Experimental Evaluation of Interchangeability in Soft CSPs

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

[1] Artificial Intelligence Laboratory (LIA), Computer Science Department, EPFL
CH-1015, Ecublens, Switzerland
{boi.faltings,nicoleta.neagu}@epfl.ch
[2] Dipartimento di Scienze, Universitá "D'Annunzio"
Viale Pindaro 87, I-65127 Pescara, Italy
bista@sci.unich.it
[3] Istituto di Informatica Telematica (IIT), CNR,
Via G. Moruzzi 1, I-56124, Pisa, Italy,
Stefano.Bistarelli@iit.cnr.it

Abstract. Freuder in [9] defined interchangeability for classical Constraint Satisfaction Problems (CSPs). Recently [2], we extended the definition of interchangeability to *Soft* CSPs and we introduced two notions of relaxations based on degradation δ and on threshold α ($^{\delta}$neighborhood interchangeability ($^{\delta}NI$)and $_{\alpha}$neighborhood interchangeability ($_{\alpha}NI$)). In this paper we extend the study introduced in [11] and we analyze the presence of the relaxed version of interchangeability in random soft CSPs. We give a short 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 CSPs and weighted CSPs. Thus, these algorithms can be used successfully in solution update applications. Moreover, it is also showed that NI interchangeability can well approximate full interchangeability (FI).

1 Introduction

Interchangeability in constraint networks has been first proposed by Freuder [9] 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 [10, 16, 6] and solution adaptation [15]. 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 [8, 7, 13, 14, 3]. The soft constraints framework of c-semirings [3] 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 [2] 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, ${}^{\delta}$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 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 $\leq \alpha$.

The behaviour 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 first remind some details about Interchangeability and soft Constraint Satisfaction Problems. We give also some details about the java implementation [11] we used to compute to ${}^{\delta}/_{\alpha}$substitutability/interchangeability. Central results of the paper are described in Section 4 where the results of some tests are described. A conclusions and possible future work section conclude the paper.

2 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].

[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 $a + b = b$.

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).

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

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 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 (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 and Neighbourhood 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.

Extensivity ($NI_v(a/b) \implies FI_v(a/b)$) and transitivity ($b \in NS_v(a), a \in NS_v(c) \implies b \in NS_v(c)$) of interchangeability can be used to define an algorithm able to compute a subset of the interchangeabilities. When the times operator of the semiring is idempotent the algorithm , instead of considering the combination of all the constraint C_v involving a certain variable v, can check interchangeability on each constraint itself [2] giving raise to a low complexity bound.

Algorithm 1 shows the algorithm that can be used to find domain values that are Neighbourhood Interchangeable. It uses a data structure similar to

1: Create the root of the discrimination tree for variable v_i
2: Let $C_{v_i} = \{c \in C \mid v_i \in supp(c)\}$
3: Let $D_{v_i} = \{$the set of domain values d_{v_i} for variable $v_i\}$
4: **for all** $d_{v_i} \in D_{v_i}$ **do**
5: **for all** $c \in C_v$ **do**
6: execute Algorithm NI-Nodes(c, v, d_{v_i}) to build the nodes associated with c
7: Go back to the root of the discrimination tree.

Algorithm 1: Algorithm to compute neighbourhood interchangeable sets for variable v_i.

the *discrimination trees*, first introduced by Freuder in [9] . Algorithm 1 can compute different versions of neighbourhood interchangeability depending on the algorithm $NI - nodes$ used. Algorithm 2 shows the simplest version without threshold or degradation.

1: **for all** assignments η_c to variables in $supp(c)$ **do**
2: compute the semiring level $\beta = c\eta_c[v_i := d_{v_i}]$,
3: **if** there exists a child node corresponding to $\langle c = \eta_c, \beta\rangle$ **then**
4: move to it,
5: **else**
6: construct such a node and move to it.
7: Add $v_i, \{d_{v_i}\}$ to annotation of the last build node,

Algorithm 2: NI-Nodes(c, v, d_{v_i}) for Soft-NI.

