On the Computation of Local Interchangeability in Soft Constraint Satisfaction Problems

Nicoleta Neagu[1] and **Stefano Bistarelli**[2,3] and **Boi Faltings**[1]

[1]Arti£cial Intelligence Laboratory (LIA), Computer Science Department, EPFL
CH-1015, Ecublens, Switzerland
{boi.faltings,nicoleta.neagu}@ep¤.ch

[2]Istituto di Informatica Telematica (IIT), CNR,
Via G. Moruzzi 1, I-56124, Pisa, Italy,
Stefano.Bistarelli@iit.cnr.it

[3]Dipartimento di Scienze, Universitá "D'Annunzio"
Viale Pindaro 87, I-65127 Pescara, Italy
bista@sci.unich.it

Abstract

Freuder in (1991) de£ned interchangeability for classical Constraint Satisfaction Problems (CSPs). Recently (2002), we extended the de£nition of interchangeability to *Soft* CSPs and we introduced two notions of relaxations based on degradation δ and on threshold α ($^{\delta}$neighborhood interchangeability ($_{\alpha}NI$)and $_{\alpha}$neighborhood interchangeability $^{\delta}NI$).

In this paper we study the presence of these relaxed version of interchangeability in random soft CSPs. We give a description of the implementation we used to compute interchangeabilities and to make the tests. The experiments show that there is high occurrence of $_{\alpha}NI$ and $^{\delta}NI$ interchangeability around optimal solution in Fuzzy CSP and weighted CSPs. Thus, these algorithms can be used succesfully in solution update applications. Moreover, it is also showed that NI interchangeability can well approximate full interchangeability (FI).

Introduction

Interchangeability in constraint networks has been £rst proposed by Freuder (1991) to capture equivalence among the 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. *Full Interchangeability* considers all constraints in the problem and checks if a values a and b for a certain variable v can be interchanged without affecting the global solution. The localized notion of *Neighbourhood Interchangeability* considers only the constraints involving a certain variable v. Interchangeability has since found other applications in abstraction frameworks (Haselbock 1993; Weigel & Faltings 1999; Choueiry 1994) and solution adaptation (Weigel & Faltings 1998). One of the dif£culties 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 £nding a value assignment of minimum cost. Various frameworks for solving such soft constraints have been proposed (Freuder & Wallace 1992; Dubois, Fargier, & Prade 1993; Ruttkay 1994; Schiex, Fargier, & Verfaille 1995; Bistarelli, Montanari, & Rossi 1997). The soft constraints framework of c-semirings (Bistarelli, Montanari, & Rossi 1997) 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.

In (Bistarelli, Faltings, & Neagu 2002) we extended the notion of interchangeability to Soft CSPs. 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. Nevertheless, 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. The $^{\delta}$substitutability/interchangeability concept 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 suf£ciently 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 $\leq \alpha$.

In this paper we present a java implementation to deal with soft CSPs on top of the Java Constraint Library (JCL) (Bruchez & Torrens 1996). We develop some basic solution search strategies and a module to compute $^{\delta}/_{\alpha}$substitutability/interchangeability. The behavior of NI sets in the Soft CSP frameworks is still unexploited. For this motivation we study and evaluate here how NI behaves in soft CSPs frameworks (mainly fuzzy and weighted CSPs).

In the following we £rst remind some details about In-

terchangeability and soft Constraint Satisfaction Problems. Then, we discuss the implemented system we use to compute ($^{\delta}/_{\alpha}$)interchangeability. A section describing the results of the tests follows. A conclusions and possible future work section conclude the paper.

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 £nite 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 de£ne $\mathcal{C} = \eta \to A$ 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 de£ne 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 modi£ed with the association $v := d_1$ (that is the operator [ ] has precedence over application).

Combining soft constraints. Given the set $\mathcal{C}$, the combination function $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ is de£ned as $(c_1 \otimes c_2)\eta = c_1\eta \times_S c_2\eta$.

In words, combining two constraints means building a new constraint whose support involve all the variables of the original ones, and which associates to each tuple of domain values for such variables a semiring element which is obtained by multiplying the elements associated by the original constraints to the appropriate subtuples.

