

# Conditional and Composite Constraints with Preferences

Malek Mouhoub and Amrudee Sukpan

University of Regina

Wascana Parkway, Regina, SK, Canada, S4S 0A2

{mouhoubm,sukpan1a}@cs.uregina.ca

## Abstract

Preferences in constraint problems are common but significant in many real world applications. In this paper, we extend our conditional and composite CSP (CCCSP) framework, managing CSPs in a dynamic environment, in order to handle preferences. Unlike the existing CSP models managing one form of preferences, ours supports four types, namely: *variable value* and *constraint preferences*, *composite preferences* and *conditional preferences*. This offers more expressive power in representing a wide variety of constraint problems. The preferences are considered here as a set of soft constraints using a *c-semiring* structure with combination and projection operators. Solving constraint problems with preferences consists of finding a solution satisfying all the constraints while optimizing the preference values. This is handled by a variant of the branch and bound algorithm, we propose in this paper, and where constraint propagation is used to improve the time efficiency. Experimental tests, we conducted on randomly generated CCCSPs with preferences, favor the MAC principle as the constraint propagation strategy to be used within the branch and bound procedure.

**Keywords.** Constraint Satisfaction, Local Search, Soft Constraints, Preferences.

## Introduction

Preferences play an important role in many real world constraint applications. Obviously, preferences are not hard constraints that have to be fully satisfied, but have an effect on choosing a good or the best solution satisfying all the hard constraints. Moreover, often preferences are implicit. Furthermore, these preference functions are often combined with other forms of preferences in order to have a global preference for a given consistent scenario.

In (Mouhoub & Sukpan 2007) we have proposed a modeling framework that allows the management of conditional and composite CSPs (CCCSPs) within a unique constraint network. This model enables the addition of variables and their related constraints dynamically to the problem to solve, during the resolution process, via composite variables and activity constraints. Composite variables are variables whose possible values are CSP variables<sup>1</sup>. In other words

this allows us to represent disjunctive CSP variables. An activity constraint has the following form  $X_1 \wedge \dots \wedge X_p \xrightarrow{\text{condition}} Y$  where  $X_1, \dots, X_p$  and  $Y$  are variables (composite or CSP variables). This activity constraint will activate  $Y$  ( $Y$  will be added to the problem to solve) if  $X_1 \wedge \dots \wedge X_p$  are active (currently present in the problem to solve) and *condition* holds between these variables. We call Conditional and Composite Constraint Satisfaction Problem (CCCSP) this model we have proposed.

In this paper, the CCCSP is extended to include four types of preferences: *variable value*, *constraint*, *composite* and *conditional preferences*. We call this model CCCSP with Preferences (or CCCSPP). *variable value* and *constraint preferences* associate degrees of preferences respectively to variables domain values and constraints<sup>2</sup>, in order to favor some decisions. A *composite preference* is a higher level of preference among the choices of a composite variable. *Conditional preferences* allow some preference functions (variable value, constraint or composite) to be added dynamically to the problem (associated to a given CSP variable, constraint or composite variable), during the resolution process, if a given condition on some variables is true. Solving a CCCSP is a decision problem which consists of finding an assignment of values to the CSP variables such that all the constraints are satisfied. This can be handled by approximation methods based on stochastic local search or by a systematic backtrack search algorithm where constraint propagation is used to prevent earlier later failure (Mouhoub & Sukpan 2007). On the other hand, solving a CCCSPP is an optimization problem which consists of finding the best solution according to the preference values. This can be done by a variant of the branch a bound algorithm we propose in this paper. Note that constraint propagation is also used here to prune some inconsistent values at the early stage of the resolution process. Experimental tests, we conducted on randomly generated CCCSPPs, favor the MAC principle (Haralick & Elliott 1980; Dechter 2003) as the constraint propagation strategy to be used within the branch and bound algorithm.

<sup>2</sup>We are assuming here that the constraints are binary and are defined in extension. For instance, the constraint  $C_{ij}$  between 2 variables  $X_i$  and  $X_j$  is the subset of the Cartesian product of  $X_i$ 's and  $X_j$ 's domains.

<sup>1</sup>We call CSP variables, the variables of a traditional CSP.

