Preferences in constraint satisfaction and optimization

Francesca Rossi, K. Brent Venable, Toby Walsh

Abstract

We review constraint-based approaches to handle preferences. We start by defining the main notions of constraint programming, then give various concepts of soft constraints and show how they can be used to model quantitative preferences. We then consider how soft constraints can be adapted to handle other forms of preferences, such as bipolar, qualitative, and temporal preferences. Finally, we describe how AI techniques such as abstraction, explanation generation, machine learning, and preference elicitation, can be useful in modelling and solving soft constraints.

Introduction

Preferences and constraints occur in real-life problems in many forms. Intuitively, *constraints* are restrictions on the possible scenarios. For a scenario to be feasible, all its constraints must be satisfied. For example, if we want to buy a PC, we may pose a lower limit on the size of its screen. Only PCs that respect this limit will be considered. Constraint programming (Rossi, Van Beek, & Walsh 2006) is an area of AI which provides the formalisms and solving techniques to model and solve problems with constraints.

Preferences, on the other hand, express desires, satisfaction levels, rejection degrees, or costs. For example, we may prefer a tablet PC to a regular laptop, we may desire having a webcam, and we may want to spend as little as possible. In this case, all PCs will be considered, but some will be more preferred than others. Such concepts can be expressed in either a qualitative or a quantitative way.

Preferences and constraints are closely related notions, since preferences can be seen as a form of "tolerant" constraints. For this reason, there are several constraint-based frameworks to model preferences. One of the most general framework, based on *soft constraints* (Meseguer, Rossi, & Schiex 2006), extends the classical constraint formalism to model preferences in a *quantitative* way, by expressing several degrees of satisfaction that can be either totally or partially ordered. When there are both levels of satisfaction and levels of rejection, preferences are *bipolar*, and can be modelled by extending the soft constraint formalism (Bistarelli *et al.* 2006).

Preferences can also be modelled in a *qualitative* way (also called *ordinal*), that is, by pairwise comparisons. In this case, soft constraints (or their extensions) are not suitable. However, other AI preference formalisms are able to express preferences qualitatively, such as CP-nets (Boutilier *et al.* 2004). CP-nets and soft constraints can be combined, providing a single environment where both qualitative and quantitative preferences can be modelled and handled.

Specific types of preferences come with their own reasoning methods. For example, *temporal* preferences are quantitative preferences that pertain to distances and durations of events in time. Soft constraints can be embedded naturally in a temporal constraint framework to handle this kind of preferences (Khatib *et al.* 2001; Peintner & Pollack 2004).

While soft constraints generalize the classical constraint formalism providing a way to model several kinds of preferences, this added expressive power comes at a cost, both in the modelling task as well as in the solving process. To mitigate these drawbacks, various AI techniques have been adopted. For example, *abstraction theory* (Cousot & Cousot 1977) has been exploited to simplify the process of finding a most preferred solution of a soft constraint problem (Bistarelli, Codognet, & Rossi 2002). Also, *explanations* have been considered to ease the understanding of the result of the solving process (Freuder *et al.* 2003).

On the modelling side, it may be tedious or too demanding of a user to specify all the soft constraints. *Machine learning techniques* have therefore been used to learn the missing preferences (Rossi & Sperduti 1998; Vu & O'Sullivan 2007). Alternatively, *preference elicitation* techniques (Chen & Pu 2004), interleaved with search and propagation, have been exploited to minimize the user's effort in specifying the problem while still being able to find a most preferred solution (Gelain *et al.* 2007).

Constraints

Constraint programming (Rossi, Van Beek, & Walsh 2006; Dechter 2003) is a powerful paradigm for solving combinatorial search problems that draws on a wide range of techniques from artificial intelligence, computer science, databases, programming languages, and operations research. Constraint programming is currently applied with success to many domains, such as scheduling, planning, vehicle routing, configuration, networks, and bioinformatics.

The basic idea in constraint programming is that the user states the constraints and a general purpose *constraint solver* is used to solve them.

Constraints are just relations, and a *constraint satisfaction problem* (CSP) states which relations should hold among the given decision variables. For example, in scheduling activities in a company, the decision variables might be the starting times and the durations of the activities and the resources needed to perform them, and the constraints might be on the availability of the resources and on their use by a limited number of activities at a time. Another example is configuration, where constraints are used to model compatibility requirements among components or user's requirements. For example, if we were to configure a laptop, some video boards may be incompatible with certain monitors. Also, the user may pose constraints on the weight and/or the screen size.

Constraint solvers take a real-world problem like this, represented in terms of decision variables and constraints, and find an assignment to all the variables that satisfies the constraints. Constraint solvers search the solution space either systematically, as with *backtracking* or *branch and bound* algorithms, or use forms of local search which may be incomplete. Systematic methods often interleave search and inference, where inference consists of propagating the information contained in one constraint to the neighboring constraints. Such inference (usually called *constraint propagation*) is useful since it may reduce the parts of the search space that need to be visited.

Rather than trying to satisfy a set of constraints, we may want to *optimize* them. This means that there is an objective function that measures the quality of each solution, and the aim is to find a solution with optimal quality, where the quality of a solution can be expressed in terms of preferences. For such problems, techniques such as branch and bound are usually used to find an optimal solution.

