Dealing with Incomplete Preferences in Soft Constraint Problems

Mirco Gelain, Maria Silvia Pini, Francesca Rossi, and K. Brent Venable

Dipartimento di Matematica Pura ed Applicata, Università di Padova, Italy
{mgelain,mpini,frossi,kvenable}@math.unipd.it

Abstract. We consider soft constraint problems where some of the preferences may be unspecified. This models, for example, situations with several agents providing the data, or with possible privacy issues. In this context, we study how to find an optimal solution without having to wait for all the preferences. In particular, we define an algorithm to find a solution which is necessarily optimal, that is, optimal no matter what the missing data will be, with the aim to ask the user to reveal as few preferences as possible. Experimental results show that in many cases a necessarily optimal solution can be found without eliciting too many preferences.

1 Introduction

Traditionally, tasks such as scheduling, planning, and resource allocation have been tackled using several techniques, among which constraint reasoning is one of the winning ones: the task is represented by a set of variables, their domains, and a set of constraints, and a solution of the problem is an assignment to all the variables in their domains such that all constraints are satisfied. Preferences or objective functions have been used to extend such scenario and allow for the modelling of constraint optimization, rather than satisfaction, problems. However, what is common to all these approaches is that the data (variables, domains, constraints) are completely known before the solving process starts.

On the contrary, the increasing use of web services and in general of multi-agent applications demands for the formalization and handling of data that is only partially known when the solving process works, and that can be added later, for example via elicitation. In many web applications, data may come from different sources, which may provide their piece of information at different times. Also, in multi-agent settings, data provided by some agents may be voluntarily hidden due to privacy reasons, and only released if needed to find a solution to the problem.

Recently, some lines of work have addressed these issues by allowing for open settings in CSPs: both open CSPs [5,7] and interactive CSPs [11] work with domains that can be partially specified, and in dynamic CSPs [4] variables, domains, and constraints may change over time. It has been shown that these approaches are closely related. In fact, interactive CSPs can be seen as a special case of both dynamic and open CSPs [10].

Here we consider the same issues but we focus on constraint optimization problems rather than CSPs, thus looking for an optimal solution rather than any solution. In particular, we consider problems where constraints are replaced by soft constraints, in which

C. Bessiere (Ed.): CP 2007, LNCS 4741, pp. 286–300, 2007.

each assignment to the variables of the constraint has an associated preference coming from a preference set [1]. In this setting, for the purpose of this paper we assume that variables, domains, and constraint topology are given at the beginning, while the preferences can be partially specified and possibly added during the solving process.

Constraint optimization has also been considered in the context of open CSPs, in a cost-minimization setting in [6] and in a fuzzy setting in [7]. However, the incompleteness considered in [7,6] is on domain values as well as on their preferences or costs. We assume instead that all values are given at the beginning, and that some preferences are missing. Because of the setting of [9], the assumption that new values and costs are provided monotonically is needed, while it is not necessary here. Working under this assumption means that the agent that provides new values/costs for a variable knows all possible costs, since it is capable of providing the best value first. If the cost computation is expensive or time consuming, then computing all such costs (in order to give the most preferred value) is not desirable. This is not needed in our setting, where single preferences are elicited.

There are several application domains where such setting is useful. One regards the fact that quantitative preferences, and needed in soft constraints, may be difficult and tedius to provide for a user. Another one concerns multi-agent settings, where agents agree on the structures of the problem by they may provide their preferences on different parts of the problem at different times. Finally, some preferences can be initially hidden because of privacy reasons.

Formally, we take the soft constraint formalism when preferences are totally ordered and we allow for some preferences to be left unspecified. Although some of the preferences can be missing, it could still be feasible to find an optimal solution. If not, then we ask the user and we start again from the new problem with some added preferences.

More precisely, we consider two notions of optimal solution: *possibly optimal* solutions are assignments to all the variables that are optimal in *at least one way* in which 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. This notation comes from multi-agent preference aggregation [12], where, in the context of voting theory, some preferences are missing but still one would like to declare a winner.

Given an incomplete soft constraint problem (ISCSP), 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, to be on the safe side: such solutions are optimal regardless of how the missing preferences would be specified. However, since such a set may be empty, in this case there are two choices: either to be satisfied with a possibly optimal solution, or to ask users to provide some of the missing preferences and try to find, if any, a necessarily optimal solution of the new ISCSP. In this paper we follow this second approach, and we repeat the process until the current ISCSP has at least one necessarily optimal solution.

To achieve this, we employ an approach based on branch and bound which first checks whether the given problem has a necessarily optimal solution (by just solving the completion of the problem where all unspecified preferences are replaced by the worst preference). If not, then finds the most promising among the possibly optimal

solutions (where the promise is measured by its preference level), and asks the user to reveal the missing preferences related to such a solution. This second step is then repeated until the current problem has a necessarily optimal solution.

We implemented our algorithm and we tested it against classes of randomly generated binary fuzzy ISCSPs, where, beside the usual parameters (number of variables, domain size, density, and tightness), we added a new parameters measuring the percentage of unspecified preferences in each constraint and domain. The experimental results show that in many cases a necessarily optimal solution can be found by eliciting a small amount of preferences.