In the next section we will introduce the CCCSP model through an example. In section 3 we will summarize the related work in the area of constraint preferences. Section 4 is then dedicated to variable value, constraint, composite and conditional preferences. In Section 5 we present the branch and bound algorithm for solving CCCSPPs. Section 6 is dedicated to the experimental tests we conducted on randomly generated CCCSPPs. Conclusion and perspectives are finally listed in Section 7.

## Managing Conditional Constraints and Composite Variables

**Definition 1: Conditional and Composite Constraint Satisfaction Problem (CCCSP).** A CCCSP is a tuple  $\langle X, D_X, Y, D_Y, IV, C, A \rangle$ , where :

- $X = \{x_1, \dots, x_n\}$  is a finite set of CSP variables.
- $D_X = \{D_{x_1}, \dots, D_{x_n}\}$  is the set of domains of the CSP variables. Each domain  $D_{x_i}$  contains the possible values that  $X_i$  can take.
- $Y = \{y_1, \dots, y_m\}$  is the finite set of composite variables.
- $D_Y = \{D_{y_1}, \dots, D_{y_m}\}$  is the set of domains of the composite variables. Each domain  $D_{y_i}$  is the set of variables that the composite variable  $y_i$  can take.
- $IV$  is the set of initial variables (including composite variables):  $IV \subseteq X \cup Y$ .
- $C = \{C_1, \dots, C_p\}$  is the set of *compatibility constraints*. Each compatibility constraint is a binary relation between variables in case these latter variables are not composite, or a set of binary relations if at least one of the two variables involved is composite.
- $A$  is the set of *activity constraints*. Each activity constraint has the following form where  $Z_1, \dots, Z_p$  and  $T$  are variables (can be composite).  $Z_1 \wedge \dots \wedge Z_p \xrightarrow{\text{condition}} T$ . This activity constraint will activate  $T$  if  $Z_1, \dots, Z_p$  are active and *condition* holds on these variables. *condition* corresponds here to the assignment of particular values to the variables  $Z_1, \dots, Z_p$ .

*Example 1.* Let us consider the dress up game (see Figure 1) described as follows. This entertaining game provides the user with sets of clothes, accessories and shoes. The user will then use his free style Mix and Match imagination to create a complete outfit. In order to assist the user to have an appropriate outfit and be in a fashion trend, we can enhance the dress up game by adding tips and advices (expertise) from fashion designers. The fashion designers knowledge can be viewed as the set of compatible constraints, activity constraints and composite variables including the following: “running shoes does not match with pants”, “pant suit with business shirt make you look elegant”, “jacket with jeans is for a weekend look”, “set of skirt, jacket and shirt makes you look feminine”, and “two piece matched suit without jewelry is an appropriate dress for job interview”. A possible mix-match dress up is to wear a T-shirt, short and running shoes with a baseball cap. An another possible

clothes set is a dress, necklace(J1) and handbag (HB3) with pump shoes.

The dress up game can be formulated in a natural way using the CCCSP framework as follows (see Figure 1 for the CCCSP graph representation).

- Apparel and Shoes are initial active variables; the rest are nonactive variables. Apparel is a composite variable.
- Activity constraints:  $APPAREL = TOP \xrightarrow{\text{active}} BOTTOM$ ,  $APPAREL = BOTTOM \xrightarrow{\text{active}} TOP$ ,  $SET = Dress \xrightarrow{\text{active}} JEWELRY, HANDBAG$ ,  $TOP = T\text{-Shirt} \xrightarrow{\text{active}} HAT$ ,  $BOTTOM = Pant \xrightarrow{\text{active}} BELT$ ,  $BOTTOM = Jean \xrightarrow{\text{active}} BELT$ , and  $BOTTOM = Skirt \xrightarrow{\text{active}} HANDBAG$
- Compatibility constraints: binary relations between the following pairs of variables : (TOP, SHOES), (SET, SHOES), (BOTTOM, SHOES), (TOP, BELT), (HAT, BOTTOM), (JEWELRY, HANDBAG).

