Mixed constraint satisfaction: a framework for decision problems under incomplete knowledge

Hélène Fargier, Jérôme Lang
IRIT - Université Paul Sabatier
31062 Toulouse Cedex (France)

Thomas Schiex
INRA
31320 Castanet Cedex (France)

Abstract

Constraint satisfaction is a powerful tool for representing and solving decision problems with complete knowledge about the world. We extend the CSP framework so as to represent decision problems under incomplete knowledge. The basis of the extension consists in a distinction between controllable and uncontrollable variables — hence the terminology "mixed CSP" — and a "solution" gives actually a conditional decision. We study the complexity of deciding the consistency of a mixed CSP. As the problem is generally intractable, we propose an algorithm for finding an approximate solution.

Introduction

Most of the time, solving a constraint satisfaction problem (CSP) means finding an assignment of the variables satisfying all the constraints, if such an assignment exists. Such a request often relies on the assumption (generally implicit) that the variables are *controllable*, *i.e.*, that the agent has the power to fix their values. However, one may think to give variables a status of *uncontrollable*: the value of an uncontrollable variable is fixed by the external world and thus it is outside the control of the agent. In the rest of the paper we will refer to controllable variables as *decision variables* and to uncontrollable as *parameters*. If all variables of a CSP are controllable, the purpose of the resolution of the CSP is to find a good decision which the agent may apply: we will refer to this kind of CSP (which is the most common in the CSP literature) as *decision-oriented CSP*. If all variables are uncontrollable, the constraints represent pieces of knowledge about the possible actual values of the variables. In this case, what is expected is not necessarily to find a consistent assignment, but rather to find the *actual value* of each variable. This means that the resolution process has to be interpreted not in terms of decision but in terms of *reasoning* (as this is the case in most logical approaches for knowledge representation). We will refer to this kind of CSP as *reasoning-oriented CSP*. Examples of reasoning-oriented CSPs in the literature are far less numerous. For instance, CSP used for scene interpretation (e.g. (Cooper 1992)) clearly belong in this class.

The fundamental difference between these two kinds of CSPs thus lies in the interpretation of the data (variables and constraints) and in the output of the resolution process; interpreting the number of solutions is particularly worth of interest: if a *decision-oriented* CSP has *several solutions*, this is rather good news since the agent will be (in principle) happy with any of them; on the contrary, if a *reasoning-oriented* CSP has several solutions, which means that the knowledge is incomplete, giving only one solution means arbitrary selecting one possible situation, which is not necessarily the real one – giving all solutions could be more adequate. Now, if a decision-oriented CSP has no solution, the problem is over-specified; inconsistency for reasoning-oriented CSP has to be interpreted in a different way, since the variables *do* have a value in the actual world – and therefore at least one of the constraints is not sound.

Now, beyond pure decision or reasoning-oriented problems, there are practical problems where a decision has to be taken, while the actual world is not completely known (which is the basic assumption in applications of decision theory). For instance, in scheduling, the decision (sequencing the tasks) may depend on ill-known parameters such as the reliability of some machines. Our motivation is to show that the CSP framework, which has been successfully developed and applied in the past years, can be extended in such a way that it can represent and solve problems of *decision making under incomplete knowledge*. These problems will be represented by so-called *mixed CSPs* involving both controllable and uncontrollable variables.

The definition of a *solution* of a mixed CSP depends crucially on the assumptions concerning the agent's awareness of the parameter values (the state of the world) at the time the decision must be made (deadline for acting) – where a *decision* is an assignment of decision variables. We will consider these two epistemological assumptions:
FO (*full observability*): the actual world will be completely revealed to the agent *before* the deadline is

reached (possibly, *just* before), so that it is useful to compute "off-line" a ready-to-use *conditional decision*, mapping possible cases to decisions, that the agent will be able to instantiate in real time, as soon as the actual world is known.

NO (*no observability*): the agent will never learn anything new about the actual state of the world before the deadline; all it will ever know is already encoded in the initial specification of the problem. In this case, it is useless to provide a conditional decision, and a suitable solution of the problem consists in pure, unconditional decision to be taken in any possible case.

The intermediate case where some partial knowledge about the state of the world may be learned (for instance through information-gathering actions) is not considered in this paper. The two extreme assumptions we consider lead to two definitions of a solution of a mixed CSP and two different notions of consistency, that we study in Section 2. In Section 3 we propose an anytime algorithm for solving mixed CSP under assumption **(FO)**. Possible extensions and applications are mentioned in conclusion.

