Interchangeability in Soft CSPs

Stefano Bistarelli[1], Boi Faltings[2] and Nicoleta Neagu[2]

[1] Istituto di Informatica e Telematica (IIT), CNR, Pisa, Italy
`Stefano.Bistarelli@iit.cnr.it`,
[2] Artificial Intelligence Laboratory (LIA), EPFL, Ecublens, Switzerland
`boi.faltings|nicoleta.neagu@epfl.ch`

Abstract. Substitutability and interchangeability in constraint satisfaction problems (CSPs) have been used as a basis for search heuristics, solution adaptation and abstraction techniques. In this paper, we consider how the same concepts can be extended to *soft* constraint satisfaction problems (SCSPs).
We introduce the notions of *threshold* α and *degradation* δ for substitutability and interchangeability. In $_{\alpha}$interchangeability, values are interchangeable in any solution that is better than a threshold α, thus allowing to disregard differences among solutions that are not sufficiently good anyway. In $^{\delta}$interchangeability, values are interchangeable if their exchange could not degrade the solution by more than a factor of δ.
Theorems, algorithms to compute $(^{\delta}/_{\alpha})$interchangeable sets of values, and a more general treatment of all the ideas presented in this paper can be found in [2].

1 Introduction

Substitutability and interchangeability in CSPs have been introduced by Freuder ([8]) in 1991 with the intention of improving search efficiency for solving CSP.

Interchangeability has since found other applications in abstraction frameworks ([10, 15, 8, 5]) and solution adaptation ([14, 11]). One of the difficulties with interchangeability has been that it does not occur very frequently.

In many practical applications, constraints can be violated at a cost, and solving a CSP thus means finding a value assignment of minimum cost. Various frameworks for solving such soft constraints have been proposed [9, 6, 12, 7, 13, 3, 4, 1]. The soft constraints framework of c-semirings [3, 1] has been shown to express most of the known variants through different instantiations of its operators, and this is the framework we are considering in this paper.

The most straightforward generalization of interchangeability to soft CSP would require that exchanging one value for another does not change the quality of the solution at all. This generalization is likely to suffer from the same weaknesses as interchangeability in hard CSP, namely that it is very rare.

Fortunately, soft constraints also allow weaker forms of interchangeability where exchanging values may result in a degradation of solution quality by some measure δ. By allowing more degradation, it is possible to increase the amount of interchangeability in a problem to the desired level. We define $^{\delta}$substitutability/interchangeability as a concept which ensures this quality. This is particularly useful when interchangeability is used for solution adaptation.

Another use of interchangeability is to reduce search complexity by grouping together values that would never give a sufficiently good solution. In $_{\alpha}$substitutability/interchangeability, we consider values interchangeable if they give equal solution quality in all solutions better than α, but possibly different quality for solutions whose quality is $\leq \alpha$.

Just like for hard constraints, full interchangeability is hard to compute, but can be approximated by neighbourhood interchangeability which can be computed efficiently and implies full interchangeability. We define the same concepts for soft constraints.

2 Background

2.1 Soft CSPs

A soft constraint may be seen as a constraint where each instantiations of its variables has an associated value from a partially ordered set which can be interpreted as a set of preference values. Combining constraints will then have to take into account such additional values, and thus the formalism has also to provide suitable operations for combination ($\times$) and comparison ($+$) of tuples of values and constraints. This is why this formalization is based on the concept of c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, which is just a set A plus two operations[1].

Constraint Problems. Given a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and an ordered set of variables V over a finite domain D, a *constraint* is a function which, given an assignment $\eta : V \to D$ of the variables, returns a value of the semiring.

By using this notation we define $\mathcal{C} = \eta \to A$ as the set of all possible constraints that can be built starting from S, D and V.

Consider a constraint $c \in \mathcal{C}$. We define his support as $supp(c) = \{v \in V \mid \exists \eta, d_1, d_2 . c\eta[v := d_1] \neq c\eta[v := d_2]\}$, where

$$\eta[v := d]v' = \begin{cases} d & \text{if } v = v', \\ \eta v' & \text{otherwise.} \end{cases}$$

Note that $c\eta[v := d_1]$ means $c\eta'$ where η' is η modified with the association $v := d_1$ (that is the operator [ ] has precedence over application).

Fig. 1 shows the graph representation of a fuzzy CSP[2]. Variables and constraints are represented respectively by nodes and by undirected (unary for c_1 and c_3 and binary for c_2) arcs, and semiring values are written to the right of the corresponding tuples. Here we assume that the domain D of the variables contains only elements a and b and c.

[1] In [3] several properties of the structure are discussed. Let us just remind that it is possible to define a partial order $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$.

[2] Fuzzy CSPs can be modeled in the SCSP framework by choosing the c-semiring $S_{FCSP} = \langle [0, 1], max, min, 0, 1 \rangle$.