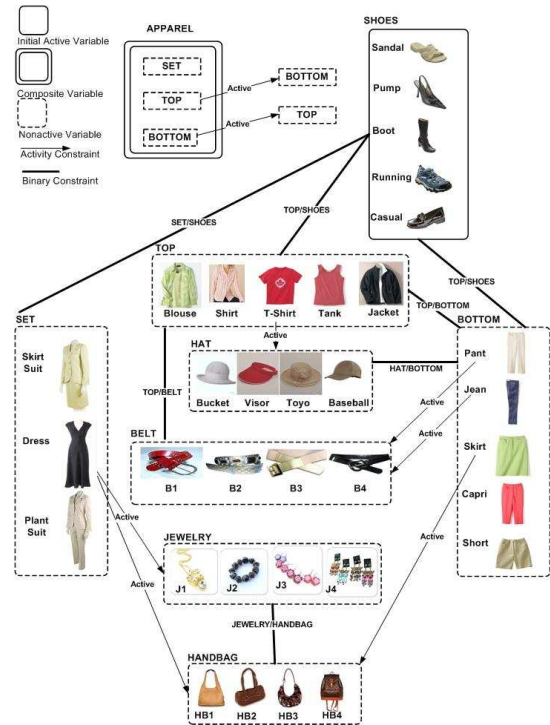


Figure 1: A dress up game example

## Related work

Managing preferences has been extensively studied in the past decade. The CP-net framework (Boutilier *et al.* 1999; 2004; Apt, Rossi, & Venable 2005) is a model for qualitative and conditional preferences under *ceteris paribus*. Preferences are represented separately from hard constraints. *Lexicographically ordered CSP* in (Freuder, Wallace, & Heffernan 2003) is an another alternative framework for preferred variables and values. In this latter model, variable selection is the primary factor while value assignment is secondary.

Recently, this framework has been extended to *Conditional lexicographic CSP* (Wallace 2005) for conditional preferences. Finally, quantitative preferences are modeled as a set of soft constraints in (Bistarelli, Montanari, & Rossi 1995; 1997; Schiex, Fargier, & Verfaillie 1995) supporting different kinds of soft constraints including fuzzy CSPs, weighted CSPs and partial CSPs. These latter frameworks based on a semiring structure have been widely used for quantitative preferences in CSPs (Bistarelli, Montanari, & Rossi 1995; 1997). A semiring is a tuple  $\langle A, +, \times, 0, 1 \rangle$  such that :

- $A$  is a set and  $0, 1 \in A$ ;
- $+$ , called the additive operation, is a commutative and associative operation such that  $0$  is its unit element;
- $\times$ , called the multiplicative operation, is an associative operation such that  $1$  is its unit element and  $0$  is its absorbing element.  $\times$  distributes over  $+$ .

The set of the semiring specifies the values to be associated with each tuple of values of the variable domain. The two semiring operations ( $+$  and  $\times$ ) represent constraint projection and combination respectively. A semiring for handling constraints is called *c-semiring*. A *c-semiring* is a semiring with additional properties on the two operations such that  $+$  is idempotent,  $\times$  is commutative, and  $1$  is the absorbing element of  $+$ . A partial order relation  $\leq$  is defined over  $A$  to compare tuples of values and constraints.

### Variable Value, Constraints, Composite and Conditional Preferences

In the following we will define, through several examples, the four types of preferences using the *c-semiring* structure  $\langle A, +, \times, 0, 1 \rangle$  for quantitative preferences (Bistarelli, Montanari, & Rossi 1995; 1997).

*Example 2.*

1. *There are four occasions to consider during the dress up: job interview, party, sport and camping. If the occasion selected is job interview or party, we prefer "set" instead of mix-and-match pieces ("top" and "bottom"). If it is camping or sport, we prefer "top" and "bottom" instead of "set".*
2. *We always prefer to wear "Casual" and "Boot" instead of "Sandal", "Running" and "Pump".*
3. *We like handbag "HB3" and "HB4" the most.*
4. *For match clothes (Top&Bottom constraint), we like "Blouse with Skirt", "T-Shirt with Capri" and "Jacket with Jean" the most.*
5. *For Set&Shoes constraint, we prefer "Skirt-Suit with Boot" and "Pant-Suit with Casual" to the rest.*

### Variable Value and Constraint Preferences

We define two types of preferences at the traditional CSP level. The first one is imposed on variable values while the second concerns the pairs of values within the binary constraints. Both preferences are defined over a *c-semiring*. We call the first one *Soft Unary Constraint (SUC)* and the second one *Soft Constraint (SC)*. As mentioned before in introduction, we are assuming here that the binary constraints are defined in extension.

