation is to identify reassignments to a subset of user-generated unary constraints in the case of an inconsistency or value restoration. Consequently, Li et al. [2013] improved the performance of this algorithm by requiring fewer consistency checks.

Felfernig et al. [2012] presented an algorithm called FASTDIAG, which is a QUICKXPLAIN-inspired strategy that identifies one minimal conflict at a time, motivated on configuration problems. Diagnosis approaches usually require the calculation of at least n minimal conflict sets to find a minimal diagnosis of cardinality n. The FASTDIAG algorithm determines the preferred diagnosis without calculating the corresponding conflict sets. Laborie [2014] proposed a method, namely ADEL, to find a MUS, and they showed that it requires fewer consistency checks than QUICKXPLAIN. Subsequently, Marques-Silva and Previti [2014] revisited definitions of MUS, MCS, and MSS under preferences, and analysed different algorithms and pruning techniques.

Ferguson and O'Sullivan [2007] developed an explanation framework for quantified constraint satisfaction problems (QCSP). While in the classic CSP variables are implicitly existentially quantified, in a QCSP some variables can also be universally quantified. For universally quantified variables the constraint set must be satisfied for every possible assignment to these variables. The space of possible relaxations and conflicts in a QCSP is much more complex: relaxations correspond to restricting the domains of universally quantified variables, expanding the domains of existentially quantified variables, moving universal quantifiers leftwards in the quantifier sequences, or replacing universal quantifiers with existential ones. Mehta et al. [2015] extended the notion of preferred explanations into the Quantified CSP framework.

3.2 Model-based Diagnosis

One of the earliest studies on constraint-based explanation was conducted by Raymond Reiter in the 1980s in the context of *model-based diagnosis* using hitting-set trees (HS-tree) [Reiter, 1987]. In this approach, the nodes of an HS-tree are labeled with conflicts, and the edges are labelled with the elements of conflicts. Given a set of inputs and a set of expected outputs, a diagnosis is expected to help with analyzing the observed output and understanding why a system is not working as expected. Reiter showed that finding all diagnoses corresponds to finding all minimal hitting-sets of conflicts, and provided a breadth-first search procedure over HS-trees with different tree-pruning rules to compute all diagnoses. An important remark is that the diagnosis running time depends heavily on the cost of consistency checking at each node.

Han and Lee [1999] proposed a method for deriving all minimal conflict sets of a given conflict set using a different tree structure, called a CS-tree, for the diagnosis of circuits using satisfiability. The nodes of a CS-tree are labeled with subsets of the given conflict set, where no two nodes share the same label. The technique is based on enumerating all subsets in the tree and not to visit successors of a node if the satisfiability test at the node is true. Their proposed approach is independent of the order of node generation in CS-tree, and the successor check provides good pruning to the search.

Stumptner and Wotawa [2001] further studied an alternative to Reiter's procedure for computing diagnoses. Their method uses a tree-structured system, whose components can be expressed using either mathematical functions or constraints. They speed-up the reasoning for finding all minimal diagnoses by exploiting the tree using a top-down approach using forward and backward propagation, and limiting the maximal cardinality of diagnoses. More recently, Jannach et al. [2015] improved Reiter's approach in terms of speeding up the computation by parallelizing construction of the tree.

Marques-Silva et al. [2013a] used MAXSAT for finding an MCS, and highlighted that although MCS calculation seems to be promising, existing algorithms do not scale well to the very hard instances of MAXSAT. Additionally, Marques-Silva et al. [2013b] presented progression-based algorithms for MUS extraction. Metodi et al. [2012; 2014] studied this impracticality of model-based diagnosis systems in the presence of many faults. They proposed an efficient SAT-based approach to finding either single or all minimal cardinality diagnoses that performed well on standard benchmarks. Marques-Silva et al. [2015] built on this work and proposed a novel encoding into MAXSAT. Several recent algorithms have been proposed for efficient enumeration of MCSes [Grégoire *et al.*, 2018; Previti *et al.*, 2018].