2 Soft Constraints

A soft constraint [1] is just a classical constraint [3] where each instantiation of its variables has an associated value from a (totally or partially ordered) set. This set has two operations, which makes it similar to a semiring, and is called a c-semiring. More precisely, a c-semiring is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ such that: A is a set, called the carrier of the c-semiring, and $\mathbf{0}, \mathbf{1} \in A$; $+$ is commutative, associative, idempotent, $\mathbf{0}$ is its unit element, and $\mathbf{1}$ is its absorbing element; $\times$ is associative, commutative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element.

Consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. Then: $\leq_S$ is a partial order; $+$ and $\times$ are monotone on $\leq_S$; $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum; $\langle A, \leq_S\rangle$ is a lattice and, for all $a, b \in A$, $a + b = lub(a, b)$. Moreover, if $\times$ is idempotent, then $\langle A, \leq_S\rangle$ is a distributive lattice and $\times$ is its glb.

Informally, the relation $\leq_S$ gives us a way to compare (some of the) tuples of values and constraints. In fact, when we have $a \leq_S b$, we will say that *b is better than a*. Thus, $\mathbf{0}$ is the worst value and $\mathbf{1}$ is the best one.

Given a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$, a finite set D (the domain of the variables), and an ordered set of variables V, a constraint is a pair $\langle def, con\rangle$ where $con \subseteq V$ and $def : D^{|con|} \rightarrow A$. Therefore, a constraint specifies a set of variables (the ones in con), and assigns to each tuple of values of D of these variables an element of the semiring set A. A soft constraint satisfaction problem (SCSP) is just a set of soft constraints over a set of variables.

Many known classes of satisfaction or oprimization problem can be cast in this formalism. A classical CSP is just an SCSP where the chosen c-semiring is: $S_{CSP} = \langle\{false, true\}, \vee, \wedge, false, true\rangle$. On the other hand, fuzzy CSPs [13,9] can be modeled in the SCSP framework by choosing the c-semiring: $S_{FCSP} = \langle[0, 1], max, min, 0, 1\rangle$. For weighted CSPs, the semiring is $S_{WCSP} = \langle\Re^+, min, +, +\infty, 0\rangle$. Here preferences are interpreted as costs from 0 to $+\infty$, which are combined with the sum and compared with min. Thus the optimization criterion is to minimize the sum of costs. For probabilistic CSPs [8], the semiring is $S_{PCSP} = \langle[0, 1], max, \times, 0, 1\rangle$. Here preferences are interpreted as probabilities ranging from 0 to 1, which are combined using the product and compared using max. Thus the aim is to maximize the joint probability.

Given an assignment t to all the variables of an SCSP, we can compute its preference value $pref(t)$ by combining the preferences associated by each constraint to the subtuples of the assignments referring to the variables of the constraint. More

precisely, $pref(P, s) = \Pi_{\langle idef, con\rangle \in C} def(s_{\downarrow con})$, where Π refers to the $\times$ operation of the semiring and $s_{\downarrow con}$ is the projection of tuple s on the variables in con.

For example, in fuzzy CSPs, the preference of a complete assignment is the minimum preference given by the constraints. In weighted constraints, it is instead the sum of the costs given by the constraints.

An optimal solution of an SCSP is then a complete assignment t such that there is no other complete assignment t'' with $pref(t) <_S pref(t'')$. The set of optimal solutions of an SCSP P will be written as $Opt(P)$.

3 Incomplete Soft Constraint Problems (ISCSPs)

Informally, an incomplete SCSP, written ISCSP, is an SCSP where the preferences of some tuples in the constraints, and/or of some values in the domains, are not specified. In detail, given a set of variables V with finite domain D, and c-semiring $S = \langle A, +, \times, 0, 1\rangle$ with a totally ordered carrier, we extend the SCSP framework to incompleteness by the following definitions.

Definition 1 (incomplete soft constraint). *Given a set of variables V with finite domain D, and a c-semiring $\langle A, +, \times, 0, 1\rangle$, an incomplete soft constraint is a pair $\langle idef, con\rangle$ where $con \subseteq V$ is the scope of the constraint and $idef : D^{|con|} \longrightarrow A \cup \{?\}$ is the preference function of the constraint. All tuples mapped into ? by $idef$ are called incomplete tuples.*

In an incomplete soft constraint, the preference function can either specify the preference value of a tuple by assigning a specific element from the carrier of the c-semiring, or leave such preference unspecified. Formally, in the latter case the associated value is ?. A soft constraint is a special case of an incomplete soft constraint where all the tuples have a specified preference.

Definition 2 (incomplete soft constraint problem (ISCSP)). *An incomplete soft constraint problem is a pair $\langle C, V, D\rangle$ where C is a set of incomplete soft constraints over the variables in V with domain D. Given an ISCSP P, we will denote with $IT(P)$ the set of all incomplete tuples in P.*

Definition 3 (completion). *Given an ISCSP P, a completion of P is an SCSP P' obtained from P by associating to each incomplete tuple in every constraint an element of the carrier of the c-semiring. A completion is partial if some preference remains unspecified. We will denote with $C(P)$ the set of all possible completions of P and with $PC(P)$ the set of all its partial completions.*

Example 1. A travel agency is planning Alice and Bob's honeymoon. The candidate destinations are the Maldive islands and the Caribbean, and they can decide to go by ship or by plane. To go to Maldives, they have a high preference to go by plane and a low preference to go by ship. For the Caribbean, they have a high preference to go by ship, and they don't give any preference on going there by plane.