Modelling quantitative preference via soft constraints

While constraints have been successfully applied to many real-life combinatorial problems, in some cases the classical constraint framework is not expressive enough. For example, it is possible that after having listed the desired constraints among the decision variables, there is no way to satisfy them all. In this case, the problem is said to be *over-constrained*, and the model may be refined manually to ignore certain constraints. This process, when it is feasible, is rarely formalized and normally difficult and time consuming. Even when all the constraints can be satisfied, we may want to discriminate between the (equally good) solutions. These scenarios often occur when constraints are used to formalize desired properties rather than requirements that cannot be violated. Such desired properties are not faithfully represented by constraints, but should rather be considered as *preferences* whose violation should be avoided as far as possible.

As an example, consider a typical timetabling problem which aims at assigning courses and teachers to classrooms and time slots in a university. There are usually many constraints, such as the size of the classrooms, the opening hours of the building, or the fact that the same teacher cannot teach two different classes at the same time. However, there are usually also many preferences, which state for example the desires of the teachers (like that he prefers not to teach on Fridays), or also university policies (like that it is preferable to use smaller classrooms if possible). If all these preferences are modelled by constraints, it is easy to find scenarios where there is no way to satisfy all of them. However, there could be ways to satisfy all hard requirements while violating the desires as little as possible, which is what we are looking for in the real situation. Moreover, modelling preferences in a faithful way allows us to discriminate among all the solutions which satisfy the hard constraints. In fact, there could be two timetables which both satisfy the hard requirements, but where one of them satisfies better the desires, and this should be the chosen one. Similar scenarios can be found in most of the typical application fields for constraints, such as scheduling, resource allocation, rostering, vehicle routing, etc.

In general, preferences can be *quantitative* or *qualitative* (e.g. "I like this at level 10" versus "I like this more than that"). Preferences can also be conditional (e.g., "If the main dish is fish, I prefer white wine to red wine"). Preferences and constraints may also co-exist. For example, in a product configuration problem, there may be production constraints (for example, a limited number of convertible cars can be built each month), marketing preferences (for example, that it would be better to sell the standard paint types), whilst the user may have preferences of various kind (for example, that if it is sports car, she prefers red).

To cope with some of these scenarios, classical constraints have been generalized in various ways in the past decades. The underlying observation of all such generalizations is that classical constraints are relations, and thus they can either be satisfied or violated. Preferences need instead a notion that has several levels of satisfiability. In the early '90s, several attempts have been made to generalize the notion of constraint to an object with more than just two levels of satisfiability, also called a *soft constraint*.

For example, *fuzzy constraints* (Dubois, Fargier, & Prade 1993; Ruttkay 1994) allow for (discretized) preferences between 0 and 1. Then, the quality of a solution is the minimum preference associated to constraints for that solution. The aim is then to find a solution whose quality is highest. Since only the minimum preference is considered in a scenario, fuzzy constraints employ a pessimistic approach, that can be useful or even necessary in critical applications such as medical or aerospace ones. However, this so-called "drowning effect" (where the worst level of satisfiability "drowns" all the others) is too pessimistic in some cases. For this reason, *lexicographic constraints* were introduced (Fargier, Lang, & Schiex 1993), to obtain a more discriminating ordering of the solutions: to order two solutions, we compare lexicographically the ordered sequence of all the preferences given by the constraints to those two solutions. In this way, solutions with different minimum preferences are ordered as in the fuzzy constraint setting, but also solutions with the same minimum preference (that would be equally preferred in fuzzy constraints) can be discriminated.

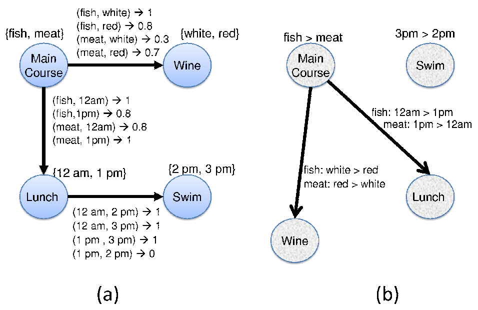

Figure 1: A fuzzy constraint problem (a) and a CP-net (b).

An example of a fuzzy constraint problem can be seen in Figure 1 (a). We are deciding on what to have for lunch and when to go swimming. The fuzzy CSP has four variables (represented by the circles), each with two values. For example, wine can be either red or white. There are three constraints, represented by solid arrows, and giving a preference value between 0 and 1 to each assignment of the variables of the constraint. An optimal solution of this fuzzy CSP is having fish with white wine at 12am and going for a swim at 3pm.

Another extension to classical constraints are the so-called *probabilistic constraints* (Fargier & Lang 1993), where, in the context of an uncertain model of the real world, each constraint is associated to the probability of being present in the real problem. Solutions are then associated to their conjoint probability (assuming independence of the constraints), and the aim is to find a solution with the highest probability.

In *weighted constraints*, instead, each constraint is associated a weight, and the aim is to find a solution for which the sum of the weights of the satisfied constraints is maximal. A very useful instance of weighted constraints are MaxCSPs, where weights are just 0 or 1 (0 if the constraint is violated and 1 if it is satisfied). In this case, we want to satisfy as many constraints as possible.