**Definition 2: Soft Unary Constraint (SUC)** A Soft Unary Constraint (SUC) is a function  $f_{suc:x_i} : D_{x_i} \rightarrow A$ , where  $x_i$  is a CSP variable and  $D_{x_i}$  its domain of values.

*Example 3.* *The information in 2 and 3 in example 2 above can be formulated with SUCs as they concern preferences on the values of CSP variables SHOES and HANDBAG respectively. The SUC corresponding to 2 is the function  $f_{suc:SHOES}$  defined, for example, as follows.*

$$\begin{aligned} f_{suc:SHOES}(Casual) &= f_{suc:SHOES}(Boot) = 1.0. \\ f_{suc:SHOES}(Sandal) &= f_{suc:SHOES}(Running) = \\ f_{suc:SHOES}(Pump) &= 0.5. \end{aligned}$$

**Definition 3: Soft Constraint (SC)** A Soft Constraint (SC) is a function  $f_{sc:C_{ij}} : C_{ij} \rightarrow A$ , where  $C_{ij}$  is the binary constraint between the variables  $i$  and  $j$ .

*Example 4.* *The information in 4 and 5, in example 2 above, can be formulated with SCs as they are related to preferences on the constraints (Top,Bottom) and (Set,Shoes) respectively. The SC corresponding to 4 is the function  $f_{sc:(Top,Bottom)}$  defined, for example, as follows.  $f_{sc:(Top,Bottom)}((Blouse,Skirt)) = f_{sc:(Top,Bottom)}((T-Shirt,Capri)) = f_{sc:(Top,Bottom)}((Jacket,Jeans)) = 0.9$ . For any other possible pair of the constraint (Top,Bottom) the value of the SC function is for example equal to 0.6. Similarly, the SC corresponding to 5 is the function  $f_{sc:(Set,Shoes)}$  defined, for example, as follows.  $f_{sc:(Set,Shoes)}((skirt-suit,boot)) = f_{sc:(Set,Shoes)}((plant-suit,casual)) = 0.9$ . For any other possible pair of the constraint (Set,Shoes) the value of the SC function is for example equal to 0.6.*

### Composite and Conditional Preferences

A Composite Preference (CompP) is a function  $f_{c:X} : D_X \rightarrow A$ , where  $X$  is a composite variable and  $D_X$  its domain of values (CSP variables). This function allows us to favor some CSP variables within the domain of a given composite variable. The SUC  $f_{suc:X_i,x}$  of a CSP variable  $x$ , selected during the backtrack search from the domain of a composite variable  $X$ , is recomputed from the composite preference of  $X$  as follows.

**Definition 4: Composite Preference (CompP).** **Given:** a composite variable  $X$ , its domain  $D_X = \{x_1, \dots, x_p\}$ , a composite preference function  $f_{c:X}$ , and the selected CSP variable  $x_i$ , **then:**  $f_{suc:X_i,x}(v) = f_{c:X}(x_i) * f_{suc:x_i}(v)$ , **where**  $v$  is a possible value of  $x_i$ .

A Conditional Preference (CP) allows a preference function (SUC, SC and CompP) to be added dynamically to the CCCSPP when a given condition on CSP or composite variables is true. The condition corresponds here to the assignment of particular values to variables.

**Definition 5: Conditional Preference (CP)** Given a list of variables  $X_1, \dots, X_p$  and  $Y$  (CSP or composite variable) and a preference function  $f$ , a conditional preference has the following form :

$$X_1 \wedge \dots \wedge X_p \xrightarrow{\text{condition}} \text{associate } f \text{ to } Y. \text{ This conditional pref-}$$

erence will associate  $f$  to  $Y$  if *condition* holds on the variables  $X_1, \dots, X_p$ . *condition* corresponds here to the assignment of particular values to the variables  $X_1, \dots, X_p$ . Note that the variable  $Y$  can be associated to only one conditional preference  $f$ .