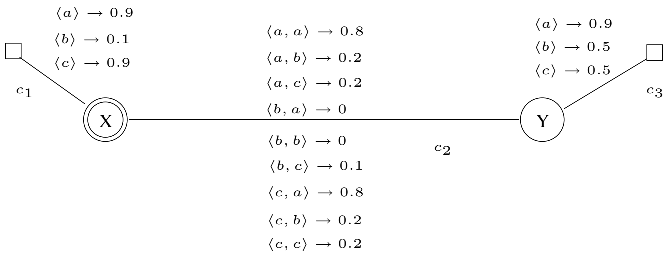

Fig. 1: A fuzzy CSP.

Combining soft constraints. When there is a set of soft constraints $\mathcal{C}$, the combined weight of the constraints is computed using the operator $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ is defined as $(c_1 \otimes c_2)\eta = c_1\eta \times_S c_2\eta$.

For instance, consider again the fuzzy CSP of Fig. 1. For the tuple $\langle a, a \rangle$ (that is, $x = y = a$), we have to compute the minimum between 0.9 (which is the value assigned to $x = a$ in constraint c_1), 0.8 (which is the value assigned to $\langle x = a, y = a \rangle$ in c_2) and 0.9 (which is the value for $y = a$ in c_3). Hence, the resulting value for this tuple is 0.8.

2.2 Interchangeability

Interchangeability in constraint networks has been first proposed by Freuder [8] to capture equivalence among values of a variable in a discrete constraint satisfaction problem. Value $v = a$ is *substitutable* for $v = b$ if for any solution where $v = a$, there is an identical solution except that $v = b$. Values $v = a$ and $v = b$ are *interchangeable* if they are substitutable both ways.

3 Interchangeability in Soft CSPs

In soft CSPs, there is no crisp notion of consistency. In fact, each tuple is a possible solution, but with different level of preference. Therefore, in this framework, the notion of interchangeability become finer: to say that values a and b are interchangeable we have also to consider the assigned semiring level.

More precisely, if a domain element a assigned to variable v can be substituted in each tuple solution with a domain element b without obtaining a worse semiring level we say that b is full substitutable for a.

Definition 1 (Full Substitutability (FS)). *Consider two domain values b and a for a variable v, and the set of constraints C; we say that b is Full Substitutable for a on v ($b \in FS_v(a)$) if and only if $\bigotimes C\eta[v := a] \leq_S \bigotimes C\eta[v := b]$*

When we restrict this notion only to the set of constraints C_v that involve variable v we obtain a local version of substitutability.

Definition 2 (Neighborhood Substitutability (NS)). *Consider two domain values b and a for a variable v, and the set of constraints C_v involving v; we say that b is neighborhood substitutable for a on v ($b \in NS_v(a)$) if and only if $\bigotimes C_v\eta[v := a] \leq_S \bigotimes C_v\eta[v := b]$*

When the relations hold in both directions, we have the notion of Full/Neighborhood interchangeability of b with a.

Definition 3 (Full and Neighborhood Interchangeability (FI and NI)). *Consider two domain values b and a, for a variable v, the set of all constraints C and the set of constraints C_v involving v. We say that b is Full interchangeable with a on v ($FI_v(a/b)$) if and only if $b \in FS_v(a)$ and $a \in FS_v(b)$. We say that b is Neighborhood interchangeable with a on v ($NI_v(a/b)$) if and only if $b \in NS_v(a)$ and $a \in NS_v(b)$.*

This means that when a and b are interchangeable for variable v they can be exchanged without affecting the level of any solution.

As an example of interchangeability and substitutability consider the fuzzy CSP represented in Fig. 1. The domain value c is neighborhood interchangeable with a on x ($NI_x(a/c)$); in fact, $c_1 \otimes c_2\eta[x := a] = c_1 \otimes c_2\eta[x := c]$ for all η.

The domain values c and a are also neighborhood substitutable for b on x ($\{a, c\} \in NS_v(b)$). In fact, for any η we have $c_1 \otimes c_2\eta[x := b] \leq c_1 \otimes c_2\eta[x := c]$ and $c_1 \otimes c_2\eta[x := b] \leq c_1 \otimes c_2\eta[x := a]$.

3.1 Degradations and Thresholds

In soft CSPs, it is possible to obtain more interchangeability by allowing degrading the solution quality when values are exchanged. We call this ${}^{\delta}$interchangeability, where δ is the *degradation* factor.

When searching for solutions to soft CSP, we can gain efficiency by not distinguishing values that could in any case not be part of a solution of sufficient quality. In ${}_{\alpha}$interchangeability, two values are interchangeable if they do not affect the quality of any solution with quality better than α. We call α the *threshold* factor.