While fuzzy, lexicographic, and probabilistic constraints were defined to model real-life situations that could not be faithfully modelled via classical constraints, weighted constraints and MaxCSPs have mainly been defined to address over-constrained problems, where there are so many constraints that the problem has no solution. In this case, the attempt was to satisfy as many constraints as possible, possibly giving them some importance levels.

This second line of reasoning lead also to the definition of the first general framework to extend classical constraints, called *partial constraint satisfaction* (Freuder & Wallace 1992). In partial CSPs, over-constrained problems are addressed by defining a metric over constraint problems, and by trying to find a solution of a problem which is as close as possible to the given one according to the chosen metric. MaxCSPs are then just an instance of partial CSPs where the metric is based on the number of satisfied constraints.

Another general constraint-based formalism to model preferences is the *semiring-based formalism* (Bistarelli, Montanari, & Rossi 1997; Meseguer, Rossi, & Schiex 2006), that encompasses most of the previous extensions with the aim of providing a single environment where properties can be proven once and for all, and then inherited by all the instances. At the technical level, this is done by introducing a structure representing the levels of satisfiability of the constraints. Such a structure is just a set with two operations: one (written +) is used to generate an ordering over the levels, while the other one ($\times$) is used to define how two levels can be combined and which level is the result of such combination. Because of the properties required on such operations, this structure is usually a type of semiring. This gives *semiring-based soft constraint problems* (SCSPs), where constraints have several levels of satisfiability, that are (totally or partially) ordered according to the semiring structure. Fuzzy, lexicographic, probabilistic, weighted, and MaxCSPs are all instances of the semiring-based framework.

In the same year in which semiring-based soft constraints were introduced (1995), *valued constraints* were introduced as an alternative general formalism to model constraints with several levels of satisfiability (Schiex, Fargier, & Verfaillie 1995). Valued constraints are very similar to semiring-based soft constraints, except that their levels of satisfiability cannot be partially ordered (Bistarelli *et al.* 1999), and thus they can model only cardinal preferences.

The possibility of partially ordered sets of levels of satisfiability can be useful is several scenarios. When the levels are the result of the combination of several optimization criteria, it is natural to have a Pareto-like approach in combining such criteria, and this leads to a partial order. Also, even if we have just one optimization criterion, we may want to insist on declaring some levels as incomparable, because of what they model. In fact, the elements of the semiring structure do not need to be numbers, but can be any object that we want to associate to a way of giving values to the variables of a constraint. If, for example, the objects are all the subsets of a certain set, then we can have a partial order under subset inclusion.

Given two solutions of a soft constraint problem, checking whether one is preferable to the other one is easy: we compute the desirability values of the two solutions and compare them in the preference order. However, finding an optimal solution for a soft constraint problem is a combinatorially difficult problem. Many search techniques have been developed to solve specific classes of soft constraints, like fuzzy or weighted. However, all have an exponential worst case complexity. Systematic approaches like backtracking search and constraint propagation can be adapted to soft constraints. For example, backtracking search gives branch and bound where the bounds are given by the preference levels in the constraints. Constraint propagation, which is very successful in pruning parts of the search tree in constraint solving, can also be generalized to certain classes of soft constraints.

By the way soft constraints are defined, they are able to model quantitative preferences. They cannot model directly conditional and/or qualitative preferences.

Other kinds of preferences

Bipolar preferences

Bipolarity is an important topic in several fields, such as psychology and multi-criteria decision making, and it has recently attracted interest in the AI community, especially in argumentation (Amgoud, Bonnefon, & Prade 2005), qualitative reasoning (Dubois & Fargier 2005; 2006), and decision theory (Labreuche & Grabisch 2006). Bipolarity in preference reasoning can be seen as the possibility to stating both degrees of satisfaction (that is, *positive* preferences) and degrees of rejection (that is, *negative* preferences).

Positive and negative preferences can be thought as two symmetric concepts, and thus one might try to deal with them via the same operators. However, this may not model what one usually expects in real scenarios. For example, if we have a dinner menu with fish and white wine, and we like them both, then having both should be more preferred than having just one of them. On the other hand, if we don't like any of them, then the preference of the having them both should be smaller than the preferences of each of them alone. In fact, the combination of positive preferences should usually produce a higher (positive) preference, while the combination of negative preferences should usually give us a lower (negative) preference.

When dealing with both kinds of preferences, it is natural to express also *indifference*, which means that we express neither a positive nor a negative preference over an object. Moreover, we also want to be able to combine positive with negative preferences. The most natural and intuitive way to do so is to allow for *compensation*. Comparing positive against negative aspects and compensating them w.r.t. their strength is one of the core features of decision-making processes, and it is, undoubtedly, a tactic universally applied to solve many real life problems.

Positive and negative preferences might seem as just two different criteria to reason with, and thus techniques such as those usually adopted by multi-criteria optimization (Ehrgott & Gandibleux 2002), such as Pareto-like approaches, could appear suitable for dealing with them. However, this interpretation would hide the fundamental nature of bipolar preferences, that is, positive preferences are naturally the opposite of negative preferences.

Soft constraints can only model negative preferences, since in this framework preference combination returns lower preferences. However, soft constraints can be generalized to model both positive and negative preferences (Bistarelli *et al.* 2006), and preference compensation is allowed.