*Example 5. Information 1 in example 2 is formulated with the following conditional preference.*

1.  $OCCASION \xrightarrow{condition_1}$  assign the composite preference  $f_1$  to the composite variable APPAREL.
2.  $OCCASION \xrightarrow{condition_2}$  assign the composite preference  $f_2$  to the composite variable APPAREL.

where :

- *condition*<sub>1</sub> is:  $OCCASION = job-interview \vee OCCASION = party$
- *condition*<sub>2</sub> is:  $OCCASION = camping \vee OCCASION = sport$
- $f_1 = \{set = 0.9, top = 0.6, bottom = 0.6\}$
- $f_2 = \{top = 0.9, bottom = 0.9, set = 0.6\}$

### Global Preferences and Optimal Solution to the CCCSPP

In order to define the global preference of a solution to a CCCSPP, two other types of preference, namely Associated Local Constraint Preference (ALCP) and Consistent Binary Assignment Preference (CBAP), are introduced in the following. If  $C$  is a binary constraint between two CSP variables  $x_i$  and  $x_j$  then the ALCP of  $C, f_{as:C}$ , can be deduced from the SUC preferences associated to  $x_i$ 's and  $x_j$ 's values as follows.

**Definition 6: Associated Local Constraint Preference (ALCP).** **Given:**  $C_{ij}$  a constraint between two variables  $x_i$  and  $x_j$ , **then:** for each  $c \in C_{ij}$  such that  $c = (v_i, v_j)$ , for a given  $v_i \in D(x_i)$  and  $v_j \in D(x_j)$ ,  $f_{as:C_{ij}(c)} = \min(f_{suc:x_i}(v_i), f_{suc:x_j}(v_j))$ , **where**  $f_{suc:x_i}$  and  $f_{suc:x_j}$  are the SUC respectively for the variables  $x_i$  and  $x_j$ .

A solution to the CCCSPP is an assignment of values to all the CSP variables of the problem such that all the compatible constraints are satisfied. The global preference of a solution can be computed by performing the *min* operation on all the Consistent Binary Assignment Preferences (CBAPs) defined as follows using the ALCP defined above.

**Definition 7: Consistent Binary Assignment Preference (CBAP).** **Given:** two variables  $x_i$  and  $x_j$  sharing a constraint  $C_{ij}$ , a consistent binary assignment  $c = ([x_i = v_i], [x_j = v_j]) \in C_{ij}$  where:  $v_i \in Domain(x_i)$  and  $v_j \in Domain(x_j)$ ,  $\alpha_i = f_{suc:x_i}(v_i)$ ,  $\alpha_j = f_{suc:x_j}(v_j)$ ,  $\alpha_c = f_{sc:C_{ij}(c)}$ , and a ALCP  $f_{as:C_{ij}} = \min(\alpha_i, \alpha_j)$  **then**  $CBAP(v_i, v_j) = \min(f_{as:C_{ij}(c)}, \alpha_c)$ .

*Example 6. Let us assume that during the backtrack search we have made the following decision (assignment):*

- $OCCASION = job-interview$

*According to example 5 above, assigning job-interview to OCCASION will activate the composite preference  $f_1$  which*

*will favor the value set over top and bottom. set will then be the first value to assign to APPAREL. Since there is no SUC preference on the values of SET's domain, the choice for the first value to assign to SET will be guided by the SC of the constraint this latter variable shares with other active variables. Since SC of the constraint (SET, SHOES) favors 2 pairs involving skirt-suit and plant-suit, these latter are the first two values to choose for SET. Let us assume that skirt-suit is assigned to SET. SHOES will then be assigned to boot and the ALCP of the constraint (SET, SHOES) with the pair (skirt-suit, boot) will be computed as follows.  $f_{as} = f_{as:(SET, SHOES)}((skirt-suit, boot)) = \min(f_{suc:SET}(skirt-suit), f_{suc:SHOES}(boot)) = \min(1.0, 1.0) = 1.0$ . The CBAP of the constraint (SET, SHOES) will then be computed as follows.  $CBAP((skirt-suit, boot)) = \min(f_{as}, f_{sc:(SET, SHOES)}((skirt-suit, boot))) = \min(1.0, 0.9) = 0.9$ .*

**Definition 8: Global Preference (GP).** **Given:** a solution  $s = \{v_1, v_2, \dots, v_n\}$  to a CCCSPP, where  $n$  is the number of variables and each of the  $v_i$ 's belongs to the domain of the corresponding variable  $x_i$ ; and a set of consistent assignments  $ca = \{(v_i, v_j)$  where  $1 \leq i, j \leq n$  and  $v_i, v_j \in s$  and such that there is a constraint between the variables  $x_i$  and  $x_j\}$ , **then**  $GP(s) = \min \{CBAP(v_i, v_j) \text{ where } (v_i, v_j) \in ca\}$ .