Interchangeability. In soft CSPs, there are not any 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 £ner: 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 (that is $b \in FS_v(a)$ 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 (that is $b \in NS_v(a)$ 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. This means that when a and b are interchangeable for variable v they can be exchanged without affecting the level of any solution.

Degradations and Thresholds. In soft CSPs, any value assignment is a solution, but may have a very bad preference value. This allows broadening the original interchangeability concept to one that also allows 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, it is possible to gain ef£ciency by not distinguishing values that could in any case not be part of a solution of suf£cient 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). By extending the previous de£nitions we obtain thresholds and degradation version of full/neighbourhood substitutability/interchangeability:

- 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, $\bigotimes C\eta[v := a] \geq \alpha \implies \bigotimes C\eta[v := a] \leq_S \bigotimes C\eta[v := b]$.

The Java Implementation

We implemented the soft CSP module and the interchangeability module as an extension of the JCL, developed at the Arti£cial Intelligence Laboratory (EPFL) (Bruchez & Torrens 1996). JCL is implemented in java which ensures portability on all the platforms. The library contain a CSP package describing constraint satisfaction problems and several solvers. In this section, we describe brie¤y the JCL andthe CSP module.

The Java Constraint Library. The Java Constraint Library (JCL) is a library containing common constraint satisfaction techniques which provides services for creating and managing discrete CSPs and applying preprocessing and search algorithms.

We developed two new packages on top of JCL: the £rst one models and solve soft constraint satisfaction problems; the second one computes interchangeabilities for crisp and soft CSPs, see Figure 1.

The Soft_CSP Module. The Soft_CSP package extends the CSP class in order to support softness. We implemented the scheme from (Bistarelli, Montanari, & Rossi 1997) where preferences levels are assigned both to variables values (implemented as soft unary constraints by the class `SoftUnaryConstraint`) and to tuples of values over the constraints. In particular, in the actual

[1] In (Bistarelli, Montanari, & Rossi 1997) several properties of the structure are discussed. Let us just remind that it is possible to de£ne a partial order $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$.

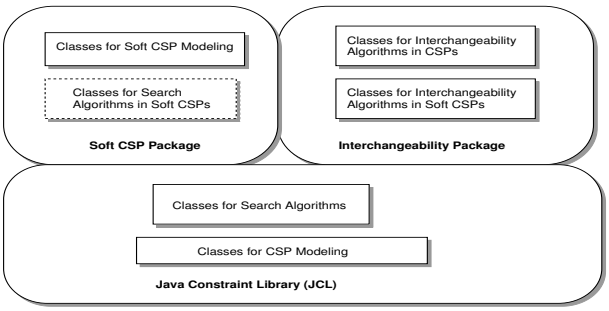

Figure 1: *Soft CSP and Interchangeability Modules on top of Java Constraint Language(JCL).*

implementation we only consider binary constraints (by `SoftBinaryConstraint`), but the class can be easily extended in this direction.

The Soft_CSP package supports classical, fuzzy, probabilistic and weighted CSPs by using appropriate semirings as showed in Figure 2. The semiring class parameterizes the type of the CSP and the respective operations of constraints combinations (and projection).

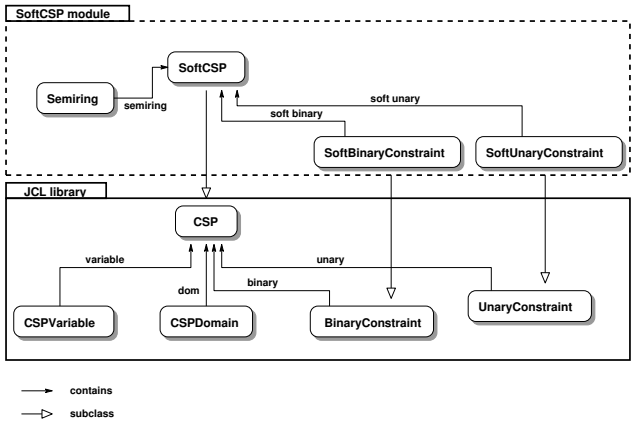

Figure 2: *The Soft CSP module.*

The Interchangeability Module. After reading the Soft CSP from a data £le (we use to describe problem SCSPDL an extension of the CSP Description Language (Bruchez & Torrens 1996)) a Soft CSP java object is constructed. Each CSP domain and variable is represented as an object. After this, unary and binary constraints are created.