Assume we use the fuzzy c-semiring $\langle [0, 1], max, min, 0, 1\rangle$. Then we can model this problem by using two variables T (standing for $Transport$) and D (standing for

$Destination$) with domains $D(T) = \{p, sh\}$ (p stands for $plane$ and sh for $ship$) and $D(D) = \{m, c\}$ (m stands for $Maldives$, c for $Caribbean$), and an incomplete soft constraint $\langle idef, con\rangle$ with $con = \{T, D\}$ and with preference function as shown in Figure 1. The only incomplete tuple in this soft constraint is (p, c).

Also, assume that for the considered season the Maldives are slightly preferrable to the Caribbean. Moreover, Alice and Bob have a high preference to plane as a way of transport, while they don't give any preference to ship. Moreover, as far as accommodations, which can be in a standard room, a suite, or a bungalow, assume that a suite in the Maldives is too expensive while a standard room in the Caribbean is not special enough for a honeymoon. To model this new information we use a variable A (standing for $Accommodation$) with domain $D(A) = \{r, su, b\}$ (r stands for $room$, su for $suite$ and b for $bungalow$), and three constraints: two unary incomplete soft constraints, $\langle idef1, \{T\}\rangle$, $\langle idef2, \{D\}\rangle$ and a binary incomplete soft constraint $\langle idef3, \{A, D\}\rangle$. The definition of such constraints is shown in Figure 1. The set of incomplete tuples of the entire problem is $IT(P) = \{(sh), (p, c), (su, c), (su, m), (r, m), (b, c)\}$.

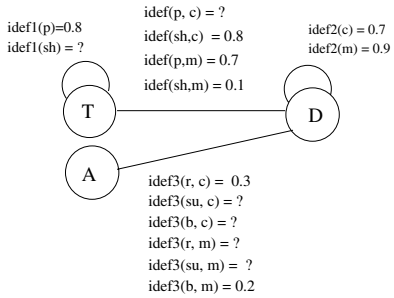


Fig. 1. An ISCSP

Definition 4 (preference of an assignment, and incomplete tuples). *Given an ISCSP $P = \langle C, V, D\rangle$ and an assignment s to all its variables we denote with $pref(P, s)$ the preference of s in P. In detail, $pref(P, s) = \Pi_{<idef,con>\in C|idef(s_{\downarrow con})\neq ?} idef(s_{\downarrow con})$. Moreover, we denote by $it(s)$ the set of all the projections of s over constraints of P which have an unspecified preference.*

The preference of an assignment s in an incomplete problem is thus obtained by combining the known preferences associated to the projections of the assignment, that is, of the appropriated subtuples in the constraints. The projections which have unspecified preferences, that is, those in $it(s)$, are simply ignored.

Example 2. Consider the two assignments $s_1 = (p, m, b)$ and $s_2 = (p, m, su)$, we have that $pref(P, s_1) = min(0.8, 0.7, 0.9, 0.2) = 0.2$, while $pref(P, s_2) = min($ $0.8, 0.7, 0.9) = 0.7$. However, while the preference of s_1 is fixed, since none of its projections is incomplete, the preference of s_2 may become lower that 0.7 depending on the preference of the incomplete tuple (su, m).

As shown by the example, the presence of incompleteness generates a partition of the set of assignments into two sets: those which have a certain preference which is independent of how incompleteness is resolved, and those whose preference is only an upperbound, in the sense that it can be lowered in some completions.

Given an ISCSP P, we will denote the first set of assignments as $Fixed(P)$ and the second with $Unfixed(P)$. In Example 2, $Fixed(P) = \{s_1\}$, while all other assignments belong to $Unfixed(P)$.

In SCSPs we have that an assignment is an optimal solution if its global preference is undominated. This notion can be generalized to the incomplete setting. In particular, when some preferences are unknown, we will speak of necessarily and possibly optimal solutions, that is, assignments which are undominated in all (resp., some) completions.

Definition 5 (necessarily and possibly optimal solution). *Given an ISCSP $P = \langle C, V, D\rangle$, an assignment $s \in D^{|V|}$ is a necessarily (resp, possibly) optimal solution iff $\forall Q \in C(P)$ (resp., $\exists Q \in C(P)$ such that) $\forall s' \in D^{|V|}$, $pref(Q, s') \not> pref(Q, s)$.*

Given an ISCSP P, we will denote with $NOS(P)$ (resp., $POS(P)$) the set of necessarily (resp., possibly) optimal solutions of P. Notice that, while $POS(P)$ is never empty, in general $NOS(P)$ may be empty. In particular, $NOS(P)$ is empty whenever the available preferences do not allow to determine the relation between an assignment and all the others.

Example 3. In the ISCSP P of Figure 1, we can easily see that $NOS(P) = \emptyset$ since, given any assignment, it is possible to construct a completion of P in which it is not an optimal solution. On the other hand, $POS(P)$ contains all assignments not including tuple (sh, m).

4 Characterizing POS(P) and NOS(P)

In this section we characterize the set of necessarily and possibly optimal solutions of an ISCSP given the preferences of the optimal solutions of two of the completions of P. In particular, given an ISCSP P defined on a totally ordered c-semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$, we consider:

- the SCSP $P_0 \in C(P)$, called the $\mathbf{0}$-completion of P, obtained from P by associating preference $\mathbf{0}$ to each tuple of $IT(P)$.
- the SCSP $P_1 \in C(P)$, called the $\mathbf{1}$-completion of P, obtained from P by associating preference $\mathbf{1}$ to each tuple of $IT(P)$.