Mixed constraint satisfaction

Definitions

A classical *constraint satisfaction problem* is a triple $P = \langle X, D, \mathcal{C}\rangle$, where X is a set of *variables*, each of which takes its possible values in a domain D_i (supposed here finite, we note $D = \times_i D_i$), and $\mathcal{C}$ is a a set of *constraints*, each of which restricting the possible values of some variables. $Sol(P)$ denotes the set of all assignments satisfying all constraints of P.

Roughly speaking, a mixed CSP is a CSP equipped with a partition between (controllable) decision variables and (uncontrollable) parameters.

Definition 1 (mixed CSP) *A* mixed CSP *is defined by a 6-uple* $\mathcal{P} = \langle \Lambda, L, X, D, \mathcal{K}, \mathcal{C}\rangle$ *where:*

- $\Lambda = \{\lambda_1, \ldots, \lambda_p\}$ *is a set of parameters;*
- $L = L_1 \times \cdots \times L_p$*, where* L_i *is the domain of* λ_i*;*
- $X = \{x_1, \ldots, x_n\}$ *is a set of decision variables;*
- $D = D_1 \times \cdots \times D_n$*, where* D_i *is the domain of* x_i*;*
- $\mathcal{K}$ *is a set of constraints involving only parameters;*
- $\mathcal{C}$ *is a set of constraints, each of them involving at least one decision variable.*

A complete assignment of the parameters will be called a *world* and will be denoted by ω. A complete assignment of the decision variables will be called a*decision* and will be generally denoted by ω (resp. by d). The complete assignment of both parameters and decision variables resulting from the concatenation of ω and d will be denoted by (ω, d).

The important distinction between the constraints of $\mathcal{K}$ and those of $\mathcal{C}$ is due to their very different interpretations: the constraints of $\mathcal{K}$ involve only parameters and thus represent the *knowledge* we have about the real world. Now, $\mathcal{C}$ contains *decision* constraints, which restrict the allowed values of decision variables, *possibly conditioned* by some parameters — if so they will be called *mixed* (or equivalently *parameterized*) constraints. If $\Lambda = \emptyset$ (resp. $X = \emptyset$), $\mathcal{P}$ is a classical decision-oriented (resp. reasoning-oriented) CSP.

Definition 2 *A solution of the reasoning-oriented CSP* $\langle \Lambda, L, \mathcal{K}\rangle$ *is a* **possible world**. *The set of all possible worlds for* $\mathcal{P}$ *is denoted by* $Poss(\mathcal{P})$.

If $\mathcal{K}$ were inconsistent, we would get $Poss(\mathcal{P}) = \emptyset$. But this means that at least one of the constraints of $\mathcal{K}$ is not sound. A non-trivial handling of this inconsistency being outside the scope of the paper, we assume that $\mathcal{K}$ is consistent.

A mixed CSP defines a collection $Candidates(\mathcal{P})$, of classical, decision-oriented CSPs, one for each possible world ω — it will be called the set of *candidate problems* induced by $\mathcal{P}$. Briefly (we omit technical details for space considerations), the candidate problem induced by a world ω is a CSP over X whose constraints are the original pure decision constraints plus those obtained by reducing each mixed constraint C to a pure decision constraint by letting its parameters take their value as specified in ω[1].

Now, a decision d is said to *cover a world* ω iff it satisfies the candidate problem induced by ω; this means that if the real world appears to be ω, then d will be a suitable decision. Among the possible worlds, those which are covered by at least one decision, *i.e.*, whose induced candidate problem is consistent, are called *good worlds* — and the others are called *bad worlds*.

Definition 3 *A decision* d *is said to cover a world* ω *iff* (ω, d) *is a solution of* $\langle \Lambda \cup X, L \times D, \mathcal{C}\rangle$. *The set of all worlds (possible or not) covered by* d *is denoted by* $Covers_{\mathcal{P}}(d)$.

Definition 4
$Good(\mathcal{P}) = Poss(\mathcal{P}) \cap (\cup_{d \in D} Covers_{\mathcal{P}}(d))$
$Bad(\mathcal{P}) = Poss(\mathcal{P}) \setminus Good(\mathcal{P})$

Equivalently,
$Good(\mathcal{P}) = \cup_{d\in D}\{\omega \in Poss(\mathcal{P}) | (\omega, d)$ satisfies $\mathcal{C}\}$.