**Definition 9: Optimal Solution (Opt).** An Optimal Solution (Opt) of a given CCCSPP  $P$  is the solution having the highest global preference degree. **Given:** CCCSPP  $P$  and a set of solutions  $S = \{s_1, \dots, s_n\}$  **then**  $Opt(P) = \max \{GP(s_1), \dots, GP(s_n)\}$ .

### Solving CCCSPPs

*Branch and Bound* is a well known method for solving optimization problems. In the case of CCCSPPs we apply this algorithm to find the optimal solution as follows. **Step 1.** The method starts with an initial problem containing a list of initially activated CSP and composite variables. In order to ensure that domain values are considered according to their preference functions, all the values within each domain are sorted in decreasing order of their SUC or CompP values (depending whether they belong to CSP variable or composite variable domains). Arc consistency is then applied on the initial CSP and composite variables in order to reduce some inconsistent values which will reduce the size of the search space. If the initial CSP is not arc consistent (in the case of an empty domain) then the method will stop. The CCCSPP is inconsistent in this case.

**Step 2.** Following the forward check principle (Haralick & Elliott 1980), pick an active variable  $x$ , assign a value to it and perform arc consistency between this variable and the non assigned active variables. If one domain of the non assigned variables becomes empty then assign another value to  $x$  or backtrack to the previously assigned variable if there are no more values to assign to  $x$ . Activate any preference function (through conditional preference) and any variable  $x'$  (through activity constraint) resulting from this assign-

ment and perform arc consistency between  $x'$  and all the active variables. If arc inconsistency is detected then deactivate  $x'$  and choose another value for  $x$  (since the current assignment of  $x$  leads to an inconsistent CCCSPP). If  $x$  is a composite variable then assign a CSP variable to it. Basically, this consists of replacing the composite variable with one variable  $x_i$  of its domain. We then assign a value to  $x_i$  and proceed as shown before except that we do not backtrack in case all values of  $x_i$  are explored. Instead, we will choose another CSP variable from the domain of the composite variable  $x$  or backtrack to the previously assigned variable if all values (CSP variables) of  $x$  have been explored. This process will continue until all the variables are assigned in which case we obtain a solution to the CCCSPP. Since we are looking for the highest global preference degree, the GP value of this solution will be used as a lower bound ( $LB$ ) of our branch and bound algorithm. Note that anytime a preference function  $f$  is activated (added to the CCCSPP) through a conditional preference, the domain of values of the variable associated to  $f$  is sorted according to this latter.

**Step 3.** The rest of the search space is then systematically explored as follows. Each time the current variable (CSP variable or composite) is assigned a value, an overestimation of the GP value of any possible solution following this decision is computed and used as an upper bound ( $UB$ ). If  $UB < LB$  then the current variable is assigned another value or backtrack to the previous variable if all the values have been explored. The overestimated GP is the minimum of the CBAPs of all the assigned variables and the estimated CBAPs involving non assigned variables (including those that can be activated during the remaining search process). An estimated CBAP involving a non assigned variable  $X_i$  is calculated as follows.

**If** the other variable  $X_j$  involved by the CBAP is an assigned variable then the estimated CBAP is the minimum of the following: the SUC of the value assigned to  $X_j$ , the maximum of the SCs of all the pairs within the constraint between  $X_i$  and  $X_j$ , and the maximum of the SUCs of all the values belonging to  $X_i$ 's domain.

**Else** ( $X_j$  is not assigned yet): the maximum of the SUCs of all the values belonging to  $X_j$ 's and  $X_i$ 's domains, and the minimum of the SCs of all the pairs within the constraint between  $X_i$  and  $X_j$ .