Let us indicate respectively with $pref_0$ and $pref_1$ the preference of an optimal solution of P_0 and P_1. Due to the monotonicity of $\times$, and since $\mathbf{0} \leq \mathbf{1}$, we have that $pref_0 \leq pref_1$.

In the following theorem we will show that, if $pref_0 > \mathbf{0}$, there is a necessarily optimal solution of P iff $pref_0 = pref_1$, and in this case $NOS(P)$ coincides with the set of optimal solutions of P_0.

Theorem 1. *Given an ISCSP P and the two completions $P_0, P_1 \in C(P)$ as defined above, if $pref_0 > \mathbf{0}$ we have that $NOS(P) \neq \emptyset$ iff $pref_1 = pref_0$. Moreover, if $NOS(P) \neq \emptyset$, then $NOS(P) = Opt(P_0)$.*

Proof. Since we know that $pref_0 \leq pref_1$, if $pref_0 \neq pref_1$ then $pref_1 > pref_0$. We prove that, if $pref_1 > pref_0$, then $NOS(P) = \emptyset$. Let us consider any assignment s of P. Due to the monotonicity of $\times$, for all $P' \in C(P)$, we have $pref(P', s) \leq pref(P_1, s) \leq pref_1$.

- If $pref(P_1, s) < pref_1$, then s is not in $NOS(P)$ since P_1 is a completion of P where s is not optimal.
- If instead $pref(P_1, s) = pref_1$, then, since $pref_1 > pref_0$, we have $s \in Unfixed(P)$. Thus we can consider completion P_1' obtained from P_1 by associating preference $\mathbf{0}$ to the incomplete tuples of s. In P_1' the preference of s is $\mathbf{0}$ and the preference of an optimal solution of P_1' is, due to the monotonicity of $\times$, at least that of s in P_0, that is $pref_0 > \mathbf{0}$. Thus $s \notin NOS(P)$.

Next we consider when $pref_0 = pref_1$. Clearly $NOS(P) \subseteq Opt(P_0)$, since any assignment which is not optimal in P_0 is not in $NOS(P)$. We will show that $NOS(P) \neq \emptyset$ by showing that any $s \in Opt(P_0)$ is in $NOS(P)$. Let us assume, on the contrary, that there is $s \in Opt(P_0)$ such that $s \notin NOS(P)$. Thus there is a completion P' of P with an assignment s' with $pref(P', s') > pref(P', s)$. By construction of P_0, any assignment $s \in Opt(P_0)$ must be in $Fixed(P)$. In fact, if it had some incomplete tuple, its preference in P_0 would be $\mathbf{0}$, since $\mathbf{0}$ is the absorbing element of $\times$. Since $s \in Fixed(P)$, $pref(P', s) = pref(P_0, s) = pref_0$. By construction of P_1 and monotonicity of $\times$, we have $pref(P_1, s') \geq pref(P', s')$. Thus the contradiction $pref_1 \geq pref(P_1, s') \geq pref(P', s') > pref(P', s) = pref_0$. This allows us to conclude that $s \in NOS(P) = Opt(P_0)$. □

In the theorem above we have assumed that $pref_0 > \mathbf{0}$. The case in which $pref_0 = \mathbf{0}$ needs to be treated separately. We consider it in the following theorem.

Theorem 2. *Given ISCSP $P = \langle C, V, D \rangle$ and the two completions $P_0, P_1 \in C(P)$ as defined above, assume $pref_0 = \mathbf{0}$. Then, if $pref_1 = \mathbf{0}$, $NOS(P) = D^{|V|}$. Also, if $pref_1 > \mathbf{0}$, $NOS(P) = \{s \in Opt(P_1) | \forall s' \in D^{|V|}$ with $pref(P_1, s') > \mathbf{0}$ we have $it(s) \subseteq it(s')\}$.*

The formal proof is omitted for lack of space. Intuitively, if some assignment s' has an incomplete tuple which is not part of another assignment s, then we can make s' dominate s in a completion by setting all the incomplete tuples of s' to $\mathbf{1}$ and all the remaining incomplete tuples of s to $\mathbf{0}$. In such a completion s is not optimal. Thus s is not a necessarily optimal solution. However, if the tuples of s are a subset of the incomplete tuples of all other assignments then it is not possible to lower s without lowering all other tuples even further. This means that s is a necessarily optimal solution.

We now turn our attention to possible optimal solutions. Given a c-semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, it has been shown in [2] that idempotency and strict monotonicity of the $\times$ operator are incompatible, that is, at most one of these two properties can hold. In the following two theorems we show that the presence of one or the other of such two properties plays a key role in the characterization of $POS(P)$ where P is an ISCSP.

In particular, if $\times$ is idempotent, then the possibly optimal solutions are the assignments with preference in P between $pref_0$ and $pref_1$. If, instead, $\times$ is strictly monotonic, then the possibly optimal solutions have preference in P between $pref_0$ and

$pref_1$ and dominate all the assignments which have as set of incomplete tuples a subset of their incomplete tuples.

Theorem 3. *Given an ISCSP P defined on a c-semiring with idempotent $\times$ and the two completions $P_0, P_1 \in C(P)$ as defined above, if $pref_0 > \mathbf{0}$ we have that: $POS(P) = \{s \in D^{|V|} | pref_0 \leq pref(P,s) \leq pref_1\}$.*

The formal proof is omitted for lack of space. Informally, given a solution s such that $pref_0 \leq pref(P,s) \leq pref_1$, it can be shown that it is an optimal solution of the completion of P obtained by associating preference $pref(P,s)$ to all the incomplete tuples of s, and $\mathbf{0}$ to all other incomplete tuples of P. On the other hand, by construction of P_0 and due to the monotonicity of $\times$, any assignment which is not optimal in P_0 cannot be optimal in any other completion. Also, by construction of P_1, there is no assignment s with $pref(P,s) > pref_1$.