Bipolarity has been considered also in qualitative preference reasoning (Benferhat *et al.* 2002; 2006), where fuzzy preferences model the positive knowledge and negative preferences are interpreted as violations of constraints. Precedence is given to negative preference optimization, and positive preferences are used to distinguish among the optimals found in the first phase, thus not allowing for compensation. Another approach (Grabisch, de Baets, & Fodor 2003) considers totally ordered unipolar and bipolar preference scales and defines an operator, the *uninorm*, which can be seen as a restricted form of compensation.

Qualitative preferences

CP-nets (Boutilier *et al.* 2004) (Conditional Preference networks) are a graphical model for compactly representing conditional and qualitative preference relations. Qualitative preferences are also called *ordinal* preferences, since they are modelled via an ordering over a set of alternatives.

CP-nets exploit conditional preferential independence among the features of a problem by structuring a user's possibly complex preference ordering via a set of preference statements (called *cp-statements*), interpreted under the *ceteris paribus* assumption. For instance, the statement "I prefer red wine to white wine if meat is served" asserts that, given two meals that differ only in the kind of wine served and both containing meat, the meal with a red wine is preferable to the meal with a white wine.

Figure 1 (b) shows a CP-net with the following cp-statements: fish is better than meat; swimming later is preferred; white wine is preferred to red wine if fish is served, otherwise red wine is better; an earlier lunch is preferred to a late one if there is fish, otherwise later is better.

CP-nets bear some similarity to Bayesian networks, as both utilize directed acyclic graphs where each node stands for a domain variable, and assume a set of features with finite, discrete domains (these play the same role as variables in soft constraints). Given a CP-net, an ordering is induced over the set of assignments of its features. This ordering is, in the most general case, a preorder (that is, reflexive and transitive). However, this preorder is not general, since two assignments that differ for the value of one variable are always ordered in a CP-net (that is, they are never incomparable).

Given an acyclic CP-net, finding an optimal assignment to its features can be done in linear time. However, for cyclic CP-nets, it becomes NP-hard. Moreover, comparing two outcomes is PSPACE-complete in general, and remains NP-hard even when the CP-net is acyclic.

Quantitative and qualitative preferences

It would be nice to have a single formalism for representing preferences of several kinds. To achieve this goal, we start by comparing the expressive power of soft constraints and CP-nets.

We could say that a formalism B is at least as expressive as a formalism A if and only if from a problem expressed using A it is possible to build in polynomial time a problem expressed using B such that the optimal solutions are the same. If we apply this definition to soft constraints, we see, for example, that fuzzy CSPs and weighted CSPs are at least as expressive as classical constraints. If instead we use it to compare CP-nets and soft constraints, we see that classical constraints are at least as expressive as CP-nets. In fact, it is possible to show that, given any CP-net, we can obtain in polynomial time a set of classical constraints whose solutions are the optimal outcomes of the CP-net (Brafman & Dimopoulos 2004). On the contrary, there are some classical constraint problems for which it is not possible to build in polynomial time a CP-net with the same set of optimals.

However, we could be more fine-grained in the comparison, and say that a formalism B is at least as expressive as a formalism A if and only if from a problem expressed using A it is possible to build in polynomial time a problem expressed using B such that the orderings over solutions are the same. Here not only we must maintain the set of optimals, but also the rest of the ordering over the solutions. In this case, CP-nets and soft constraints are incomparable.

However, it is possible to approximate a CP-net ordering via soft constraints, achieving tractability while sacrificing precision to some degree. Different approximations can be characterized by how much of the original ordering they preserve, the time complexity of generating the approximation, and the time complexity of comparing outcomes in the approximation. It is vital that such approximations are information preserving; that is, what is ordered in the given ordering is also ordered in the same way in the approximation. Another desirable property of approximations is that they preserve the ceteris paribus property. CP-nets can be approximated by soft constraints where the optimization criterion is the minimization of the sum of the preferences, and also by soft constraints where a fuzzy-based ordering is adopted. In both cases, the approximation is information preserving and satisfies the ceteris paribus property (Domshlak *et al.* 2003). For example, the CP-net in Figure 1 (b) is approximated by the fuzzy CSP in Figure 1 (a).

Summarizing, CP-nets and soft constraints have complementary advantages and drawbacks. CP-nets allow one to represent conditional and qualitative preferences, but dominance testing is expensive. On the other hand, soft constraints allow to represent both hard constraints and quantitative preferences, and have a cheap dominance testing.

Many problems have both constraints and preferences. Unfortunately, reasoning with them both is difficult as often the most preferred outcome is not feasible, and not all feasible outcomes are equally preferred. If we put together a CP-net and a set of constraints, it is possible to obtain all the optimal outcomes by solving a set of hard "optimality constraints" (Prestwich *et al.* 2005). In well defined cases, this avoids expensive dominance testing.

Temporal preferences

Soft constraints have been used also to model preferences in the context of temporal reasoning. Reasoning about time is a core issue in many real life problems, such as planning and scheduling for production plants, transportation, and space missions. Several approaches have been proposed to reason about temporal information. *Temporal constraints* have been among the most successful in practice.

In temporal constraint problems, variables either represent instantaneous events, such has "when a plane takes off", or temporal intervals, such as "the duration of the flight". Temporal constraints allow one to put temporal restrictions either on when a given event should occur, e.g. "the plane must take off before 10am", or on how long a given activity should last, e.g. "the flight should not last more than two hours".