Example: let $\mathcal{P}$ be the following mixed CSP:
- $X = \{x_1, x_2, x_3\}$; $D_1 = D_2 = D_3 = \{1, 2, 3\}$
- $\Lambda = \{\lambda_1, \lambda_2\}$; $L_1 = L_2 = \{a, b, c, d\}$
- $\mathcal{K}$ contains only one constraint: C_0: $\lambda_1 \neq \lambda_2$
- $\mathcal{C}$ contains three constraints C_1, C_2, C_3:

C_1

λ_1	x_1
a	1
a	2
b	1
b	3
c	2
d	3

C_2

λ_2	x_2	x_3
a	1	1/2/3
a	2	2/3
a	3	3
b	1	2/3
b	2	3
c	3	1/2/3
d	2	1/2/3

C_3

$x_1 + x_2 \leq 4$

[1] Note that among the CSPs of $Candidates(\mathcal{P})$, some may be inconsistent, which means that the actual problem may be inconsistent.

$Poss(\mathcal{P}) = \{(a,b),(a,c),(a,d),(b,a),(b,c),(b,d),(c,a),(c,b),(c,d),(d,a),(d,b),(d,c)\}$. Let $\omega_1 = (d,b)$ and $\omega_2 = (d,c)$; the decisions covering ω_1 are $(3,1,2)$ and $(3,1,3)$. It can be checked that no decision covers ω_2, and that all other possible worlds are good: $Bad(\mathcal{P}) = \{\omega_2\}$. Lastly, let $d_1 = (1,3,3)$, then $Covers_{\mathcal{P}}(d_1) = \{a,b\} \times \{a,c\}$.

Conditional and unconditional decisions as solutions

The case of full observability In the case of full observability, solving a mixed CSP consists in giving a conditional strategy (from a planning point of view, a conditional plan) associating, to each possible world, a decision that satisfies the corresponding candidate problem: once the actual values of the parameters are known, we have an appropriate decision. A weaker request consists in giving a conditional decision which is only defined on a subset of $Poss(\mathcal{P})$ (ideally on $Good(\mathcal{P})$).

Definition 5 (conditional decisions) *A* **conditional decision** *s for $\mathcal{P}$ is a map from a subset W_s of $Poss(\mathcal{P})$ to D such that $\forall \omega \in W_s$, $s(\omega)$ covers ω. If $W_s = Good(\mathcal{P})$, s is* **optimal**. *If $W_s == Good(\mathcal{P}) = Poss(\mathcal{P})$ (implying $Bad(\mathcal{P}) = \emptyset$), s is* **complete**.

Obviously, a complete conditional decision is also optimal, but the converse is generally not true: while a mixed CSP always has optimal conditional decisions, it does not always have a complete conditional decision.

Definition 6 (consistency) *A mixed CSP is* **consistent** *if it has a complete conditional decision.*

Clearly, $\mathcal{P}$ is consistent $\Leftrightarrow Bad(\mathcal{P}) = \emptyset \Leftrightarrow Good(\mathcal{P}) = Poss(\mathcal{P})$. The consistency of $\mathcal{P}$ means that all candidate problems are consistent. When it is not the case, a weaker request consists in giving an optimal conditional decision, which covers all situations which are possible to cover.

Example (continued): the following s is an optimal conditional decision for $\mathcal{P}$[2]:

$\lambda_1 \downarrow$ | $\lambda_2 \rightarrow$	a	b	c	d
a	*imposs.*	$(1,2,3)$	$(1,3,3)$	$(1,2,3)$
b	$(1,2,3)$	*imposs.*	$(1,3,3)$	$(1,2,3)$
c	$(2,2,3)$	$(2,2,3)$	*imposs.*	$(2,2,3)$
d	$(3,1,2)$	$(3,1,2)$	*bad*	*imposs.*

The case of no observability The previous approach assumes that the actual world will be known before the decision has to be taken (**FO**). Clearly, under the other extreme assumption (**NO**), it is useless to look for a conditional decision: consequently, in this case one has to look for a *pure (unconditional) decision*. Intuitively, this pure decision should cover "as many worlds as possible"; if there is no other decision covering a strict subset of the worlds covered by d, d is *undominated*; if furthermore d covers all good (resp. possible) worlds, it is *universal* (resp *strongly universal*).

[2] The table has to be read this way: a world ω is defined by the values assigned to both parameters λ_1 and λ_2 and corresponds thus to a row and a column, at the intersection of which is the decision $s(\omega)$ when defined *i.e.*, when $\omega \in W(s)$. Here, $W(s)$ contains all good worlds, so s is optimal.