3.3 Finding All Minimal Conflicts in CSPs

García de la Banda et al. [2003], motivated by the problems of relying on a single conflict as discussed in the introduction that some explanations may be easier to understand than others, proposed a method for finding all minimal unsatisfiable subsets of a set of constraints to extract the simplest explanation. They used pre-processing, reasoning about independence and redundant constraints, use incrementality to speed up the derivation towards finding satisfiable subsets, and reduce the search space to be explored. Subsequently, Bailey and Stuckey [2005] proposed a hitting set-based approach to compute all minimal unsatisfiable constraints sets based on earlier techniques [Gunopulos *et al.*, 2003], which is used for discovering interesting maximal frequent patterns in data mining. They adapted it to enumerate both minimal unsatisfiable and maximal satisfiable sets of constraints, exploiting the duality between those. The authors enhanced the procedure by adding information from the constraint graph such as visiting the sets in increasing order of their cardinalities. The results were compared with the technique from García de la Banda et al. [2003] and showed significant improvements in running time. Bailey and Stuckey's approach is quite similar to Reiter's approach, with the main difference being that Reiter uses hitting set calculations to find a new minimum unsatisfiable set at each iteration, whereas Bailey and Stuckey finds a new maximum satisfiable subset.

In interactive settings the response time is also as important as the goodness of an explanation. Some researchers have compiled a CSP into a suitable data structure, such as a binary decision diagram or automaton, and then exploited it at run-time. As an example, Amilhastre et al. [2002] proposed to compile the solutions of a given configuration problem into an automaton. Then, they proposed tractable algorithms on the automaton that maintains global consistency, immediately detecting inconsistencies, finding nogoods to offer maximal relaxations, and offer minimal explanations if a value is removed from the domain of an important variable. Papadopoulos and O'Sullivan [2009] incorporated user preferences into explanations by making use of prime implicates to suggest the best relaxation to the unsatisfiable user configuration. Their proposed approach is based on computing all the conflicts in advance and compiling these into a compact representation to be able to find quick explanations online.

3.4 Finding All Unsatisfiable Subformulas in SAT

Liffiton and Sakallah [2005] proposed CAMUS that extracts all minimal unsatisfiable subformulas following a similar concept to [Bailey and Stuckey, 2005], and showed that their method outperforms their implementation of Bailey and Stuckey's approach for CNF. The CAMUS algorithm uses hitting set dualization to compute all the MUSes in an unsatisfiable constraint system. This approach is also based on the duality between MUS and MCS [Liffiton and Sakallah, 2008]. First, they find all MCSes of a constraint system, which they show is easier than finding MUSes. Then, they compute all irreducible hitting sets of MCSes, which correspond to all MUSes of the constraint system. Their approach uses the fact that MCS and MSS are complementing each other. First they find an MSS by solving MAXSAT, then identify the corresponding MCS. Considering the complexity of finding all MUSes, they propose some further algorithms that relax the completeness of the algorithm. In the meanwhile, Gregoíre et al. [2007] presented an improvement to CAMUS. Their main improvement was to use local search to identify potential maximal satisfiable subsets before extracting the corresponding set of minimal unsatisfiable subformula. The approach can be considered as a combination

of the Liffiton and Sakallah's complete approach with local search, and not necessarily complete. They also incorporated critical clauses to their search, and show that the proposed approach offers significant improvements.

Liffiton and Malik [2013] noted that most of the algorithms proposed to enumerate interesting MUS for a constraint set perform worse in terms of time efficiency when compared to algorithms focused on finding a single MUS, more specifically, producing the first output as quickly as the single MUS algorithms and finding consecutive MUS after the similar time interval. Their paper proposed a novel algorithm, namely MARCO (Mapping Regions of Constraint sets), that attempts to address these gaps and it succeeds in quickly enumerating all the MUSes for any given constraint system. One can use it with any algorithm that detects a single MUS. The empirical analysis shows that MARCO fares better than the other enumeration algorithms, especially for applications which do not require all the MUSes in the constraint set. CAMUS is faster to enumerate all sets of MUSes than MARCO but the early results of MARCO are preferred for applications that require partial MUS enumeration within a time limit. Subsequently, Liffiton et al. [2016] proposed an optimised version of their algorithm.