Several quantitative and qualitative constraint-based temporal formalisms have been proposed, stemming from pioneering works by Allen (Allen 1983) and by Dechter, Meiri, and Pearl (Dechter, Meiri, & Pearl 1991).

In general, solving temporal constraint problems is difficult. However, there are tractable classes, such as quantitative temporal constraint problems where there is only one temporal interval for each constraint (Dechter, Meiri, & Pearl 1991).

The expressive power of classical temporal constraints may be insufficient to model faithfully all the aspects of the problem. For example, one may want to say that "the earlier the plane takes off, the better". Both qualitative and quantitative temporal reasoning formalisms have been extended with *quantitative preferences* to allow for the specification of such a kind of statements.

More precisely, Allen's approach has been augmented with fuzzy preferences (Badaloni, Falda, & Giacomin 2004), that are associated with the relations among temporal intervals allowed by the constraints. Such problems are solved by exploiting some properties of fuzzy preferences, in order to decompose the optimization problem into solving a set of classical constraint problems.

Fuzzy preferences have been combined also with non-disjunctive and disjunctive quantitative temporal constraints (Khatib *et al.* 2007; Peintner & Pollack 2004). The result are soft temporal constraints where each allowed duration or occurrence time for a temporal event is associated to a fuzzy preference representing the desirability of that specific time. The decomposition approach is the most efficient solving technique also in the quantitative setting (Khatib *et al.* 2001; 2007; Peintner & Pollack 2004).

Quantitative temporal constraints have also been extended with *utilitarian* preferences: preferences take values in the set of positive reals and the goal it to maximize their sum. Such problems have been solved using adapted branch and bound techniques as well as SAT and weighted constraint satisfaction approaches (Moffitt & Pollack 2006; Peintner & Pollack 2005).

Mastering the complexity of soft constraints

In constraint satisfaction problems we look for a solution, while in soft constraint problems we look for an optimal solution. Thus, soft constraint problems are more difficult to handle by a solver. To ease this difficulty, several AI techniques have been used. Here we cite just two of them: abstraction and explanation generation. Abstraction works on a simplified version of the given problem, thus hoping to have a significantly smaller search space, while explanation generation helps understand the result of the solver. It is not always easy for a user to understand why no better solution is returned.

An added difficulty in dealing with soft constraints comes also in the *modelling phase*, where a user has to understand how to model faithfully his real-life problem via soft constraints. In many cases, we may end up with a soft constraint problem where some preferences are missing. To reason in this scenario, we may use techniques like machine learning and preference elicitation to solve the problem.

Abstraction

Soft constraints are much more expressive than classical CSPs, but they are also more difficult to process and to solve. Therefore, sometimes it may be too costly to find all, or even only one, optimal solution. Also, although classical propagation techniques like arc-consistency can be extended to soft constraints (Meseguer, Rossi, & Schiex 2006), such techniques can be too costly to be used, depending on the size and structure of the partial order associated to the problem. Finally, sometimes we may not have a solver for the class of soft constraints we need to solve, while we may have a solver for another "simpler" class of soft constraints.

For these reasons, it may be reasonable to work on a simplified version of the given soft constraint problem, trying not to lose too much information. Such a simplified version can be defined by means of the notion of abstraction, which takes an SCSP and returns a new one which is simpler to solve. Here, as in many other works on abstraction, "simpler" may mean many things, like the fact that a certain solution algorithm finds a solution, or an optimal solution, in a fewer number of steps, or also that the abstracted problem can be processed by a machinery which is not available in the concrete context.

To define an abstraction, we may use for example the theory of *Galois insertions* (Cousot & Cousot 1977), that provides a formal approach to model the simplification of a mathematical structure with its operators. Given an SCSP (the *concrete* one), we may get an abstract SCSP by just simplifying the associated semiring, and relating the two structures (the concrete and the abstract one) via a Galois insertion. Note that this way of abstracting constraint problems does not change the structure of the problem (the set of variables remains the same, as well as the set of constraints), but just the semiring values to be associated to the tuples of values for the variables in each constraint (Bistarelli, Codognet, & Rossi 2002).

Abstraction has been used also to simplify the solution process of hard constraint problems (Lecoutre *et al.* 2000). Also, the notion of value interchangeability has been exploited to support abstraction and reformulation of hard constraint problems (Freuder & Sabin 1997). However, in the case of hard constraints, abstracting a constraint problem means dealing with fewer variables and smaller domains.

Once we reason on the abstracted version of a problem, we can bring back to the original problem some (or possibly all) of the information derived in the abstract context, and then continue the solution process on the transformed problem, which is a concrete problem equivalent to the given one. The hope is that, by following this route, we get to the final goal faster than just solving the original problem.

It is also possible to define iterative hybrid algorithms which can approximate an optimal solution of a soft constraint problem by solving a series of problems which abstract, in different ways, the original problem. These are anytime algorithms since they can be stopped at any phase, giving better and better approximations of an optimal solution.

Explanation generation

One of the most important features of problem solving in an *interactive setting* is the capacity of the system to provide the user with justifications, or explanations, for its operations. Such justifications are especially useful when the user is interested in what happens at any time during search, because he/she can alter features of the problem to facilitate the problem solving process.