## Experimentation

In order to evaluate the methods we propose, we have performed experimental tests on randomly generated CCCSPPs. The experiments are performed on a PC Pentium 4 computer running Linux. All the procedures are coded in C/C++. CCCSPPs are built from CSPs randomly generated by the model RB proposed in (Xu & Li 2000). The choice of this model is motivated by the fact that it has exact phase transition and the ability to generate asymptotically hard instances. Following the model RB, we generate each CSP instance as follows using the parameters  $n$ ,  $p$ ,  $\alpha$  and  $r$  where  $n$  is the number of variables,  $p$  ( $0 < p < 1$ ) is the constraint tightness, and  $r$  and  $\alpha$  ( $0 < \alpha < 1$ ) are two positive constants.

1) Select with repetition  $m \ln n$  random constraints. Each

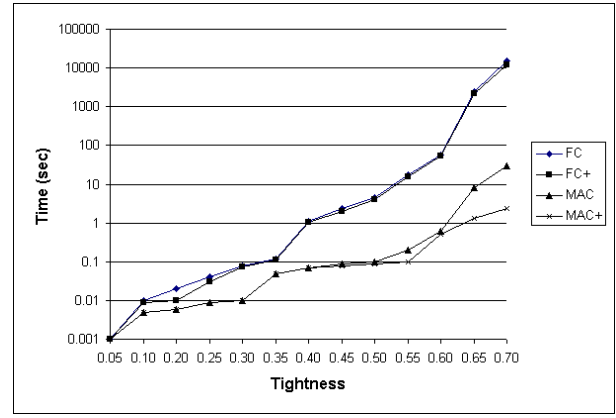


Figure 2: Comparative tests on random CCCSPPs.

random constraint is formed by selecting without repetition 2 of  $n$  variables. 2) For each constraint we uniformly select without repetition  $pd^k$  incompatible pairs of values, where  $d = n^\alpha$  is the domain size of each variable. Each CCCSPP instance is then generated as follows.

1. Randomly generate a CSP with the parameters  $n$ ,  $p$ ,  $\alpha$  and  $r$  as shown above.
2. Generate  $N$  composite variables each containing  $D$  simple variables.
3. Select with repetition  $r[(n+N)\ln(n+N) - n\ln n]$  new random constraints (between the  $n+N$  variables), each formed by selecting without repetition 2 of the  $n+N$  variables. This will guarantee that the total number of constraints is  $r(n+N)\ln(n+N)$ . For each constraint we uniformly select without repetition  $pd^k$  incompatible pairs of values.
4. Select  $I(n+N)$  initial variables from  $n+N$  ( $0 < I < 1$ ).
5. Select  $a(nd+ND)$  activity constraints for each of the  $n+N - I(n+N)$  non initial variables ( $0 < a < 1$ ).
6. Preference values are finally randomly distributed on the values of the different CSP and composite variables and also on the pair of values within each compatibility constraint.

As demonstrated in (Xu & Li 2000), when the number of variables approaches infinity the phase transition occurs when the constraint tightness  $p = 1 - e^{-\frac{\alpha}{r}}$ . Thus the phase transition is an asymptotic phenomenon since, only for infinite number of variables, we can have sharp phase transitions. In addition, the number of variables and constraints of the possible CSPs, each CCCSPP contains, is slightly different from the one of the CCCSPP they are generated from. The tests we have performed compare the following four propagation strategies used in step 2 of our branch and bound based solving method we described in the previous Section.

- 1) **Forward Check (FC).** This is the strategy we have described in the previous Section (step 2).
- 2) **Maintaining Arc Consistency (MAC).** This strategy maintains a full arc consistency on the current and future active variables (variables not assigned yet).

- 3) **FC+**. Same as FC except that the applicability of the arc consistency is extended to non active variables as well.
- 4) **MAC+**. Same as MAC except that the applicability of the arc consistency is extended to non active variables as well.

Figure 2 presents the results of comparative tests performed on random consistent CCCSPPs generated with the following parameters:  $n = 140$ ,  $N = 10$ ,  $D = 5$ ,  $\alpha = 0.8$ ,  $I = 0.8$ ,  $a = 0.2$  and  $r = 0.6$ . As mentioned earlier, the phase transition can be computed as follows:  $p = 1 - e^{-\frac{\alpha}{r}} = 1 - e^{-\frac{0.8}{0.6}} = 0.73$ . Thus, consistent instances are those with the tightness less than 0.73. For each test (corresponding to a particular tightness value  $p$ ), each of the four methods is executed on 100 instances and the average running time in seconds is taken. All the methods have similar running times in the case of under constrained problems. Indeed, in this particular case the extra effort done by MAC and MAC+ does not remove much of the inconsistent values and thus does not improve the overall running time to find a solution. However when we move toward the phase transition the extra work performed by MAC and especially MAC+ starts to pay off. At the phase transition MAC+ is almost 10,000 times faster than FC and FC+; and 10 times faster than MAC. In general MAC+ is the best method for solving random CCCSPPs.