Theorem 4. *Given an ISCSP P defined on a c-semiring with a strictly monotonic $\times$ and the two completions $P_0, P_1 \in C(P)$ as defined above, if $pref_0 > \mathbf{0}$ we have that: an assignment $s \in POS(P)$ iff $pref_0 \leq pref(P,s) \leq pref_1$ and $pref(P,s) = max\{\, pref(P,s') |\, it(s') \subseteq it(s)\}$.*

The intuition behind the statement of this theorem is that, if assignment s is such that $pref_0 \leq pref(P,s) \leq pref_1$ and $pref(P,s) = max\{pref(P,s') | it(s') \subseteq it(s)\}$, then it is optimal in the completion obtained associating preference $\mathbf{1}$ to all the tuples in $it(s)$ and $\mathbf{0}$ to all the tuples in $IT(P) \setminus it(s)$. On the contrary, if $pref(P,s) < max\{pref(P,s') | it(s') \subseteq it(s)\}$, there must be another assignment s'' such that $pref(P, s'') = max\{pref(P,s') | it(s') \subseteq it(s)\}$. It can then be shown that, in all completions of P, s is dominated by s''.

In constrast to $NOS(P)$, when $pref_0 = \mathbf{0}$ we can immediately conclude that $POS(P) = D^{|V|}$, independently of the nature of $\times$, since all assignments are optimal in P_0.

Corollary 1. *Given an ISCSP $P = \langle C, V, D \rangle$, if $pref_0 = \mathbf{0}$, then $POS(P) = D^{|V|}$.*

The results given in this section can be summarized as follows:

- when $pref_0 = \mathbf{0}$
  - not enough information to compute $NOS(P)$ (by Theorem 2);
  - $POS(P) = D^{|V|}$ (by Corollary 1);
- when $pref_0 = pref_1 = \mathbf{0}$
  - $NOS(P) = D^{|V|}$ (by Theorem 2);
  - $POS(P) = D^{|V|}$ (by Corollary 1) ;
- when $\mathbf{0} = pref_0 < pref_1$
  - $NOS(P) = \{s \in Opt(P_1) | \forall s' \in D^{|V|}$ with $pref(P_1, s') > \mathbf{0}$ we have $it(s) \subseteq it(s')\}$ (by Theorem 2);
  - $POS(P) = D^{|V|}$ (by Corollary 1);
- when $\mathbf{0} < pref_0 = pref_1$
  - $NOS(P) = Opt(P_0)$ (by Theorem 1);
  - if $\times$ is idempotent: $POS(P) = \{s \in D^{|V|} | pref_0 \leq pref(P,s) \leq pref_1\}$ (by Theorem 3);

  - if $\times$ is strictly monotonic: $POS(P) = \{s \in D^{|V|} | pref_0 \leq pref(P,s) \leq pref_1, pref(P,s) = max\{\ pref(P,s') | it(s') \subseteq it(s)\}\}$ (by Theorem 4);
- when $\mathbf{0} < pref_0 < pref_1$
  - $NOS(P) = \emptyset$ (by Theorem 1);
  - $POS(P)$ as for the case when $\mathbf{0} < pref_0 = pref_1$.

5 A Solver for ISCSPs

We want to find a necessarily optimal solution of the given problem, if it exists. In most cases, however, the available information will only allow to determine the set of possibly optimal solutions. In such cases, preference elicitation is needed to discriminate among such assignments in order to determine a necessarily optimal one of the new problem with the elicited preferences. In this section we describe an algorithm, called *Find-NOS*, to achieve this task.

Algorithm 1. Find-NOS

```
Input: an ISCSP P
Output: an ISCSP Q, an assignment s, a preference p
P_0 ← P[?/0]
s_0, pref_0 ← BB(P_0, −)
s_1 ← s_0
pref_1 ← pref_0
s_max ← s_0
pref_max ← pref_0
repeat
    P_1 ← P[?/1]
    if pref_1 > pref_max then
        s_max ← s_1
        pref_max ← pref_1
    s_1, pref_1 ← BB(P_1, pref_max)
    if s_1 ≠ nil then
        S ← it(s_1)
        P ← Elicit(P, S)
        pref_1 ← pref(P, s_1)
until s_1 = nil ;
return P, s_max, pref_max
```

Algorithm *Find-NOS* takes in input an ISCSP P over a totally ordered c-semiring and returns an ISCSP Q which is a partial completion of P, and an assignment $s \in NOS(Q)$ together with its preference p. Given an ISCSP P, *Find-NOS* first checks if $NOS(P)$ is not empty, and, if so, it returns P, $s \in NOS(P)$, and its preference. If instead $NOS(P) = \emptyset$, it starts eliciting the preferences of some incomplete tuples.

In detail, *Find-NOS* first computes the $\mathbf{0}$-completion of P, written as $P[?/0]$, called P_0, and applies Branch and Bound (BB) to it. This allows to find an optimal solution of P_0, say s_0, and its preference $pref_0$.

In our notation, the application of the BB procedure has two parameters: the problem to which it is applied, and the starting bound. When BB is applied without a starting

bound, we will write $BB(P, -)$. When the BB has finished, it returns a solution and its preference. If no solution is found, we assume that the returned items are both nil.