Basically, the aim of an explanation is to show clearly why a system acted in a certain way after certain events. Explanations have been used for hard constraint problems, especially in the context of over-constrained problems (Junker 2004; Jussien & Barichard 2000; Amilhastre, Fargier, & Marquis 2002), to understand why the problem does not have a solution and what can be modified in order to get one. In soft constraint problems, explanations should also take preferences into account, and provide a way to understand, for example, why there is no way to get a better solution.

In addition to providing explanations, interactive systems should be able to show the consequences, or implications, of an action to the user, which may be useful in deciding which choice to make next. In this way, they can provide a sort of "what-if" kind of reasoning, which guides the user towards good future choices. Fortunately, in soft constraint problems this capacity can be implemented with the same machinery that is used to give explanations.

A typical example of an interactive system where constraints and preferences may be used, and where explanations can be very useful, are configurators. A configurator is a system which interacts with a user to help him/her to configure a product. A product can be seen as a set of component types, where each type corresponds to a certain finite number of concrete components, and a set of compatibility constraints among subsets of the component types. A user configures a product by choosing a concrete component for each component type, such that all the compatibility constraints as well as personal preferences are satisfied. For example, in a car configuration problem, a user may prefer red cars, but may also not want to completely rule out other colors.

Constraint-based technology is currently used in many configurators to both model and solve configuration problems: component types are represented by variables, having as many values as the concrete components, and both compatibility and personal constraints are represented as constraints (or soft constraints) over subsets of such variables. At present, user choices during the interaction with the configurator are usually restricted to specifying unary constraints, in which a certain value is selected for a variable.

Whenever a choice is made, the corresponding (unary) constraint is added to existing compatibility and personal constraints, and some constraint propagation notion is enforced, for example arc-consistency (AC) (Rossi, Van Beek, & Walsh 2006), to rule out (some of the) future choices that are not compatible with the current choice. While providing justifications based on search is difficult, arc-consistency enforcing has been used as a source of guidance for justifications, and it has been exploited to help the users in some of the scenarios mentioned above. For example, it has been shown that AC enforcement can be used to provide both justifications for choice elimination, and also guidance for conflict resolution (Freuder, Likitvivatanavong, & Wallace 2001).

The same approach can be used also for configurators with preferences, using a generalized version of arc-consistency, whose application may decrease the preferences in some constraints. Explanations can then describe why the preferences for some values decrease, and suggest at the same time which assignment has to be retracted, in order to maximize the evaluation of a solution.

Configurators with soft constraints should help users not only to avoid conflicts or to make the next choice so that a smaller number of later choices are eliminated, but also to get to an optimal (or good enough) solution. More precisely, when the user is about to make a new choice for a component type, the configurator should show the consequences of such a choice in terms of conflicts generated, elimination of subsequent choices, and also quality of the solutions. In this way, the user can make a choice which leads to no conflict, and which presents a good compromise between choice elimination and solution quality.

Learning

In a soft constraint problem, sometimes one may know his/her preferences over some of the solutions, but have no idea on how to code this knowledge into the constraints of the problem. Such a scenario has been theoretically addressed (Rossi & Sperduti 1998) by using machine learning techniques based on gradient descent. Soft constraint learning has also been embedded in a general interactive constraint framework, where users can state both usual preferences over constraints and also preferences over solutions proposed by the system (Rossi & Sperduti 2004). Other approaches to learning soft constraints that provide a unifying framework for soft CSP learning have recently been developed (Vu & O'Sullivan 2007). Moreover, soft constraint learning has been exploited in the application domain of processing and interpreting satellite images (Michalowski *et al.* 2007).

Machine learning techniques have also been used to learn hard constraints (that is, to learn allowed and forbidden variable instantiations). For example, an interactive technique based on a hypothesis space containing the possible constraints to learn has been used to help the user formulate his constraints (O'Connell, O'Sullivan, & Freuder 2002).

Elicitation

Preference elicitation is a well-studied discipline in AI and other fields (Chen & Pu 2004). Preference elicitation may be costly. For example, asked preferences may need some work to be computed, or users may be reluctant to provide some of their preferences for privacy concerns. Therefore the usual aim is to minimize the amount of preference elicited.

With soft constraints, the task is to find an optimal solution. When some preferences are missing, we can consider two notions of optimal solutions: *possibly optimal* solutions are assignments to all the variables that are optimal in at least one way currently unspecified preferences can be revealed, while *necessarily optimal solutions* are assignments to all the variables that are optimal in all ways in which currently unspecified preferences can be revealed.

Given an incomplete soft CSP, its set of possibly optimal solutions is never empty, while the set of necessarily optimal solutions can be empty. Of course what we would like to find is a necessarily optimal solution: such solutions are optimal regardless of how the missing preferences would be specified. However, if this set is empty, we can interleave search and preference elicitation. More precisely, we can ask the user to provide some of the missing preferences and try to find, if any, a necessarily optimal solution of the new incomplete soft CSP. Then we can repeat the process until the current problem has at least one necessarily optimal solution. Experimental results show that this process ends after eliciting a very small percentage (as low as 10%) of the missing preferences (Gelain *et al.* 2007).

Other approaches elicit not just preferences but also values in the variable domains (Faltings & Macho-Gonzalez 2005), and consider either fuzzy or weighted constraints. Moreover, preference elicitation can also be used to elicit hard constraints (that is, to know if some partial assignments are allowed or not). In this context, the approach in (Wilson, Grimes, & Freuder 2007) associates a cost to each missing item, as well as a probability of being allowed. Then, while still interleaving search and elicitation, search is guided by such costs and probabilities, with the aim of minimizing the elicitation effort.