Both concepts can be combined, i.e. we can allow both degradation and limit search to solutions better than a certain threshold (${}^{\delta}_{\alpha}$interchangeability).

Thus we define:

Definition 4 (${}^{\delta}/_{\alpha}$Full Substitutability (${}^{\delta}/_{\alpha}FS$)). *Consider two domain values b and a for a variable v, the set of constraints C and the semiring levels δ and α; we say that b is ${}^{\delta}$*full Substitutable *for a on v ($b \in {}^{\delta}FS_v(a)$) if and only if for all assignments η, $\bigotimes C\eta[v := a] \times_S \delta \leq_S \bigotimes C\eta[v := b]$.*

*We say that b is ${}_{\alpha}$*full substitutable *for a on v ($b \in {}_{\alpha}FS_v(a)$) if and only if for all assignments η, $\bigotimes C\eta[v := a] \geq \alpha \implies \bigotimes C\eta[v := a] \leq_S \bigotimes C\eta[v := b]$*

Similar to the plain version, *neighbourhood*${}^{\delta}_{\alpha}$substitutability is obtained by only evaluating the definition on the neighbourhood of a variable, and ${}^{\delta}_{\alpha}$interchangeability is defined as substitutability both ways.

As an example consider Fig. 1. The domain values c and b for variable y are ${}_{0.2}$Neighborhood Interchangeable. In fact, the tuple involving c and b only differ for the tuple $\langle b, c\rangle$ that has value 0.1 and for the tuple $\langle b, b\rangle$ that has value 0. Since we are interested only to solutions greater than 0.2, these tuples are excluded from the match.

In [2], we present a number of useful theorems relating to ${}^{\delta}_{\alpha}$interchangeability, in particular that $neighbourhood^{\delta}_{\alpha}$interchangeability implies $full^{\delta}_{\alpha}$interchangeability, and results on transitivity and limit cases.

4 Conclusions

Interchangeability in CSPs has found many applications for problem abstraction and solution adaptation. In this paper, we give hints to extend the concept of Interchangeability to soft CSPs in a way that maintains the attractive properties already known for hard constraints.

The two parameters α and δ allow us to express a wide range of practical situations. The threshold α is used to eliminate distinctions that would not interest us anyway, while the allowed degradation δ specifies how precisely we want to optimize our solution.

References

[1] Bistarelli, S.: Soft Constraint Solving and programming: a general framework. PhD thesis, Dipartimento di Informatica, Università di Pisa, Italy (2001) TD-2/01.

[2] Bistarelli, S., Faltings, B., Neagu, N.: A definition of interchangeability for soft csps. In: Proc. of the Joint Workshop of the ERCIM Working Group on Constraints and the CologNet area on Constraint and Logic Programming on Constraint Solving and Constraint Logic Programming. (2002) Selected papers will be published in LNCS series.

[3] Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based Constraint Solving and Optimization. J. ACM **44** (1997)

[4] Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based Constraint Logic Programming: Syntax and Semantics. TOPLAS ACM **23** (2001)

[5] Choueiry, B.Y.: Abstraction Methods for Resource Allocation. PhD thesis, EPFL PhD Thesis no 1292 (1994)

[6] Dubois, D., Fargier, H., Prade, H.: The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In: Proc. IEEE International Conference on Fuzzy Systems. (1993)

[7] Fargier, H., Lang, J.: Uncertainty in constraint satisfaction problems: a probabilistic approach. In: Proc. European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty (ECSQARU). Volume 747 of LNCS. (1993)

[8] Freuder, E.C.: Eliminating interchangeable values in constraint satisfaction problems. In: Proc. of AAAI-91. (1991)

[9] Freuder, E., Wallace, R.: Partial constraint satisfaction. AI Journal **58** (1992)

[10] Haselbock, A.: Exploiting interchangeabilities in constraint satisfaction problems. In: Proc. of the 13th IJCAI. (1993)

[11] Neagu, N., Faltings, B.: Exploiting interchangeabilities for case adaptation. In: In Proc. of the 4th ICCBR01. (2001)

[12] Ruttkay, Z.: Fuzzy constraint satisfaction. In: Proc. 3rd IEEE International Conference on Fuzzy Systems. (1994)

[13] Schiex, T., Fargier, H., Verfaille, G.: Valued Constraint Satisfaction Problems: Hard and Easy Problems. In: Proc. IJCAI95. (1995)

[14] Weigel, R., Faltings, B.: Interchangeability for case adaptation in configuration problems. In: Proc. of the AAAI98 Spring Symposium on Multimodal Reasoning, Stanford, CA. (1998) TR SS-98-04.

[15] Weigel, R., Faltings, B.: Compiling constraint satisfaction problems. Artificial Intelligence **115** (1999)