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 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). By extending the previous definitions 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, $\bigotimes C\eta[v := a] \geq \alpha \implies \bigotimes C\eta[v := a] \leq_S \bigotimes C\eta[v := b]$.

3 The Java Implementation

We implemented the soft CSP module and the interchangeability module as an extension of the JCL, developed at the Artificial Intelligence Laboratory (EPFL) [5]. JCL is implemented in java which ensures portability on all the platforms. The library contains a CSP package describing constraint satisfaction problems and several solvers. In this section, we describe briefly the JCL and the 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 first one models and solve soft constraint satisfaction problems; the second one computes interchangeabilities for crisp and soft CSPs, see Figure 1.

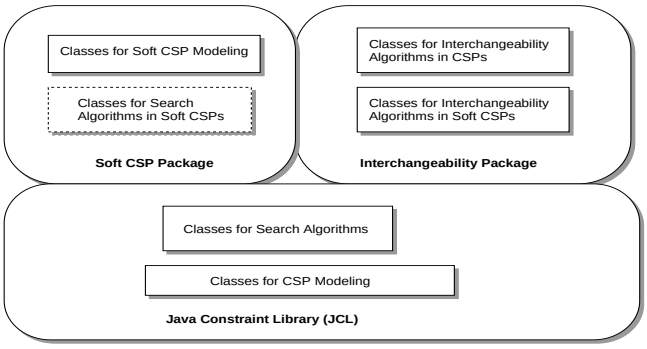


Fig. 1. *Soft CSP and Interchangeability Modules on top of Java Constraint Language(JCL).*

The Soft_CSP Module. The Soft_CSP package extends the CSP class in order to support softness. We implemented the scheme from [3] 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 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. The semiring class parameterizes the type of the CSP and the respective operations of constraints combinations (and projection).

The Interchangeability Module. This module implements all the algorithms for computing classical interchangeability for all the semiring types. It provides also the computational classes for degradation δ and/or threshold α interchangeability which finds ${}^{\delta}/_{\alpha}NI$ sets.

4 Test Evaluation

Occurrence of *NI* in classical CSP have been already studied to improve search [1], for resource allocation application [4] and for configuration problems [12]. 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 CSPs representing the important classes 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 [3].

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 $constraint\text{-}number \times domainsize$).

For both fuzzy and weighted CSPs we observed that the density and number of variables do not influence too much the 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 have instead found that the CSP tightness influence the occurrence of interchangeable values.

In the following we use the model of measuring *NI* sets developed in [4] with some adaptation needed in order to deal with softness. We report here the results for problem sizes n = 10, while varying the density $dens - csp \in$

$\{0.1, 0.2, \ldots, 0.9\}$, the tightness $tightness - csp \in \{0.1, 0.2, \ldots, 0.9\}$ and the maximum domain size $dom - size = \{\frac{n}{10}, \frac{2n}{10}, \ldots, \frac{9n}{10}, n\}$. The semiring values are generated uniformly and accordingly to the CSP tightness. For each case, ten random problems were generated and then graphically represented by considering the measures $measure_{\alpha}NI$ and $measure^{\delta}NI$ described below.

In all the graphs we highlight where is the position of the optimal solution. In fact, when dealing with crisp CSP there is no 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.

4.1 Fuzzy CSPs

Fuzzy CSPs are a representative example of soft CSP based on an idempotent times semiring operator. Informally, for fuzzy CSPs the weights assigned to the tuples represents how much the tuple is satisfied, and the semiring operations are min for combination and max for projection.