Considering the intractability of enumerating all MUSes, Bendík and Cerná [2020a] proposed domain agnostic algorithms that enumerate MUSes online. They examined the MARCO algorithm and concluded that the efficiency of it varies a lot with the constraint domain. Hence, they introduced a domain agnostic tool, which implements many MUS enumeration algorithms and provides support for three constraint domains namely SAT, SMT, and LTL. Some recent better performing approaches in MUS enumeration include Narodytska et al. [2018] and the UNIMUS algorithm proposed by Bendík and Cerná [2020b].

3.5 Finding Many Explanations

While presenting a single conflict or relaxation is not particularly insightful and enumerating them is impractical, it can be useful to be presented with an informative set of explanations. O'Sullivan et al. [2007] proposed an approach to presenting a subset of all relaxations and exclusion sets of the user's constraints along with a set of representative explanations. Their notion of *representative set* corresponds to having a relaxation that contains each constraint that appears in a relaxation and also in an exclusion set. They show that computing representative explanations is NP-hard considering polynomial-time consistency checkers. This algorithm RESPRESENTATIVEXPLAIN is based on [Bailey and Stuckey, 2005]'s method for computing all minimal conflict sets. The main difference is to reduce the set of minimal exclusion sets to produce minimal representative set of explanations. This minimisation is done by removing the explanations that are not particularly representative. The representative set is linear in the number of user constraints in the worst case.

Building upon the work of Reiter and Junker, Shchekotykhin et al. [2015] combined hitting set trees and QUICKXPLAIN principles and propose a complete approach called MERGEXPLAIN, which is a divide-and-conquer procedure that allows computing several minimal conflicts.

4 Solver-Specific Approaches

4.1 Truth Maintenance Systems

A problem solver such as a constraint solver may explore different decisions and the logical consequences of these decisions. A truth maintenance system (TMS) records all the decisions and inferences made by the problem solver and keeps justifications for the recorded inferences [Doyle, 1981]. Based on this information, a TMS is able to provide explanations for the recorded inferences and to identify the decisions that have led to these inferences. A TMS may also maintain a belief state, i.e. a subset of the decisions and recorded inferences. If a decision is retracted, the TMS will retract all inferences that are no longer supported by the remaining decisions according to the recorded justifications. Similarly, if a decision is re-added, the TMS will re-add all inferences that are again supported by the re-added decision. When the problem solver encounters a contradiction, the TMS will record this inference as well and trigger a particular process, called dependency-directed backtracking, that retracts decisions until the contradiction is no longer derived from the remaining decisions and the recorded justifications. Dependency-directed backtracking generates a no-good, i.e. an explanation for the contradiction in terms of the current decisions. Doyle's truth maintenance system retracts one of the decisions in the nogood and adds a (non-monotonic) justification for this retraction that consists of the other decisions in this nogood. If some of these other decisions are retracted, the now retracted decision may be re-added, which leads to a non-monotonic behavior. Unfortunately, this additional revision capability may be problematic in presence of cycles.

An assumption-based truth maintenance system (ATMS) avoids those problems by maintaining multiple belief states simultaneously [De Kleer, 1986]. An ATMS computes multiple explanations for each recorded inference. The ATMS keeps the explanations for the contradiction node in a particular nogood database and can thus check whether a given set of decisions is inconsistent (i.e. is a superset of some of the nogoods). Moreover, the ATMS keeps the computed explanations for each recorded inference (except for the contradiction) as a label of this recorded inference and can thus check whether a given set of decisions supports (or implies) this inference (i.e. is a superset of some of the explanations in the label). Nevertheless, these capabilities may be costly in space and time as the nogood database, as well as the labels, may be exponential in size. The ATMS tries to mitigate this by keeping only minimal nogoods and minimal explanations, but in general a focusing mechanism is necessary to make an ATMS work in practice.

4.2 Explanations for Improving Search

Several algorithms exist to deal with over-constrained problems using an intelligent backtracking strategy. For instance, Ginsberg [1993] presented a method that moves backtrack points deeper into the search space to avoid losing progress made on solving a search problem when backtrack method is called. Ginberg's technique, called dynamic backtracking, is a variant of dependency-directed backtracking [Stallman and Sussman, 1977]. The original approach by Stallman

and Sussman includes directly backtracking to the source of the problem. Some of the key elements here are storing the previous values of the variables when applying backtracking, i.e. reordering the variables already assigned values thus far. The other key idea is to erase the value given to a variable instead of backtracking to it. Prosser [1995] modified the MAC algorithm from Sabin and Freuder [1994] for maintaining arc consistency, using conflict-directed backjumping, to avoid repeating the redundant search. In the context of satisfiability, one of the most effective explanation approaches is *clause learning* which is now a standard component of SAT solvers [Marques-Silva *et al.*, 2009; Biere *et al.*, 2009].