Future Perspectives

There are many directions for future work, both from a representational and a reasoning perspective. For instance, a very active area of research currently is reasoning about *incompletely specified preferences*. This line of research involves also issues related to preference elicitation, sensitivity analysis, uncertainty, and robustness. Another active line of research is in developing *soft global constraints*. Global constraints are very useful in CSPs since they are equipped with very efficient propagation algorithms. It would be very useful to define similar global constraints also in the soft constraint framework. It would also be interesting to study *quantified soft constraint problems*, where preferences can be associated to quantified statements.

The recently born area of *computational social choice* is devoted to exploit computer science results in the field of social choice, and addresses issues in topics like multi-agent preference aggregation. Like other multi-disciplinary topics, this area shows considerable promise. Some work in this area has already lead to interesting results where complexity has been used to mitigate some of the classical impossibility results proven by social choice for aggregating multi-agent preferences.

Many classical CSP applications, such as scheduling and resource allocation, can benefit from the ability to handle preferences. Moreover, newly emerging applications are heavily based on a flexible, adaptive, and tolerant behavior, that can be found only in preference-based approaches. For example, as synthetic avatars interact more and more with humans, it is crucial to equip them with the ability to extract, model, and handle humans' preferences.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(1):832–843.

Amgoud, L.; Bonnefon, J.-F.; and Prade, H. 2005. An argumentation-based approach to multiple criteria decision. In *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005)*, volume 3571 of *LNCS*, 269–280. Springer.

Amilhastre, J.; Fargier, H.; and Marquis, P. 2002. Consistency restoration and explanations in dynamic CSPs - application to configuration. *Artificial Intelligence* 135(1-2):199–234.

Badaloni, S.; Falda, M.; and Giacomin, M. 2004. Integrating quantitative and qualitative constraints in fuzzy temporal networks. *AI Communications* 17(4):183–272.

Benferhat, S.; Dubois, D.; Kaci, S.; and Prade, H. 2002. Bipolar representation and fusion of preferences on the possibilistic logic framework. In *Proceedings of the Eight International Conference on Principles and Knowledge Representation and Reasoning (KR-02)*, 421–448. Morgan Kaufmann.

Benferhat, S.; Dubois, D.; Kaci, S.; and Prade, H. 2006. Bipolar possibility theory in preference modeling: Representation, fusion and optimal solutions. *Information Fusion* 7(1):135–150.

Bistarelli, S.; Montanari, U.; Rossi, F.; Schiex, T.; Verfaillie, G.; and Fargier, H. 1999. Semiring-based CSPs and Valued CSPs: Frameworks, properties, and comparison. *Constraints* 4(3):199–240.

Bistarelli, S.; Pini, M. S.; Rossi, F.; and Venable, K. B. 2006. Bipolar preference problems: Framework, properties and solving techniques. In *Recent Advances in Constraints (CSCLP 2006)*, volume 4651 of *LNCS*, 78–92. Springer.

Bistarelli, S.; Codognet, P.; and Rossi, F. 2002. Abstracting soft constraints: Framework, properties, examples. *Artif. Intell.* 139(2):175–211.

Bistarelli, S.; Montanari, U.; and Rossi, F. 1997. Semiring-based constraint satisfaction and optimization. *J. ACM* 44(2):201–236.

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)* 21:135–191.

Brafman, R. I., and Dimopoulos, Y. 2004. A new look at the semantics and optimization methods of CP-networks. *Computational Intelligence* 20(2):218–245.

Chen, L., and Pu, P. 2004. Survey of preference elicitation methods. Technical Report IC/200467, Swiss Federal Institute of Technology in Lausanne (EPFL).

Cousot, P., and Cousot, R. 1977. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth ACM Symposium on Principles of Programming Languages (POPL 1977)*, 238–252.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Domshlak, C.; Rossi, F.; Venable, K. B.; and Walsh, T. 2003. Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 215–220. Morgan Kaufmann.

Dubois, D., and Fargier, H. 2005. On the qualitative comparison of sets of positive and negative affects. In *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005)*, volume 3571 of *LNCS*, 305–316. Springer.

Dubois, D., and Fargier, H. 2006. Qualitative decision making with bipolar information. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR-06)*, 175–186. AAAI Press.

Dubois, D.; Fargier, H.; and Prade, H. 1993. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In *2nd IEEE Int. Conf. on Fuzzy Systems*. IEEE.

Ehrgott, M., and Gandibleux, X., eds. 2002. *Multiple Criteria Optimization. State of the art annotated bibliographic surveys*. Kluwer Academic, Dordrecht.

Faltings, B., and Macho-Gonzalez, S. 2005. Open constraint programming. *AI Journal* 161(1-2):181–208.

Fargier, H., and Lang, J. 1993. Uncertainty in constraint satisfaction problems: a probabilistic approach. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU 1993)*, volume 747 of *LNCS*, 97–104. Springer.

Fargier, H.; Lang, J.; and Schiex, T. 1993. Selecting preferred solutions in fuzzy constraint satisfaction problems. In *Proceedings of the First European Congress on Fuzzy and Intelligent Technologies (EUFIT-93)*. Verlag der Augustinus Buchhandlung, Aachen.