$\delta/\alpha NI$ Occurrency in Fuzzy CSPs

$measure_{\alpha}NI$ measures the "occurrence" of NI $_{\alpha}$interchangeable value pairs in the sense that it computes the average number of $_{\alpha}NI$ interchangeable pairs values over the whole CSP divided by the potential number of relations using the formula:

$$measure_{\alpha}NI = \frac{\sum_{k=1}^{n} \frac{\alpha NIV_k * 2}{domSize_{V_k} * (domSize_{V_k} - 1)}}{n}.$$

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

Similarly, $measure_{\delta}NI$ measures the "occurrence" of NI $^{\delta}$interchangeable values:

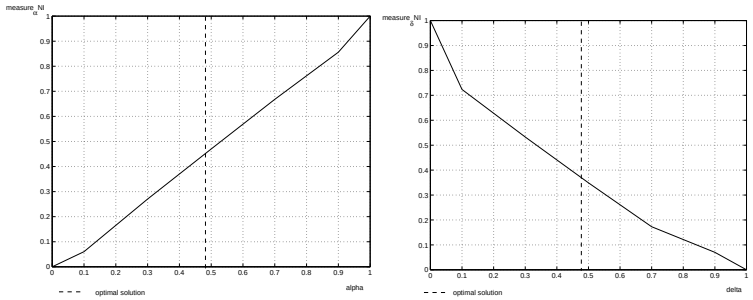
$$measure_{\delta}NI = \frac{\sum_{k=1}^{n} \frac{\delta NIV_k * 2}{domSize_{V_k} * (domSize_{V_k} - 1)}}{n} -$$

As before, n represents the problem size and $^{\delta}NIV_k$ all the $^{\delta}$interchangeable pairs values for variable V_k.

In Figure 2(a), we represent $measure_{\alpha}NI$ varying the value of α. We found that there are a lot of $_{\alpha}$interchangeable values close to the optimal solution. We also notice that the interchangeabilities decrease when using lower values for α.

Similarly, in Figure 2(b), we observe also for δ the same results: *close to the optimum we have many* $^{\delta}/_{\alpha}$*substitutability/interchangeability.*

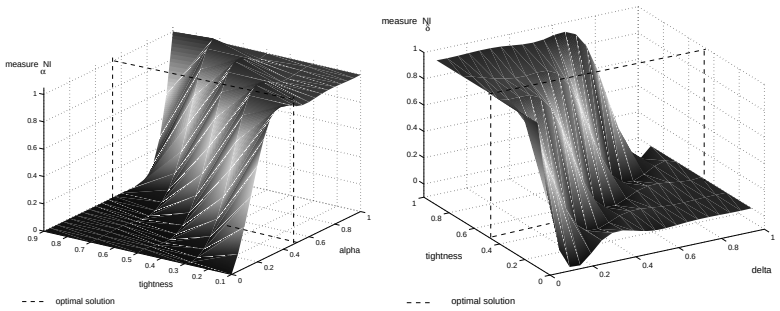
In the Figure 3, we represent how the occurrence of ($^{\delta}/_{\alpha}$)interchangeability depends on α and δ (respectively in Figure (a) and (b)), and also on the problem tightness.



(a) (b)

Fig. 2. A statistical measure of the number of NI of α/δ interchangeability for Fuzzy CSP with uniform weight distribution, for sets of problems with 10 variables and 10 values domains (Figure (a) for α and Figure (b) for δ).

As we can see in the Figure 3(a), the number of α interchangeable values depend on α, but also on the problem tightness. For low tightness, the number of interchangeabilities increases faster, while for higher values of tightness interchangeable values appear only for high values of α.

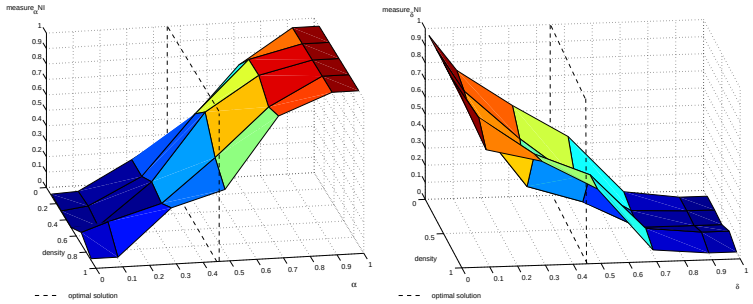


(a) (b)

Fig. 3. A statistical measure of the number of α/δ interchangeability for Fuzzy CSP, with uniform weight distribution, for sets of problems with 10 variables and 10 values domains and varying the problem tightness.