The next step is to give the SoftCSP object as input to `DeltaInterchangeability` class (possibly with the desired α and δ a tolerance factor) as shown in Figure 3.

Figure 4 describe the classes implementing the Neighbourhood Interchangeability algorithms implemented by the Interchangeability Module. By now, this module supports computation of NI sets for crisp CSPs and ${}^{\delta}/_{\alpha}NI$ sets for soft CSPs.

They are implemented in a way to ensure an easy extension to several interchangeability approximation and algorithms. For this motivation we implemented the abstract class `InterchangeabilityFinder` which is extended

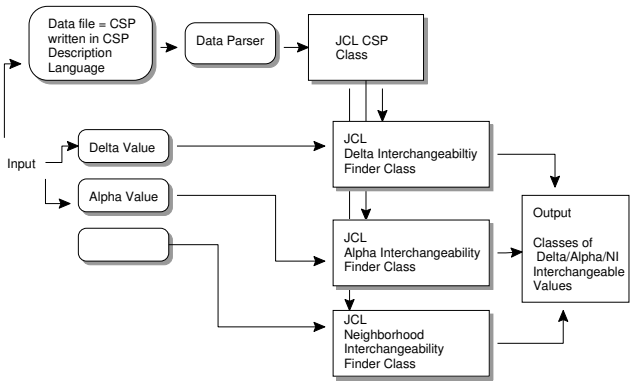

Figure 3: *Interchangeability computation ¤ow.*

for the speci£ed `Delta` and `Alpha`.

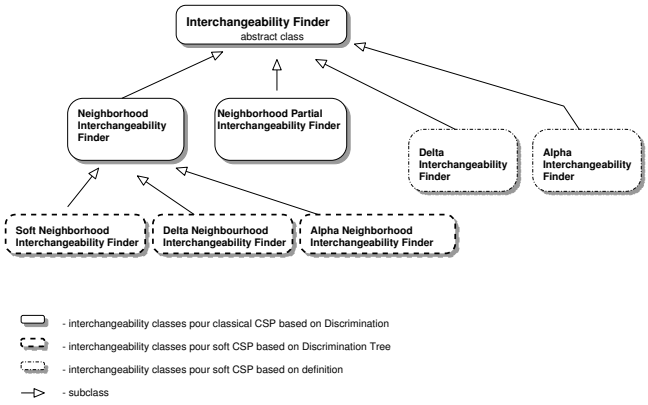

Figure 4: *Classes structures of Interchangeability Algorithms.*

Test Evaluation

Occurrence of NI in classical CSP have been already studied to improve search (Benson & Freuder 1992), for resource allocation application (Boi Faltings 1995) and for con£guration problems (Neagu & Faltings 1999). One of the main result is that in problems of small density the number of NI sets increases with the domain size.

The behavior of NI sets in the Soft CSP frameworks is still unexploited. For this motivation we study and evaluate here how NI behaves in the Soft CSP framework.

We have done our experiments for fuzzy and weighted CSP representing the important class of Soft CSPs dealing with an idempotent and non-idempotent times operation respectively. The motivation for considering both classes come from the fact that solving Soft CSP when the combination operation is not idempotent is extremely hard (Bistarelli, Montanari, & Rossi 1997).

Usually the structure of a problem is characterised by four parameters:

- *Problem Size*: This is usually the number of its variables;
- *Domain Size*: The average of the cardinality of the domain of the variables;
- *Problem Density*: This value (measured on the interval [0,1]) is the ratio of the number of constraints relatively

to the minimum and maximum number of allowed constraints in the given problem; Considering the constraint problem as a constraint graph $G = (V, E)$ where V represents the vertices (variables) (with $n := |V|$) and E edges (constraints) (with $e := |E|$); the density is computed as $denscsp = \frac{e - e\_min}{e\_max - e\_min}$, where $e\_min = n - 1$ and $e\_max = \frac{n(n-1)}{2}$;

- *Problem tightness* This measure is obtained as the average of tightness of all the constraints. For soft constraints we consider it as the ratio between the sum of the semiring values associated to all the tuples in all the constraints, and the value obtained by multiplying the **1** element of the semiring (that is the maximum) for the number of all possible tuple (that is the $constraintnumber \times domainsize$.