Other work in this area include the explanation-based solver, PaLM [Jussien and Barichard, 2000], and related approaches such as the work by Cambazard et al. [2004] which use Benders decomposition techniques to generate nogoods and eventually solve a hard real time allocation problem.

4.3 Explanations and Global Constraints

Providing explanations are straightforward when working with basic constraints, however as Jussien [2000] identified, it is not that apparent for global constraints due to their complex nature. The general approach is to express the global constraints in terms of basic ones first. The decomposed versions of the global constraints often introduce extra variables, but decomposition is mostly necessary for producing explanations. As an example, Schutt et al. [2011] explained the CUMULATIVE global propagator by decomposing it into basic constraints. They use lazy clause generation, which is a hybrid approach that combines finite domain propagation and SAT solving. First, a SAT model of the corresponding problem is generated, then the explanations concluded by the SAT solver are used to form an explanation.

Similarly, Downing et al. [2012] observed and examined various propagation mechanisms for ALLDIFFERENT constraint, provided explanations for their propagation, and experimented with an extended version in order to include explainability. They also studied the effects of clause learning on propagation tradeoffs. They observed that it is desirable to have different propagators or decompositions of the global constraint for explanation as some methods work better for some specific problems. They concluded that learning nogoods is beneficial in general, but is not effective on some models. More recently, ongoing study of Gontier et al. highlightd the difficulty of explaining global constraints in off-the-shelf solvers [Gontier *et al.*, 2020]. They propose an approach based on conflict-driven clause learning that can generate explanations for any constraint decomposition, and demonstrate in on the CUMULATIVE constraint.

5 Explanations and Puzzles

Puzzle solving has long been used as a domain for demonstrating the value of explanation techniques for constraint programming. In these settings the puzzle is encoded as a constraint satisfaction problem [Little *et al.*, 2002]. When solving puzzles and providing explanations to the user, the main focus is on providing meaningful and short explanations that are similar to human reasoning so that the user can understand the rationale behind it. For example, Sqalli and Freuder [1996] used inference-based constraint satisfaction to support explanations of problem solving with a case study on logic puzzles. Their motivation is not only to provide a solution to the puzzle, but solve the puzzle in a way that the player would find the thought process natural to follow so that explainable. Subsequently, Freuder et al. [2001] studied the problem of how the implications of user choices can be explained in interactive settings such as when solving the Sudoku problem. They aim to answer why a user is presented a particular solution by using dependency records, why a specific value was assigned to a variable, or why the choices led to a failure. They use inference methods to determine if selecting a value leads to a solution or failure.

Escamocher and O'Sullivan [2019] proposed an algorithm that mimics human-type reasoning to solve logic grid puzzles. The expectation is that, by following a similar reasoning with the human, the explanations of the system will be more clear to the user. Their system is capable of producing a solution for the user, or explaining at each step why a cell is assigned a specific value. Somewhat related, Bogaerts et al. [2020] examined the inference steps taken during propagation in a non-interactive setting to solve logic grid puzzles. The greedy procedure iteratively builds explanation sequences, and a cost function is used evaluate the explanation sequences at each step to select the lowest scoring next explanation.

6 Conclusions and Future Directions

We have brought together the many strands of research related to explanation generation in constraint satisfaction, including the related area of Boolean satisfiability. The core ideas in this field go back a number of decades, and there are many parallel lines of work that have significant commonalities. We have attempted to draw these out through our categorisation of the literature. It is our hope that other subfields of AI will draw upon the work in constraint satisfaction to find new ways of applying these results and methods, but also find inspiration to apply it in their own fields.

Current CP systems are focusing on performance and whether a solution to the constraint system can be found in a reasonable time or report none exists. A reported solution can be verified using certifying solvers. However, it is often the case that the user wants to understand the reported solution, or needs to interact with the system to take steps that lead to a solution. This is the focus of an exciting line of new work on *proof logging* [Gocht *et al.*, 2020]. It is also interesting to consider how CSPs can be reformulated to support better explanation [Cambazard and O'Sullivan, 2008]. As we deal with larger and more complex models, we expect to see greater focus on directions like these.