Definition 7 *A decision $d \in D$ is a* **undominated** *for $\mathcal{P}$ iff there is no $d' \in D$ such that $Covers_{\mathcal{P}}(d')$ strictly contains $Covers_{\mathcal{P}}(d)$; it is an* **universal decision** *for $\mathcal{P}$ if $Covers_{\mathcal{P}}(d) = Good(\mathcal{P})$ and a* **strong universal decision** *for $\mathcal{P}$ if $Covers_{\mathcal{P}}(d) = Poss(\mathcal{P})$.*

While a mixed CSP always has an undominated decision (and generally has several incomparable ones), it is not always the case that it has a universal decision (and *a fortiori* a strong universal decision). Note that a pure decision is an extreme case of a conditional decision, where all considered worlds are mapped to the same decision. Here is a summary of the relations between different notions of decisions:

strong universal decisions
$\subseteq$ universal decisions
$\subseteq$ undominated decisions
$\subseteq$ conditional decisions

and

strong universal decisions
$\subseteq$ complete conditional decisions
$\subseteq$ optimal conditional decisions
$\subseteq$ conditional decisions

Definition 8 (uniform & strong consistency) *A mixed CSP is* **uniformly consistent** *iff it has a universal decision, and* **strongly consistent** *iff it has a strong universal decision.*

Clearly, $\mathcal{P}$ is stronlyg consistent if and only if it is both consistent and uniformly consistent. The converses are generally not true. Note that universal (resp. strong universal) consistency implies that any undominated decision is a universal decision (resp. a strong universal decision).

Example (continued): $\mathcal{P}$ has no universal decision. Consider the mixed CSP $\mathcal{P}'$ obtained from $\mathcal{P}$ by replacing $C_3 : x_1 + x_2 \leq 4$ by $C_3' : 2x_1 + x_2 \leq 4$. Then, $Good(\mathcal{P}') = \{a,b\} \times \{a,b,d\}$ and $d = (1,2,3)$ is a universal decision (but not a strong universal decision).

Complexity Interestingly, the problems of deciding consistency and strong consistency fall in complexity classes which are *above* NP in the polynomial hierarchy. We recall (Stockmeyer 1977) that a decision problem is in $\mathsf{NP}^{\mathsf{NP}} = \Sigma_2^P$ iff it can be solved in polynomial time on a nondeterministic Turing machine using NP-oracles; a decision problem is in co-$\Sigma_2^P = \Pi_2^P$ iff its complementary problem is in Σ_2^P. The role of the "canonical" complete problem (which is played by SAT for NP) is played by 2-QBF for Σ_2^P; 2-QBF is the problem of deciding whether the following quantified boolean formula is true: $\exists \vec{a} \forall \vec{b} F(\vec{a},\vec{b})$. The complementary problem $\overline{\text{2-QBF}}$ ($\forall \vec{a} \exists \vec{b} F(\vec{a},\vec{b})$) is complete for Π_2^P[3]. Strong consistency (resp. consistency) looks sim-

[3] and remains so when F is restricted to 3-CNF

ilar to 2-QBF (resp. to $\overline{\text{2-QBF}}$) since it involves the same alternation of quantifiers.

Proposition 1 *The problem* MIXED-SAT *of deciding consistency of a binary mixed CSP is* Π_2^P*-complete.*

Sketch of proof: the problem is in Π_2^P since its complementary problem can be solved by the following algorithm: guess a world ω, check that it is possible and that the corresponding candidate problem is inconsistent (co-NP-complete oracle). The completeness can be proven by reducing $\overline{\text{2-QBF}}$ restricted to 3-CNF to the MIXED-SAT problem (see (Fargier *et al.* 1996).

Note that MIXED-SAT remains Π_2^P-complete even when we restrict ourself to binary mixed CSP with an empty $\mathcal{K}$. This results furthermore offers a "canonical" Π_2^P-complete problem to the CSP community. The problem of deciding strong consistency is easily shown to be in Σ_2^P, but completeness is still to be proven in the general case.

In practice, the complexity of MIXED-SAT may be reasonable if we consider that $\mathcal{K}$ should usually drastically reduce the number of possible worlds. Lastly, when $\mathcal{K} = \emptyset$ (or equivalently $Poss(\mathcal{P}) = L$), parameters are independent (since their possible values are not constrained). This assumption has a drastic effect on the complexity of strong consistency:

Proposition 2 *Under the parameter independence assumption, deciding strong consistency in a binary mixed CSP is* NP*-complete.*

Sketch of proof: membership in NP is straightforward (the decision d is a polynomial length certificate which is checkable in polynomial time since $Poss(\mathcal{P}) = L$). The completeness comes from the fact that the consistency of any binary CSP can be reduced to the strong consistency of the same CSP, considered as a mixed CSP with an empty parameter set Λ.