$({}^{\delta}/_{\alpha})NI$ in Fuzzy and Weighted CSPs

For both fuzzy and weighted CSPs we observed that the density and number of variables do not in¤uence much occurrence of interchangeable values. There is instead a (weak) dependency from the domain size: *the number of interchangeabilities increases with the resources*. This result from the test is obvious when dealing with crisp CSPs, but for soft problems this could be not so obvious.

We followed the model of measuring NI sets developed in (Boi Faltings 1995) with some adaptation needed in order to deal with soft constraints. We report here the results for problem sizes n = 10 and n = 20, while varying the density $dens - csp \in \{0.1, 0.2, \ldots, 1\}$ and the maximum domain size $dom - size = \{\frac{n}{10}, \frac{2n}{10}, \ldots, \frac{9n}{10}, n\}$. For each case, ten random problems were generated and then graphically represented by considering the measures described below.

In all the graph we highlight where is the position of the optimal solution. In fact, when dealing with crisp CSP there is not any notion of optimality, but for soft CSP each solution has an associated level of preference. It is important to study NI occurrence around optimal solutions because we are often interested to discard solutions of bad quality.

${}^{\delta}/_{\alpha}NI$ Occurrency in Fuzzy CSPs. In Figure 5 and in Figure 6, we represent ${}_{\alpha}NI$ and ${}^{\delta}NI$ occurrency in Fuzzy CSPs.

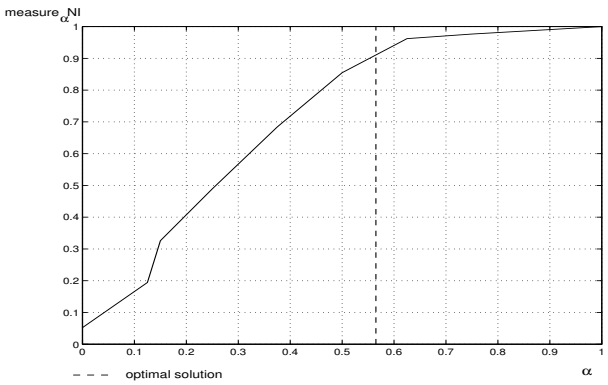

Figure 5: *Occurrence of ${}_{\alpha}$interchangeability around optimal solution in fuzzy CSP.*

The parameters $measure_{\alpha}NI$ (and $measure^{\delta}NI$) computes the average number of ${}_{\alpha}NI$ (${}^{\delta}NI$)interchangeable pairs values over the whole problem, divided by the maximum potential number of relations using the formula:

$$measure^{\delta}_{\alpha}NI = \frac{\sum_{k=1}^{n} \frac{\alpha NIV_k * 2}{domSize_{V_k} * (domSize_{V_k} - 1)}}{n},$$

where n represents the problem size and ${}_{\alpha}NI$ V_k all the ${}_{\alpha}$interchangeable pairs values for variable V_k.

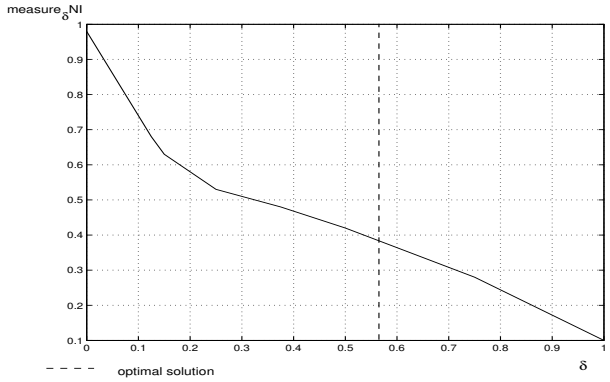

Figure 6: *Occurrence of ${}^{\delta}$interchangeability around optimal solution in fuzzy CSP.*