Acknowledgements

The authors are extremely grateful for the feedback and comments provided by Ulrich Junker and anonymous reviewers. This publication has emanated from research conducted with 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.

References

[Amilhastre *et al.*, 2002] Jérôme Amilhastre, Hélène Fargier, and Pierre Marquis. Consistency restoration and explanations in dynamic CSPs application to configuration. *Artif. Intell.*, 135(1-2):199–234, 2002.

[Bailey and Stuckey, 2005] James Bailey and Peter J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *Practical Aspects of Declarative Languages*, pages 174–186, 2005.

[Bakker *et al.*, 1993] R. R. Bakker, F. Dikker, F. Tempelman, and P. M. Wognum. Diagnosing and solving over-determined constraint satisfaction problems. In *Proceedings of IJCAI-93*, pages 276–281, 1993.

[Bendík and Cerná, 2020a] Jaroslav Bendík and Ivana Cerná. MUST: minimal unsatisfiable subsets enumeration tool. In *Proceedings TACAS 2020*, pages 135–152, 2020.

[Bendík and Cerná, 2020b] Jaroslav Bendík and Ivana Cerná. Replication-guided enumeration of minimal unsatisfiable subsets. In *Proc. of CP*, pages 37–54, 2020.

[Biere *et al.*, 2009] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*. IOS Press, 2009.

[Bogaerts *et al.*, 2020] Bart Bogaerts, Emilio Gamba, Jens Claes, and Tias Guns. Step-wise explanations of constraint satisfaction problems. In *Proceedings of ECAI 2020*, volume 325, pages 640–647, 2020.

[Cambazard and O'Sullivan, 2008] Hadrien Cambazard and Barry O'Sullivan. Reformulating table constraints using functional dependencies - an application to explanation generation. *Constraints An Int. J.*, 13(3):385–406, 2008.

[Cambazard *et al.*, 2004] Hadrien Cambazard, Pierre-Emmanuel Hladik, Anne-Marie Déplanche, Narendra Jussien, and Yvon Trinquet. Decomposition and learning for a hard real time task allocation problem. In *Proceedings of CP 2004*, pages 153–167, 2004.

[De Kleer, 1986] Johan De Kleer. An assumption-based tms. *Artif. Intell.*, 28(2):127–162, 1986.

[de la Banda *et al.*, 2003] Maria J. García de la Banda, Peter J. Stuckey, and Jeremy Wazny. Finding all minimal unsatisfiable subsets. In *Proceedings of ACM SIGPLAN 2003*, pages 32–43, 2003.

[Downing *et al.*, 2012] Nicholas Downing, Thibaut Feydy, and Peter J. Stuckey. Explaining alldifferent. In *Proceedings of ACSC 2012*, ACSC '12, page 115–124, AUS, 2012.

[Doyle, 1981] Jon Doyle. A truth maintenance system. In *Readings in Artificial Intelligence*, pages 496 – 516. 1981.

[Escamocher and O'Sullivan, 2019] Guillaume Escamocher and Barry O'Sullivan. Solving logic grid puzzles with an algorithm that imitates human behavior. *CoRR*, abs/1910.06636, 2019.

[Felfernig *et al.*, 2012] Alexander Felfernig, Monika Schubert, and Christoph Zehentner. An efficient diagnosis algorithm for inconsistent constraint sets. *Artif. Intell. Eng. Des. Anal. Manuf.*, 26(1):53–62, 2012.

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

[Freuder *et al.*, 2001] Eugene C. Freuder, Chavalit Likitvivatanavong, and Richard J. Wallace. Deriving explanations and implications for constraint satisfaction problems. In *Proceedings of CP 2001*, pages 585–589, 2001.

[Friedrich, 2004] Gerhard Friedrich. Elimination of spurious explanations. In *Procs of ECAI*, pages 813–817, 2004.

[Ginsberg, 1993] Matthew L. Ginsberg. Dynamic backtracking. *J. Artif. Intell. Res.*, 1:25–46, 1993.