Variables s_1 and $pref_1$ (resp., s_{max} and $pref_{max}$) represent the optimal solution and the corresponding preference of the $\mathbf{1}$-completion of the current problem (written $P[?/1]$) (resp., the best solution and the corresponding preference found so far). At the beginning, such variables are initialized to s_0 and $pref_0$.

The main loop of the algorithm, achieved through the **repeat** command, computes the $\mathbf{1}$-completion, denoted by P_1, of the current problem. In the first iteration the condition of the first **if** is not satisfied since $pref_1 = pref_{max} = pref_0$. The execution thus proceeds by applying BB to P_1 with bound $pref_{max} = pref_0 \geq \mathbf{0}$. This allows us to find an optimal solution of P_1 and its corresponding preference, assigned to s_1 and $pref_1$. If BB fails to find a solution, s_1 is nil. Thus the second **if** is not executed and the algorithm exits the loop and returns P, $s_{max} = s_0$, and $pref_{max} = pref_0$.

If instead BB applied to P_1 with bound $pref_{max}$ does not fail, then we have that $pref_0 < pref_1$. Now the algorithm elicits the preference of some incomplete tuples, via procedure $Elicit$.

This procedure takes an ISCSP and a set of tuples of variable assignments, and asks the user to provide the preference for such tuples, returning the updated ISCSP. The algorithm calls procedure $Elicit$ over the current problem P and the set of incomplete tuples of s_1 in P. After elicitation, the new preference of s_1 is computed and assigned to $pref_1$.

Since $s_1 \neq nil$, a new iteration begins, and BB is applied with initial bound given by the best preference between $pref_1$ and $pref_{max}$. Moreover, if $pref_1 > pref_{max}$, then s_{max} and $pref_{max}$ are updated to always contain the best solution and its preference. Iteration continues until the elicited preferences are enough to make BB fail to find a solution with a better preference w.r.t. the previous application of BB. At that point, the algorithm returns the current problem and the best solution found so far, together with its preference.

Theorem 5. *Given an ISCSP P in input, algorithm* Find-NOS *always terminates and returns an ISCSP Q such that $Q \in PC(P)$, an assignment $s \in NOS(Q)$, and its preference in Q.*

Proof. At each iteration, either $pref_{max}$ increases or, if it stays the same, a new solution will be found since after elicitation the preference of s_1 has not increased. Thus, either $pref_{max}$ is so high that BB doesn't find any solution, or all the optimal solutions have been considered. In both cases the algorithm exits the loop.

At the end of its execution, the algorithm returns the current partial completion of given problem and a solution s_{max} with the best preference seen so far $pref_{max}$. The **repeat** command is exited when $s_1 = nil$, that is, when $BB(P[?/1], pref_{max})$ fails. In this situation, $pref_{max}$ is the preference of an optimal solution of the $\mathbf{0}$-completion of the current problem P. Since BB fails on $P[?/1]$ with such a bound, by monotonicity of the $\times$ operator, $pref_{max}$ is also the preference of an optimal solution of $P[?/1]$. By Theorems 1 and 2, we can conclude that $NOS(P)$ is not empty. If $pref_{max} = \mathbf{0}$, then $NOS(P)$ contains all the assignments and thus also s_0. The algorithm correctly returns the same ISCSP given in input, assignment s_0 and its preference $pref_0 = \mathbf{0}$. If

instead $\mathbf{0} < pref_{max}$, again the algorithm is correct, since by Theorem 1 we know that $NOS(P) = Opt(P[?/0])$, and since $s_{max} \in Opt(P[?/0])$. □

Notice also that the algorithm performs preference elicitation only on solutions which are possibly optimal in the current partial completion of the given problem (and thus also in the given problem). In fact, by Theorems 3 and 4, any optimal solution of the $\mathbf{1}$-completion of the current partial completion Q is a possibly optimal solution of Q. Thus no useless work is done to elicit preferences related to solutions which cannot be necessarily optimal for any partial completion of the given problem. This also means that our algorithm works independently of the properties of the $\times$ operator.

6 Experimental Setting and Results

We have implemented Algorithm *Find-NOS* in Java and we have tested it on randomly generated ISCSPs with binary constraints and based on the Fuzzy c-semiring. To generate such problems, we use the following parameters:

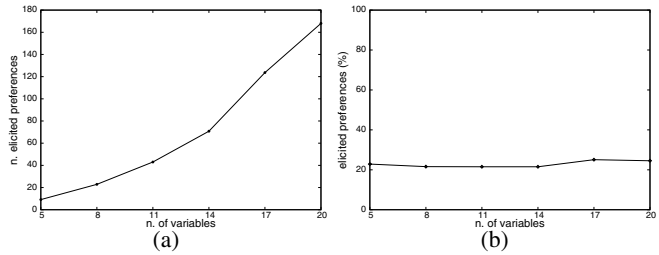
- n: number of variables;
- m: cardinality of the domain of each variable;
- d: density of the constraints, that is, the percentage of binary constraints present in the problem w.r.t. the total number of possible binary constraints that can be defined on n variables;
- t: tightness, that is, the percentage of tuples with preference 0 in each constraint w.r.t. the total number of tuples (m^2 since we have only binary constraints), and in each domain;
- i: incompleteness, that is, the percentage of incomplete tuples (formally, tuples with preference $?$) in each constraint and in each domain.