Freuder, E. C., and Sabin, D. 1997. Interchangeability supports abstraction and reformulation for multi-dimensional constraint satisfaction. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 191–196. AAAI Press / The MIT Press.

Freuder, E. C., and Wallace, R. J. 1992. Partial constraint satisfaction. *Artif. Intell.* 58(1-3):21–70.

Freuder, E. C.; Likitvivatanavong, C.; Moretti, M.; Rossi, F.; and Wallace, R. J. 2003. Computing explanations and implications in preference-based configurators. In *Recent Advances in Constraints (CSCLP 2003)*, volume 2627 of *LNCS*, 76–92. Springer.

Freuder, E. C.; Likitvivatanavong, C.; and Wallace, R. J. 2001. Deriving explanations and implications for constraint satisfaction problems. In *Proceedings of the 7th International Conference of Principles and Practice of Constraint Programming (CP 2001)*, volume 2239 of *LNCS*, 585–589. Springer.

Gelain, M.; Pini, M. S.; Rossi, F.; and Venable, K. B. 2007. Dealing with incomplete preferences in soft constraint problems. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP 2007)*, volume 4741 of *LNCS*, 286–300. Springer.

Grabisch, M.; de Baets, B.; and Fodor, J. 2003. The quest for rings on bipolar scales. *Int. Journ. of Uncertainty, Fuzziness and Knowledge-Based Systems*.

Junker, U. 2004. Quickxplain: Preferred explanations and relaxations for over-constrained problems. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*, 167–172. AAAI Press / The MIT Press.

Jussien, N., and Barichard, V. 2000. The PaLM system: explanation-based constraint programming. In *Proceedings of TRICS: Techniques foR Implementing Constraint programming Systems, a post-conference workshop of CP 2000*, 118–133.

Khatib, L.; Morris, P. H.; Morris, R. A.; and Rossi, F. 2001. Temporal constraint reasoning with preferences. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, 322–327. Morgan Kaufmann.

Khatib, L.; Morris, P. H.; Morris, R.; Rossi, F.; Sperduti, A.; and Venable, K. B. 2007. Solving and learning a tractable class of soft temporal constraints: Theoretical and experimental results. *AI Commun.* 20(3):181–209.

Labreuche, C., and Grabisch, M. 2006. Generalized Choquet-like aggregation functions for handling bipolar scales. *European Journal of Operational Research* 172(3):931–955.

Lecoutre, C.; Merchez, S.; Boussemart, F.; and Grégoire, E. 2000. A CSP abstraction framework. In *Proceedings of the 4th International Symposium on Abstraction, Reformulation, and Approximation (SARA 2000)*, volume 1864 of *LNCS*. Springer.

Meseguer, P.; Rossi, F.; and Schiex, T. 2006. Soft constraints. In *Handbook of constraint programming*. Elsevier. chapter 9, 281–328.

Michalowski, M.; Knoblock, C. A.; Bayer, K. M.; and Choueiry, B. Y. 2007. Exploiting automatically inferred constraint-models for building identification in satellite imagery. In *Proceedings of the 15th ACM International Symposium on Geographic Information Systems,(ACM-GIS 2007)*, 6. ACM.

Moffitt, M. D., and Pollack, M. E. 2006. Temporal preference optimization as weighted constraint satisfaction. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI 2006)*. AAAI Press.

O'Connell, S.; O'Sullivan, B.; and Freuder, E. 2002. Strategies for interactive constraint acquisition. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming, (CP 2002)*, volume 2470 of *LNCS*. Springer.

Peintner, B., and Pollack, M. E. 2004. Low-cost addition of preferences to DTPs and TCSPs. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, (AAAI 2004)*, 723–728. AAAI Press / The MIT Press.

Peintner, B., and Pollack, M. E. 2005. Anytime, complete algorithm for finding utilitarian optimal solutions to STPPs. In *Proceedings of The Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, 443–448. AAAI Press / The MIT Press.

Prestwich, S. D.; Rossi, F.; Venable, K. B.; and Walsh, T. 2005. Constraint-based preferential optimization. In *AAAI*, 461–466. AAAI Press / The MIT Press.

Rossi, F., and Sperduti, A. 1998. Learning solution preferences in constraint problems. *J. Exp. Theor. Artif. Intell.* 10(1):103–116.

Rossi, F., and Sperduti, A. 2004. Acquiring both constraint and solution preferences in interactive constraint systems. *Constraints* 9(4):311–332.

Rossi, F.; Van Beek, P.; and Walsh, T., eds. 2006. *Handbook of Constraint Programming*. Elsevier.

Ruttkay, Z. 1994. Fuzzy constraint satisfaction. In *3rd IEEE Int. Conf. on Fuzzy Systems*. IEEE.

Schiex, T.; Fargier, H.; and Verfaillie, G. 1995. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 95)*, 631–639. Morgan Kaufmann.

Vu, X.-H., and O'Sullivan, B. 2007. Semiring-based constraint acquisition. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, 251–258. IEEE Computer Society.

Wilson, N.; Grimes, D.; and Freuder, E. C. 2007. A cost-based model and algorithms for interleaving solving and elicitation of CSPs. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP 2007)*, volume 4741 of *LNCS*, 666–680. Springer.