Searching for a conditional decision

The high complexity of finding a solution of a mixed CSP leads us to look for an algorithm which gives an optimal conditional solution if we let it run until it stops naturally, or otherwise gives an approximate (non optimal) conditional decision (the longer we let it run, the closer it gets to optimality). Intuitively, the algorithm incrementally builds a list of decisions which eventually covers a superset of all good worlds. Repeatedly, we pick a new decision d (to be added to the list) that covers at least one possible world among those which are not yet covered, we compute the set R of worlds (possible or not) that this decision covers and we subtract this set from the set of worlds which haven't yet been covered. In order to easily compute worlds covered by d, we assume that any constraint of $\mathcal{C}$ involves *at most one parameter*. Therefore, $Covers_{\mathcal{P}}(d)$ forms a Cartesian product of subsets of the parameter domains and can be computed in polynomial time in binary mixed CSP. The successive subtractions of these Cartesian products is performed using a technique recently proposed in (Freuder and Hubbe 1995) , called *subdomain subproblem extraction.* We recall this technique before we describe our algorithm.

Sub-domain extraction

We define an *environment* E as a set of worlds of the form $l_1 \times \cdots \times l_p$, with $\forall k, l_k \subseteq L_k$. An example of environment is $(\lambda_1 \in \{a,c\}, \lambda_2 \in \{b,c,d\})$, also written $\{a,c\} \times \{b,c,d\}$ or $[{}^{ac}_{bcd}]$ when there is no ambiguity on the order of parameters. The set of all possible environments is obviously a lattice (equipped with the inclusion order), whose top is the set of all possible parameter assignments (in the example, $[{}^{abcd}_{abcd}]$) and whose bottom is the empty set.

Given two environments E and R, *sub-domain subproblem extraction* technique (Freuder and Hubbe 1995) decomposes E into a set of disjoint sub-environments $Dec(E,R)$ such that all worlds of E belong either to R or to one of the sub-environments of the decomposition. The decomposition is unique if an ordering on the variables is fixed. When $E \cap R = \emptyset$ we let $Dec(E,R) = \{E\}$; and when $E \subseteq R, Dec(E,R) = \emptyset$. Thus, this decomposition is actually a *subtraction* since we end up with a list of disjoint environments which cover exactly the worlds of $E \setminus R$. Namely:

Proposition 3 ((Freuder and Hubbe 1995)) *$\omega \in E$ and $\omega \notin R \Leftrightarrow$ there exists a unique $F \in Dec(E,R)$ such that $\omega \in F$*

The algorithm

Start
`LD` := $\emptyset$; {list of decisions with the env. covered}
`Env` := $\{L_1 \times \cdots \times L_p\}$; {list of env. not yet covered by `LD`}
`Bad` := $\emptyset$; {list of non coverable env.}
Repeat
  E := an environment from `Env`; {possible heuristics}
  If $\langle \Lambda, L, \mathcal{K} \cup E \rangle$ is consistent (1)

  {E contains possible worlds}
  then
    If $\langle \Lambda \cup X, L \times D, \mathcal{K} \cup \mathcal{C} \cup E \rangle$ is inconsistent (2)
    {E contains only non-coverable worlds}
    then `Bad` := `Bad` $\cup E$
    else s := a solution of $\langle \Lambda \cup X, L \times D, \mathcal{K} \cup \mathcal{C} \cup E \rangle$;
      d := projection of s on the decision variables;
      {d covers at least one possible world of E}
      $R := Covers_{\mathcal{P}}(d)$ {unary constraints on param.}
      Add $\langle R, d \rangle$ to `LD`;
      `Env` := $\cup_{F \in \text{Env}} Dec(F,R)$ (3)
    end {else}
  end {else}
Until `Env` $= \emptyset$ (or interruption by the user)
End {`LD` covers $Good(P)$}

Several steps of the algorithm deserve some comments:

(1) The test of consistency of $\langle \Lambda, L, \mathcal{K} \cup E \rangle$ may seem superfluous — since its role is only to distinguish bad environments from impossible environments, and in

any case no new decision will be added — but is necessary[4] if we care about about the consistency of $\mathcal{P}$ (since Lemma 4 will not hold any longer if this test is removed).

(1)+(2) The two sequences of CSP $\langle \Lambda, L, \mathcal{K} \cup E\rangle$ and $\langle \Lambda \cup X, L \times D, \mathcal{K} \cup \mathcal{C} \cup E\rangle$, repeatedly solved by the algorithm, define *dynamic* CSP (Dechter ....) and their resolution may be improved by any techniques developed to solve such problems.

(3) Once a new R is built, it is subtracted from all the awaiting environments to avoid some redundant computations. This furthermore guarantees that the computation will stop when the current list LD defines a solution.

(3) The update of the list of environments Env, may be performed *lazily* to avoid memory consumption.