[Gocht *et al.*, 2020] Stephan Gocht, Ross McBride, Ciaran McCreesh, Jakob Nordström, Patrick Prosser, and James Trimble. Certifying solvers for clique and maximum common (connected) subgraph problems. In *Proceedings of CP 2020*, volume 12333 of *LNCS*, pages 338–357, 2020.

[Gontier *et al.*, 2020] Arthur Gontier, Charlotte Truchet, and Charles Prud'homme. Conflict analysis in cp solving: Explanation generation from constraint decomposition. In *CPTAI-20, A CP 2020 Workshop*, 2020.

[Grégoire *et al.*, 2007] Éric Grégoire, Bertrand Mazure, and Cédric Piette. Boosting a complete technique to find MSS and MUS thanks to a local search oracle. In *Proceedings of IJCAI 2007*, pages 2300–2305, 2007.

[Grégoire *et al.*, 2018] Éric Grégoire, Yacine Izza, and Jean-Marie Lagniez. Boosting mcses enumeration. In *Proceedings of IJCAI 2018*, pages 1309–1315. ijcai.org, 2018.

[Gunopulos *et al.*, 2003] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharm. Discovering all most specific sentences. *ACM Trans. Datab.Syst.*, 28(2):140–174, 2003.

[Han and Lee, 1999] Benjamin Han and Shie-Jue Lee. Deriving minimal conflict sets by cs-trees with mark set in diagnosis from first principles. *IEEE Trans. Syst. Man Cybern. Part B*, 29(2):281–286, 1999.

[Hemery *et al.*, 2006] Fred Hemery, Christophe Lecoutre, Lakhdar Sais, and Frédéric Boussemart. Extracting mucs from constraint networks. In *Proceedings of ECAI 2006*, volume 141, pages 113–117. IOS Press, 2006.

[Ignatiev *et al.*, 2015] Alexey Ignatiev, Alessandro Previti, Mark H. Liffiton, and João Marques-Silva. Smallest MUS extraction with minimal hitting set dualization. In *Proceedings of CP 2015*, pages 173–182, 2015.

[Jannach *et al.*, 2015] Dietmar Jannach, Thomas Schmitz, and Kostyantyn M. Shchekotykhin. Parallelized hitting set computation for model-based diagnosis. In *Proceedings of AAAI 2015*, pages 1503–1510, 2015.

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

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

[Jussien and Barichard, 2000] Narendra Jussien and Vincent Barichard. The palm system: explanation-based constraint programming. In *Proc. of TRICS Workshop at CP*, 2000.

[Laborie, 2014] Philippe Laborie. An optimal iterative algorithm for extracting mucs in a black-box constraint network. In *Proceedings of ECAI*, pages 1051–1052, 2014.

[Li *et al.*, 2013] Hongbo Li, Haijiao Shen, Zhanshan Li, and Jinsong Guo. Reducing consistency checks in generating corrective explanations for interactive constraint satisfaction. *Knowl. Based Syst.*, 43:103–111, 2013.

[Liffiton and Malik, 2013] Mark H. Liffiton and Ammar Malik. Enumerating infeasibility: Finding multiple MUSes quickly. In *Proceedings of CPAIOR*, pages 160–175, 2013.

[Liffiton and Sakallah, 2005] Mark H. Liffiton and Karem A. Sakallah. On finding all minimally unsatisfiable subformulas. In *Proceedings of SAT*, pages 173–186, 2005.

[Liffiton and Sakallah, 2008] Mark H. Liffiton and Karem A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Auto. Reas.*, 40(1):1–33, 2008.

[Liffiton *et al.*, 2016] Mark H. Liffiton, Alessandro Previti, Ammar Malik, and João Marques-Silva. Fast, flexible MUS enumeration. *Constraints*, 21(2):223–250, 2016.

[Little *et al.*, 2002] James Little, Cormac Gebruers, Derek Bridge, and Eugene Freuder. Capturing constraint programming experience: A case-based approach, 2002.

[Lynce and Marques-Silva, 2004] Inês Lynce and João P. Marques-Silva. On computing minimum unsatisfiable cores. In *Proceedings of SAT 2004*, 2004.

[Marques-Silva and Mencía, 2020] João Marques-Silva and Carlos Mencía. Reasoning about inconsistent formulas. In *Proceedings of IJCAI 2020*, pages 4899–4906, 2020.