## Conclusions

In this paper we have proposed a unique framework managing preferences at different levels of the constraint network and in a dynamic environment. This framework is very appealing for a wide variety of real world applications such as reactive scheduling and planning, logistics and configuration problems. The approach we adopted consists of converting a given constraint problem involving all the possible change that can occur depending on the validity of certain conditions into a constraint network where conditional constraints and composite variables are used to add new information (variables and their related constraints) to the constraint network in a dynamic manner during the resolution process. Preferences are associated to variable and constraint values as well as composite variables, in order to favor some solutions of the constraint problem. Finding the best solution is carried out by a variant of the branch and bound algorithm we propose. In order to evaluate the time performance of our solving method, we conducted experimental tests comparing different propagation strategies on randomly generated CCCSPPs. The results favor the MAC principle (Haralick & Elliott 1980; Dechter 2003) over the other strategies. In the near future, we intend to conduct more experimental study on some real life applications under constraints. Another perspective is to consider approximation methods such as Stochastic Local Search (SLS) (Selman & Kautz 1993), Genetic Algorithms (GAs) (Craenen & Eiben 2003) and Ant Colony Algorithms (ACAs) (Stützle & Hoos 1998). While these techniques do not always guarantee an optimal solution to the problem, they are very efficient in time (comparing to branch

and bound) and can thus be useful if we want to trade the optimality of the solution for the time performance.

## References

- Apt, K. R.; Rossi, F.; and Venable, K. B. 2005. CP-nets and Nash equilibria.
- Bistarelli, S.; Montanari, U.; and Rossi, F. 1995. Constraint solving over semirings. In Mellish, C., ed., *IJCAI'95: Proceedings International Joint Conference on Artificial Intelligence*.
- Bistarelli, S.; Montanari, U.; and Rossi, F. 1997. Semiring-based constraint satisfaction and optimization. *Journal of the ACM* 44(2):201–236.
- Boutilier, C.; Brafman, R.; Hoos, H.; and Poole, D. 1999. Reasoning with conditional ceteris paribus statements. In *Proc. of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence UAI-99*, 71–80.
- Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. Preference-Based Constrained Optimization with CP-Nets. *Computational Intelligence* 20(2):137–157.
- Craenen, B., and Eiben, A. 2003. Comparing evolutionary algorithms on binary constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation* 7(5):424–444.
- Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.
- Freuder, E.; Wallace, R. J.; and Heffernan, R. 2003. Ordinal Constraint Satisfaction. In *Fifth Internal. Workshop on Soft Constraints - SOFT'02*.
- Haralick, R., and Elliott, G. 1980. Increasing tree search efficiency for Constraint Satisfaction Problems. *Artificial Intelligence* 14:263–313.
- Mouhoub, M., and Sukpan, A. 2007. Managing conditional and composite csps. In *The 20th Canadian Conference on Artificial Intelligence*, 216–227.
- Schiex, T.; Fargier, H.; and Verfaillie, G. 1995. Valued constraint satisfaction problems: Hard and easy problems. In Mellish, C., ed., *IJCAI'95: Proceedings International Joint Conference on Artificial Intelligence*.
- Selman, B., and Kautz, H. A. 1993. An empirical study of greedy local search for satisfiability testing. In *AAAI'93*, 46–51.
- Stützle, T., and Hoos, H. 1998. Improvements on the Ant System: Introducing the MAX-MIN Ant System. *Artificial Neural Networks and Genetic Algorithms* 245–249.
- Wallace, R. J. 2005. Conditional lexicographic orders in constraint satisfaction problems. In *Eleventh International Conference on Principles and Practice of Constraint Programming (CP 2005)*.
- Xu, K., and Li, W. 2000. Exact Phase Transitions in Random Constraint Satisfaction Problems. *Journal of Artificial Intelligence Research* 12:93–103.