Lastly, the algorithm considers an environment as impossible only when *all* of its worlds are impossible; consequently, there will be environments in LD and Bad containing impossible worlds. This is not a problem, since (i) these impossible worlds will never be observed; the important point is that any possible, coverable world is covered by a decision, and any possible, bad world is in the Bad list; and (ii) any environment added to the Bad list contains *at least* a possible world, which ensures that at the end of the algorithm, $\mathcal{P}$ is consistent *iff* Bad $= \emptyset$.

Running the algorithm on the example

1. $\texttt{Env} = \{[{}^{abcd}_{abcd}]\}$; $\texttt{LD} = \emptyset$; $\texttt{Bad} = \emptyset$

2. $E = [{}^{abcd}_{abcd}]; d_1 = (3,1,2);\ R = \{b,d\} \times \{a,b\};$
$Dec([{}^{abcd}_{abcd}],[{}^{bd}_{ab}]) = \{[{}^{bd}_{cd}],[{}^{ac}_{abcd}]\};$
$\texttt{Env} = \{[{}^{bd}_{cd}],[{}^{ac}_{abcd}]\}$; $\texttt{LD} = \{\langle[{}^{bd}_{ab}],(3,1,2)\rangle\}$;

3. $E = [{}^{bd}_{cd}]; d_2 = (1,2,3); R = \{a,b\} \times \{a,b,d\};$
$Dec([{}^{bd}_{abd}],[{}^{ab}_{abd}]) = \{[{}^{b}_{c}],[{}^{d}_{cd}]\};$
$Dec([{}^{ac}_{abcd}],[{}^{ab}_{abd}]) = \{[{}^{a}_{c}],[{}^{c}_{abcd}]\};$
$\texttt{Env} = \{[{}^{b}_{c}],[{}^{d}_{cd}],[{}^{a}_{c}],[{}^{c}_{abcd}]\};$
$\texttt{LD} = \{\langle[{}^{bd}_{ab}],(3,1,2)\rangle,\langle[{}^{ab}_{abd}],(1,2,3)\rangle\};$

4. $E = [{}^{b}_{c}]; d_3 = (1,3,3); R = \{a,b\} \times \{a,c\};$
$Dec([{}^{b}_{c}],[{}^{ab}_{ac}]) = Dec([{}^{a}_{c}],[{}^{ab}_{ac}]) = \emptyset;$
$Dec([{}^{d}_{cd}],[{}^{ab}_{ac}]) = \{[{}^{d}_{cd}]\};\ Dec([{}^{c}_{abcd}],[{}^{ab}_{ac}]) = \{[{}^{c}_{abcd}]\};$
$\texttt{Env} = \{[{}^{d}_{cd}],[{}^{c}_{abcd}]\};$
$\texttt{LD} = \{\langle[{}^{bd}_{ab}],(3,1,2)\rangle,\langle[{}^{ab}_{abd}],(1,2,3)\rangle,\langle[{}^{ab}_{ac}],(1,3,3)\rangle\}$

5. $E = [{}^{d}_{cd}]$; $\mathcal{K} \cup \mathcal{C} \cup E$ inconsistent (but $\mathcal{K} \cup E$ consistent).
$\texttt{Bad} = \{[{}^{d}_{cd}]\}$

6. $E = [{}^{c}_{abcd}]; d_4 = (2,2,3); R = \{a,c\} \times \{a,b,d\};$
$Dec([{}^{c}_{abcd}],[{}^{ac}_{abd}]) = \{[{}^{c}_{c}]\}$; $\texttt{Env} = \{[{}^{c}_{c}]\}$;
$\texttt{LD} = \{\langle[{}^{bd}_{ab}],(3,1,2)\rangle,\langle[{}^{ab}_{abd}],(1,2,3)\rangle,\langle[{}^{ab}_{ac}],(1,3,3)\rangle,$
$\langle[{}^{ac}_{abd}],(2,2,3)\rangle\}$

7. $E = [{}^{c}_{c}]$; $\mathcal{K} \cup E$ inconsistent. $\texttt{Env} = \emptyset$. STOP

Finally, the last value of LD is $\{\langle[{}^{bd}_{ab}],(3,1,2)\rangle,$ $\langle[{}^{ab}_{abd}],(1,2,3)\rangle,\langle[{}^{ab}_{ac}],(1,3,3)\rangle,\langle[{}^{ac}_{abd}],(2,2,3)\rangle\}$, and $\texttt{Bad} = \{[{}^{d}_{cd}]\}$. This allows us to build a conditional decision where each world ω is mapped to the decision $s(\omega) = d_j$ associated to the first item $\langle E, d_j\rangle$ such that E contains ω (shown on the table below). This search of a decision for a given world could be made more efficient via the use of a discrimination tree for example. The worlds marked with a "*" are actually impossible but the list LD assigns them a decision, or labels them bad.