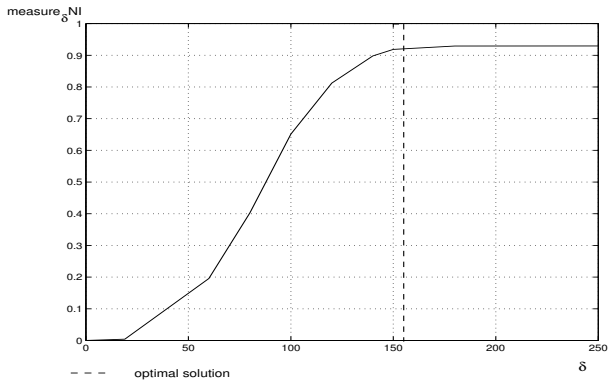

Figure 7: *Occurrence of ${}^{\delta}$interchangeability around optimal solution in weighted CSP.*

Similar results hold for ${}^{\delta}/_{\alpha}$interchangeability when dealing with weighted instead of fuzzy CSPs. In Figure 7, we represent ${}^{\delta}NI$ occurrency in weighted CSPs. Notice that the shape of the function seems the opposite to that of Figure 6 only because the order in the fuzzy semiring (max-min) is opposite to the weighted semiring (min-sum).

Looking at Figure 5, Figure 6 and Figure 7 we notice that the number of ${}_{\alpha}$interchangeability increase with α and that ${}^{\delta}$interchangeability decrease with δ (w.r.t. the semiring order). This experimental proof support the theorems in (Bistarelli, Faltings, & Neagu 2002).

The experimental facts show also that there is high occurrence of ${}_{\alpha}NI$ and ${}^{\delta}NI$ interchangeability around optimal solution in Fuzzy CSP and weighted CSPs. Thus, these algorithms can be used succesfully in solution update applica-

tions. The use of these technique to improve search have to be investigated.

$^{\delta}NI$ versus $^{\delta}FI$ Occurrency in Weighted CSPs. Since NI computes a subset of FI, it is important to investigate how many interchangeabilities we can £nd by computing locally neighborhood interchangeabilities w.r.t. full interchangeabilities. In Figures 8 and in Figure 9 $ratioNI/FI$ represent the rate between Neighborhood and Full Interchangeability for fuzzy and weighted CSP respectively.

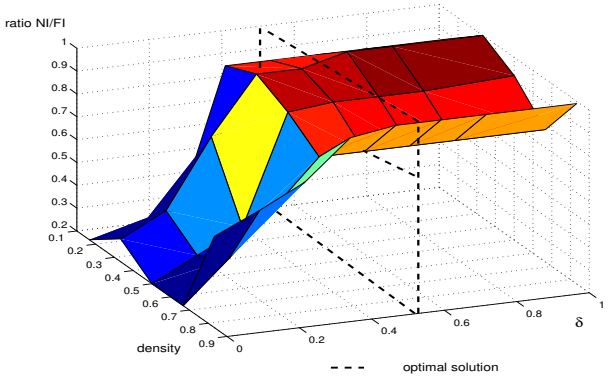

Figure 8: *Neighborhood Interchangeability relatively to Full Interchangeability for $^{\alpha}$interchangeability in Fuzzy CSP.*

In our experiments we used the measure:

$$ratioNI/FI = \frac{\sum_{k=1}^{n} {}^{\delta}NIV_k}{\sum_{k=1}^{n} {}^{\delta}FIV_k}$$

where $^{\delta}NIV_k$ represents the number of $^{\delta}NI$ interchangeable values pairs for variable V_k and $^{\delta}FIV_k$ represents the number of $^{\delta}FI$ interchangeable values pairs for variable V_k.

In Figures 8 and 9, we see that the ratio $^{\delta}NI/^{\delta}FI$ stays between 0.7 and 0.9 around optimal solution for fuzzy and weighted CSP as well. Thus, NI interchangeability can well approximate FI interchangeability.