For example, if the generator is given in input $n = 10$, $m = 5$, $d = 50$, $t = 10$, and $i = 30$, it will generate a binary ISCSP with 10 variables, each with 5 elements in the domain, 22 constraints on a total of $45 = n(n-1)/2$, 2 tuples with preference 0 and 7 incomplete tuples over a total of 25 in each constraint, and 1 missing preference in each domain.

Notice that we use a model B generator: density and tightness are interpreted as percentages, and not as probabilities. Also, when we simulate the Elicit procedure, we randomly generate values in $(0, 1]$.

We have generated classes of ISCSPs by varying one parameter at a time, and fixing the other ones. The varying parameters are the number of variables, the density, and the incompleteness. When the number of variables varies (from $n = 5$ to $n = 20$, with step 3), we set $m = 5$, $d = 50$, $t = 10$, and $i = 30$. When we vary the density (from $d = 10$ to $d = 80$ with step 5), we set $n = 10$, $m = 5$, $t = 10$, and $i = 30$. Finally, when we vary the incompleteness (from $i = 10$ to $i = 80$ with step 5), we set $n = 10$, $m = 5$, $d = 50$, and $t = 10$.

In all the experiments, we have measured the number of tuples elicited by Algorithm *Find-NOS*. We also show the percentage of elicited tuples over the total number of incomplete tuples of the problem in input. For each fixed value of all the parameters, we show the average of the results obtained for 50 different problem instances, each



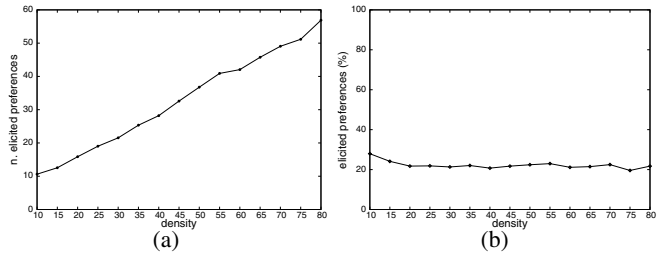
(a) (b)

Fig. 2. Number and percentage of elicited preferences, as a function of the number of variables. Fixed parameters: $m = 5$, $d = 50$, $t = 10$, $i = 30$.

given in input to *Find-NOS* 10 times. This setting is necessary since we have two kinds of randomness: the usual one in the generation phase and a specific one when eliciting preferences.

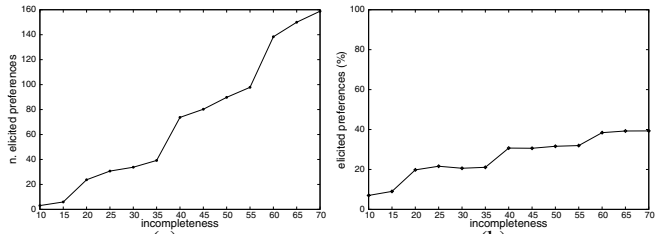
Figure 2 shows the absolute number and the percentage of elicited preferences when the number of variables varies. As expected, when the number of variables increases, the absolute number of elicited preferences increases as well, since there is a growth of the total number of incomplete tuples. However, if we consider the percentage of elicited tuples, we see that it is not affected by the increase in the number of variables. In particular, the percentage of elicited preferences remains stable around 22%, meaning that, regardless of the number of variables, the agent is asked to reveal only 22 preferences over 100 incomplete tuples. A necessarily optimal solution can be thus found leaving 88% of the missing preferences unrevealed.

Similar results are obtained when density varies (see Figure 3). We can see that the absolute number of elicited preferences grows when density increases. The maximum number of elicited preferences reached is however lower that the maximum reached when varying the variables (see Figure 2(a)). The reason for this is that the largest problems considered when varying the number of variables have more incomplete tuples than the largest obtained when varying the density. In fact, a problem with $n = 20$, given the fixed parameters, has around 685 incomplete tuples, 165 of which (about 22%) are elicited. On the other hand, a problem with $d = 80$, given the fixed parameters, has around 262 incomplete tuples, 55 (about 22%) of which are elicited. This is coherent with the fact that the results on the percentage of elicited preferences when varying the density and the number of variables are very similar.



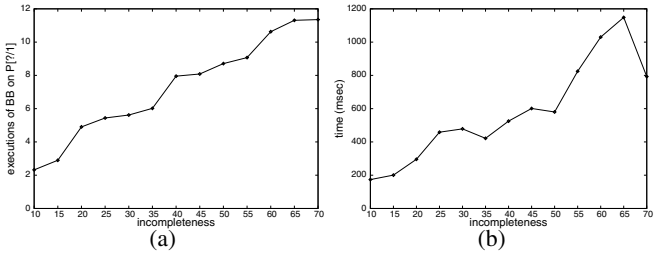
(a) (b)

Fig. 3. Number and percentage of elicited preferences, as a function of the density. Fixed parameters: $n = 10$, $m = 5$, $t = 10$, $i = 30$.



(a) (b)

Fig. 4. Number and percentage of elicited preferences, as a function of the incompleteness



(a) (b)

Fig. 5. CPU time and BB runs as a function of the incompleteness. Fixed parameters: $n = 10$, $m = 5$, $t = 10$, $i = 30$.