$\lambda_1 \downarrow \mid \lambda_2 \rightarrow$	a	b	c	d
a	$(1,2,3)^*$	$(1,2,3)$	$(1,3,3)$	$(1,2,3)$
b	$(3,1,2)$	$(3,1,2)^*$	$(1,3,3)$	$(1,2,3)$
c	$(2,2,3)$	$(2,2,3)$	*impossible*	$(2,2,3)$
d	$(3,1,2)$	$(3,1,2)$	*bad*	*bad**

Correctness of the algorithm

It is based on the following lemmas; their proof (omitted for length considerations) is in (Fargier *et al.* 1996).

Proposition 4 *At any point of the algorithm, if* $E \in$ Bad *then* $E \cap Poss(\mathcal{P}) \subseteq Bad(\mathcal{P})$ *and* $E \cap Poss(\mathcal{P}) \neq \emptyset$.

Proposition 5 *At any point of the algorithm, if* $\langle E, d\rangle \in$ LD *then* $\forall \omega \in E, (\omega, d)$ *satisfies* $\mathcal{C}$.

Proposition 6 (Correctness of the algorithm) *If run quietly, the algorithm stops, the final value of* LD *defines an optimal conditional decision for* $\mathcal{P}$ *and the final value of* Bad *contains* $Bad(\mathcal{P})$.

Sketch of proof: to prove the termination, we prove that $\cup_{E \in \mathrm{Env}} E$ decreases strictly at each iteration (using Proposition 3). The second part of the theorem relies on the following invariant, verified at any iteration: $\forall \omega \in Poss(\mathcal{P})$, $\exists \langle E, d\rangle \in$ LD such that $\omega \in E$ or $\exists! E \in$ Env such that $\omega \in E$ or $\exists! E \in$ Bad such that $\omega \in E$. The invariant is proved by induction on the algorithm (we omit the proof for length considerations). Applying the invariant to the end of the last iteration gives $\forall \omega \in Poss(\mathcal{P})$, either $\exists \langle E, d\rangle$ in LD such that $\omega \in E$, or $\exists E \in$ Bad such that $\omega \in E$. Together with Propositions 4 and 5, this achieves proving the theorem.

Since the set of possible worlds covered by LD grows monotonically as the algorithm runs (and so is Bad) it is possible to use the algorithm as an "anytime" algorithm. Actually, LD tends to a cover of $Good(\mathcal{P})$ and Bad tends to a subset of $Bad(\mathcal{P})$.

Proposition 7 *If the algorithm runs until it stops,* $\mathcal{P}$ *is consistent iff* Bad $\neq \emptyset$.

The proof follows easily from Propositions 6 and 4. Obviously, if Bad $= \emptyset$ and $|\texttt{LD}| = 1$ then $\mathcal{P}$ is strongly consistent. The converse is not true; more generally, the list LD may not be minimal[5].

[4] However, this test can be performed during the resolution (2) of $\langle \Lambda \cup X, L \times D, \mathcal{K} \cup \mathcal{C} \cup E\rangle$ by the backtrack-based algorithm simply by imposing that the parameters be instantiated first. As soon as a consistent labeling of the Λ is found, any vertical ordering heuristics may apply.

[5] For instance, it may be the case that the last decision added to the list covers all possible worlds but that many

Related work and conclusion

Partial CSP (Freuder 1989) also handles simultaneously a family of CSPs (representing the possible relaxations of an overconstrainted CSP) but the original motivation of this framework is not to handle decision under incomplete knowledge, and therefore lacks the distinction between controllable and uncontrollable variables. In (Helzerman and Harper, 1994) an efficient algorithm is proposed for enforcing arc-consistency simultaneously in a family of CSPs sharing some variables and constraints; such a family is related to our set of candidates, but again, this approach does not enable decision under incomplete knowledge. Now, in the field of knowledge representation (Boutilier 1994) proposed a logical, qualitative basis for decision theory, distinguishing like us between controllable and uncontrollable propositions, but without the notion of conditional decision. Poole's *independent choice logic* (Poole 1995) also distinguishes between the nature's choice space and the agent's choice space, in order to build strategies which are similar to our conditional decisions; it also includes the representation of probabilities, utilities and knowledge-gathering actions.

In this paper we proposed an anytime algorithm for computing conditional decisions under the full observability assumption. Further work will consist first in implementing and testing our algorithm; for this we think of expoiting results from the area of *dynamic constraint satisfaction* (Dechter ....), where several techniques are proposed in order to solve a sequence of CSP that differs only in some constraints more efficiently than by naively solving each CSP one after the other. Furthermore, the computation of a compact representation of conditional decisions could also probably be performed using Finite Automata (as (Vempaty 1992) for sets of solutions).