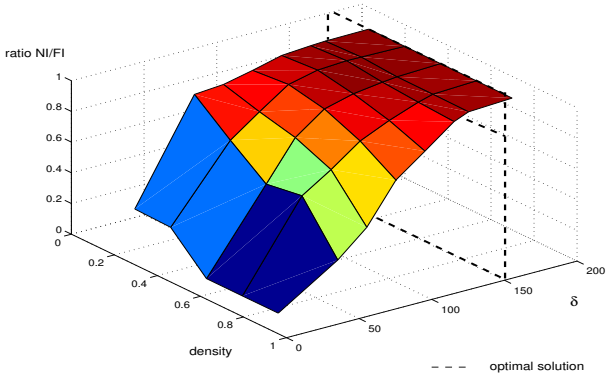

Figure 9: *Neighborhood Interchangeability relatively to Full Interchangeability for $^{\delta}$interchangeability in Weighted CSP.*

Conclusions and Future Work

In this paper we presented a java based implementation able to represent soft CSPs and to compute interchangeability. By using this implementation, we have studied the occurrence of $_{\alpha}NI$ and $^{\delta}NI$ and we have evaluated how this occurrence can depend on the values of α and δ, and how local NI relies with FI. The experimental facts show also that there is high occurrence of $_{\alpha}NI$ and $^{\delta}NI$ interchangeability around optimal solutions in Fuzzy CSP and weighted CSPs. Thus, these algorithms can be used successfully in solution update applications. Moreover, we also show that NI interchangeability can well approximate FI interchangeability. The use of these technique to improve search have to be investigated.

We believe that the results prove the reliability for using $^{\delta}_{\alpha}$interchangeability for solution updating and motivate for further studying. By now, the soft CSP module implements only a basic solver which combines all the constraints of the CSP and then projecting them over the variables of interest. We leave for further work implementation of more performant solvers as the main goal of this study is analyse the occurrence and behaviour of interchangeability in soft CSP.

References

Benson, B., and Freuder, E. 1992. Interchangeability preprocessing can improve forward checking search. In *Proc. of the 10th ECAI*.

Bistarelli, S.; Faltings, B.; and Neagu, N. 2002. A de£nition of interchangeability for soft csps. In *Proc. ERCIM/CologNet Workshop on Constraint - Selected Papers*, LNAI. Springer-Verlag. to appear.

Bistarelli, S.; Montanari, U.; and Rossi, F. 1997. Semiring-based Constraint Solving and Optimization. *J. ACM* 44(2).

Boi Faltings, Rainer Weigel, B. C. 1995. Abstraction by Interchangeability in Resource Allocation. In *Proc. of the 14 th IJCAI-95*, 1694–1701.

Bruchez, E., and Torrens, M. 1996. Java constraint library. http://liawww.ep¤.ch/ torrens/Project/JCL/.

Choueiry, B. Y. 1994. *Abstraction Methods for Resource Allocation*. Ph.D. Dissertation, EPFL PhD Thesis no 1292.

Dubois, D.; Fargier, H.; and Prade, H. 1993. The calculus of fuzzy restrictions as a basis for ¤exible constraint satisfaction. In *Proc. IEEE International Conference on Fuzzy Systems*.

Freuder, E., and Wallace, R. 1992. Partial constraint satisfaction. *AI Journal* 58.

Freuder, E. C. 1991. Eliminating interchangeable values in constraint satisfaction problems. In *Proc. of AAAI-91*.

Haselbock, A. 1993. Exploiting interchangeabilities in constraint satisfaction problems. In *Proc. of the 13th IJCAI*.

Neagu, N., and Faltings, B. 1999. Constraint Satisfaction for Case Adaptation. In *In Proc. of the workshop sesion (ICCBR99)*.

Ruttkay, Z. 1994. Fuzzy constraint satisfaction. In *Proc. 3rd IEEE International Conference on Fuzzy Systems*.

Schiex, T.; Fargier, H.; and Verfaille, G. 1995. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proc. IJCAI95*.

Weigel, R., and Faltings, B. 1998. Interchangeability for case adaptation in con£guration problems. In *Proc. of the AAAI98 Spring Symposium on Multimodal Reasoning, Stanford, CA*. TR SS-98-04.

Weigel, R., and Faltings, B. 1999. Compiling constraint satisfaction problems. *Arti£cial Intelligence* 115.