The last set of experiments vary the percentage of incompleteness (see Figure 4). As for density and number of variables, the absolute number of elicited preferences grows when the percentage of incompleteness increases. The maximum number of elicited preferences reached is close to that reached when varying the variables. However, the number of incomplete tuples of the problems with $i = 70$ is around 460 and thus smaller than that of problems with $n = 20$. Thus the percentage of elicited preferences is larger in problems with $i = 70$. This is confirmed by the corresponding result for the percentage of elicited preferences, which is shown to be around 35%. Additionally, the percentage of elicited preferences follows a slightly increasing trend as the percentage of incompleteness in the problem grows. However, it maintains itself below 35%, which means that in the worst case, where 70% of the tuples are incomplete, we are able to find a necessary optimal solution leaving 46% of the total number of tuples unspecified.

In Figure 6 we show the information about how many missing preferences we need to ask the user in a different way: each bar has a lower part showing the amount of information we have already (the one available at the beginning), while the higher part shows how much more information we need to ask the user for in order to find a necessarily optimal solution. It is possible to see that, when we have already some initial information, usually we need to ask the user for more even if the initial information amount is large. This is because some of the preferences available initially may be not useful for the computation of an optimal solution. On the other hand, if we start with no initial preferences (rightmost bar), we need to ask the user only for about 40 % of the preferences.

The focus of this work is on how many elicitation steps need to be done before finding a necessarily optimal solution. In fact, our Branch and Bound procedure could

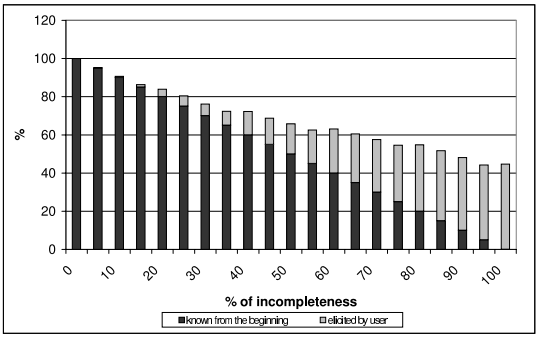


Fig. 6. Amount of preferences initially available and elicited as a function of the incompleteness. Fixed parameters: $n = 10$, $m = 5$, $t = 10$, $i = 30$.

certainly be improved in terms of efficiency. However, we show in Figure 5 the CPU time needed to solve some incomplete problems (when incompleteness varies) and also the number of runs of the Branch and Bound on the 1-completion.

7 Ongoing and Future Work

We are currently working on several variants of the algorithm described in this paper, where elicitation occurs not at the end of an entire BB search tree, but at the end of every complete branch or at every node. In this case, the algorithm runs BB only once before finding a necessarily optimal solution. Moreover, we are also considering variants of the Elicit function that asks for just one of the missing preferences: for example, in the context of fuzzy constraints, it just asks for the worst one, since it is the most useful one due to the drowning effect.

Future work will consider partially ordered preferences and also other ways to express preferences, such as qualitative ones a la CP nets, as well as other kinds of missing data, such as those considered in dynamic, interactive, and open CSPs. Moreover, other solving approaches can be considered, such as those based on local search and variable elimination.

We also would like to consider the specification of intervals, rather than a fixed preference or nothing. This would not change much our setting. We consider the [0,1] interval where no preference is specified, and thus the 0-completion and the 1-completion. With generic intervals, we can consider the ”lower bound completion” and the “upper bound completion”.

References

1. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint solving and optimization. Journal of the ACM 44(2), 201–236 (1997)
2. Bistarelli, S., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G., Fargier, H.: Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. Constraints 4(3) (1999)

3. Dechter, R.: Constraint processing. Morgan Kaufmann, San Francisco (2003)
4. Dechter, R., Dechter, A.: Belief maintenance in dynamic constraint networks. In: AAAI, pp. 37–42 (1988)
5. Faltings, B., Macho-Gonzalez, S.: Open constraint satisfaction. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 356–370. Springer, Heidelberg (2002)
6. Faltings, B., Macho-Gonzalez, S.: Open constraint optimization. In: Rossi, F. (ed.) CP 2003. LNCS, vol. 2833, pp. 303–317. Springer, Heidelberg (2003)
7. Faltings, B., Macho-Gonzalez, S.: Open constraint programming. Artif. Intell. 161(1-2), 181–208 (2005)
8. Fargier, H., Lang, J.: Uncertainty in constraint satisfaction problems: a probalistic approach. In: Moral, S., Kruse, R., Clarke, E. (eds.) ECSQARU 1993. LNCS, vol. 747, pp. 97–104. Springer, Heidelberg (1993)
9. Fargier, H., Schiex, T., Verfaille, G.: Valued Constraint Satisfaction Problems: Hard and Easy Problems. In: IJCAI-95, pp. 631–637. Morgan Kaufmann, San Francisco (1995)
10. González, S.M., Ansótegui, C., Meseguer, P.: On the relation among open, interactive and dynamic CSP. In: The Fifth Workshop on Modelling and Solving Problems with Constraints (IJCAI'05) (2005)
11. Lamma, E., Mello, P., Milano, M., Cucchiara, R., Gavanelli, M., Piccardi, M.: Constraint propagation and value acquisition: Why we should do it interactively. In: IJCAI, pp. 468–477 (1999)
12. Lang, J., Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Winner determination in sequential majority voting. In: IJCAI, pp. 1372–1377 (2007)
13. Ruttkay, Z.: Fuzzy constraint satisfaction. In: Proceedings 1st IEEE Conference on Evolutionary Computing, Orlando, pp. 542–547 (1994)