Since our contribution consists in extending the CSP framework in order to deal with decision problems under incomplete knowledge (namely by distinguishing between controllable and uncontrollable variables), it can be considered as a first step to embed decision theory into constraint satisfaction; other important steps in this direction would consist in considering *probabilities*, *utilities* and *sequences of decisions*. A first step towards handling probabilistic knowledge in constraint satisfaction has been proposed in (Fargier *et al.* 1995), where the uncertainty on the values of parameters is encoded by a given probability distribution instead of a set of constraints. Utility functions would enable us to represent *flexible goals* and it should not be hard to embed them in our framework; in the constraints of $\mathcal{C}$, an extra field is added to each tuple, namely the utility of the corresponding assignment, as in (Schiex *et al.*, 1995), (Bistarelli *et al.*, 1995). Extending our framework to sequences of decisions is significantly harder. The partition of the variables is not sufficient: not only decisions are influenced by parameters, but the value of some parameters may also be influenced by earlier decisions; for this we may structure decision variables and parameters in a directed network, in the same spirit as influence diagrams (Howard and Matheson, 1984)[6] , a link from λ_i to x_j meaning that the allowed values of the decision variable x_j depend on λ_i, and a link from x_i to λ_j meaning that the decision of assigning a value to x_j has or may have some effects on λ_j.

An interesting potential application of mixed constraint satisfaction and its possible extensions is *planning under uncertainty*. Clearly, this topic needs the notion of controllability, and would gain a lot in being encoded in an extension of the CSP framework — since it could benefit from the numerous advances on the resolution of CSPs. The similarity between our conditional decisions and conditional plans is clear, at least for a single action. For handling sequences of actions, mixed CSPs have to be extended as evoked in the previous paragraph. Non-deterministic effects of actions may be encoded by using extra parameters. Lastly, many recent approaches to planning make use of decision theory; clearly, extending mixed CSP with a larger part of decision theory goes in this direction; our long-term goal is thus to provide a constraint satisfaction based framework to decision-theoretic planning.

decisions have been added to the list before. Of course, one could think of checking afterwards whether some decisions are redundant, but this could be very costly.

References

Craig Boutilier, Toward a logic for qualitative decision theory, *Proc. of KR'94*, 75–86.

Martin Cooper, *Visual occclusion and interpretation of ambiguous pictures*, Ellis Horwood, 1992.

Rina Dechter, ...

Hélène Fargier, Jérôme Lang and Thomas Schiex, Constraint satisfaction and decision under uncertainty, Tech. Report IRIT-96-06, Université Paul Sabatier (Toulouse, France), Feb. 1996.

Hélène Fargier, Jérôme Lang, Roger Martin-Clouaire and Thomas Schiex, A constraint satisfaction framework for decision under uncertainty, *Proc. of Uncertainty in AI'95*, 167-174.

Eugene Freuder, Partial constraint satisfaction, *Proceedings of IJCAI'89*, 278–283.

Eugene Freuder and Paul Hubbe, Extracting constraint satisfaction subproblems, *Proceedings of IJCAI'95*, Montreal.

Randall A. Helzerman and Mary P. Harper, An approach to multiply segmented constraint satisfaction problems, *Proc. AAAI'94*, 350–355.

R.A. Howard and J.E. Matheson, Influence Diagrams, Influence Diagrams, in R.A. Howard and J.E. Matheson,

[6]Note that the multi-stage extension of mixed CSP would differ from influence diagrams in the representational and computational basis of our approach, which is, namely, constraint satisfaction; in particular, certain kinds of cycles are allowed in our approach, contrarily to the acyclicity requirement for infleuce diagrams.

eds., *The Principles and Applications of Decision Analysis*, vol.2 (1984), 720-761.

Alan K. Mackworth, Consistency in networks of relations, *Artificial Intelligence*, 8, 1977, 99–118.

David Poole, Exploiting the rule structure for decision making within the independent choice logic, *Proc. of Uncertainty in AI'95*, 454-463.

S. Bistarelli, F. Rossi and U. Montanari, Constraint solving over Semi-rings, *Proceedings of IJCAI'95*, Montreal.

T. Schiex, H. Fargier and G. Verfaillie, Valued Constraint Satisfaction Problems: hard and easy problems, *Proceedings of IJCAI'95*, Montreal.

L. J. Stockmeyer, The polynomial-time hierarchy, *Theoretical Computer Science*, 3, 1977, 1–22.

N. Rao Vempaty, *Solving Constraint Satisfaction Problems Using Finite State Automata*, Proceedings of AAAI'92, 453–458.