[Marques-Silva and Previti, 2014] Joao Marques-Silva and Alessandro Previti. On computing preferred MUSes and MCSes. In *Proceedings of SAT*, pages 58–74, 2014.

[Marques-Silva *et al.*, 2009] João P. Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In *Handbook of Satisfiability*, pages 131–153. 2009.

[Marques-Silva *et al.*, 2013a] João Marques-Silva, Federico Heras, Mikolás Janota, Alessandro Previti, and Anton Belov. On computing minimal correction subsets. In *Proceedings of IJCAI 2013*, pages 615–622, 2013.

[Marques-Silva *et al.*, 2013b] João Marques-Silva, Mikolás Janota, and Anton Belov. Minimal sets over monotone predicates in boolean formulae. In *Proceedings of CAV 2013*, volume 8044 of *LNCS*, pages 592–607, 2013.

[Marques-Silva *et al.*, 2015] João Marques-Silva, Mikolás Janota, Alexey Ignatiev, and António Morgado. Efficient model based diagnosis with maximum satisfiability. In *Proceedings of IJCAI*, pages 1966–1972, 2015.

[Mauss and Tatar, 2002] Jakob Mauss and Mugur M. Tatar. Computing minimal conflicts for rich constraint languages. In *Proceedings of ECAI*, pages 151–155, 2002.

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

[Metodi *et al.*, 2012] Amit Metodi, Roni Stern, Meir Kalech, and Michael Codish. Compiling model-based diagnosis to boolean satisfaction. In *Proceedings of AAAI*, 2012.

[Metodi *et al.*, 2014] Amit Metodi, Roni Stern, Meir Kalech, and Michael Codish. A novel sat-based approach to model based diagnosis. *J. Artif. Intell. Res.*, 51:377–411, 2014.

[Narodytska *et al.*, 2018] Nina Narodytska, Nikolaj Bjørner, Maria-Cristina V. Marinescu, and Mooly Sagiv. Core-guided minimal correction set and core enumeration. In *Proceedings of IJCAI 2018*, pages 1353–1361, 2018.

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

[O'Callaghan *et al.*, 2005] Barry O'Callaghan, Barry O'Sullivan, and Eugene C Freuder. Generating corrective explanations for interactive constraint satisfaction. In *Proceedings of CP 2005*, pages 445–459. Springer, 2005.

[Papadopoulos and O'Sullivan, 2009] Alexandre Papadopoulos and Barry O'Sullivan. Compiling all possible conflicts of a CSP. In *Proc. of CP*, pages 639–653, 2009.

[Previti *et al.*, 2018] Alessandro Previti, Carlos Mencía, Matti Järvisalo, and João Marques-Silva. Premise set caching for enumerating minimal correction subsets. In *Proceedings of AAAI 2018*, pages 6633–6640, 2018.

[Prosser, 1995] Patrick Prosser. MAC-CBJ: maintaining arc consistency with conflict-directed backjumping. Technical report, Tech Report 95/177, Dept.CS, Glasgow, 1995.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[Sabin and Freuder, 1994] Daniel Sabin and Eugene C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proc. of ECAI*, pages 125–129, 1994.

[Schutt *et al.*, 2011] Andreas Schutt, Thibaut Feydy, Peter J Stuckey, and Mark G Wallace. Explaining the cumulative propagator. *Constraints*, 16(3):250–282, 2011.

[Shchekotykhin *et al.*, 2015] Kostyantyn M. Shchekotykhin, Dietmar Jannach, and Thomas Schmitz. MergeXplain: Fast computation of multiple conflicts for diagnosis. In *Proceedings of IJCAI 2015*, pages 3221–3228, 2015.

[Sqalli and Freuder, 1996] Mohammed H. Sqalli and Eugene Freuder. Inference-based constraint satisfaction supports explanation. In *Proc. of AAAI*, pages 318–325, 1996.

[Stallman and Sussman, 1977] Richard M. Stallman and Gerald J. Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artif. Intell.*, 9(2):135–196, 1977.

[Stumptner and Wotawa, 2001] Markus Stumptner and Franz Wotawa. Diagnosing tree-structured systems. *Artif. Intell.*, 127(1):1–29, 2001.