In Figure 3(b), we show the dependence w.r.t. δ and the problem tightness. We can see that the occurrence of interchangeable values increases with the tightness for low δ values.

In Figure 4, we perform the same analysis but varying this time the density of the problem. We can notice that the interchangeability occurence does not vary too much with problem density (the shape is in fact very regular).

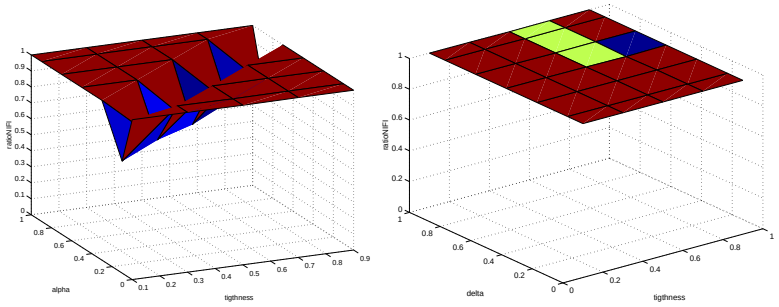


(a) (b)

Fig. 4. A statistical measure of the number of α/δ interchangeability for Fuzzy CSPs with uniform weight distribution, for sets of problems with 10 variables and 10 values domains and varying problem density.

Estimation of NI versus FI for Fuzzy CSPs

Computing full interchangeable values might be a quite costly operation as it may require computing all the solutions. There are not known efficient algorithms which can compute in polynomial time full interchangeable values. Neighbourhood interchangeability can be computed in polynomial time, but provide only a subset of full interchangeable values. In the following we study how neighbourhood interchangeability can approximate full interchangeability in soft CSPs.



(a) (b)

Fig. 5. The ratio between the number of neighbourhood and full interchangeable values varying tightness and α/δ (respectively on the left and right side of the picture), for problems of 10 variables with 10 domain values.

We consider in Figure 5 the ratio between the number of neighbourhood interchangeabilities and the number of full interchangeabilities. The value of

$ratioNIFI$ is computed as

$$ratioNIFI = \frac{\sum_{k=1}^{n} \delta NIV_k}{\sum_{k=1}^{n} \delta FIV_k}$$

for δ interchangeability, and in a similar manner also for α. In the formula, ${}^{\delta}NIV_k$ 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 Figure 5, we see that the ratios ${}^{\delta}NI/{}^{\delta}FI$ and ${}_{\alpha}NI/{}_{\alpha}FI$ are always between 0.7 and 0.9 when we consider domain values close to the optimal solution. Thus, NI interchangeability can well approximate FI interchangeability.

Estimation of NI computed by general definition versus NI computed with the Discrimination Tree algorithm for Fuzzy CSPs

For soft CSPs based on idempotent semiring as Fuzzy CSP, we can use the proposed Discrimination Tree algorithm for computing $({}^{\delta}/_{\alpha})$neighbourhood interchangeability. In this paragraph we study how much the number of interchangeability values found with Discrimination Tree algorithm can approximate the number of interchangeability values detected with the definition algorithm and full interchangeability values respectively.

In Figure 6(a) we can see how ${}_{\alpha}$ full interchangeability, ${}_{\alpha}$ neighbourhood interchangeability computed using the definition algorithm and discrimination tree algorithm respectively vary with α. The results show that the Discrimination Tree general algorithm finds a high number of interchangeable values.

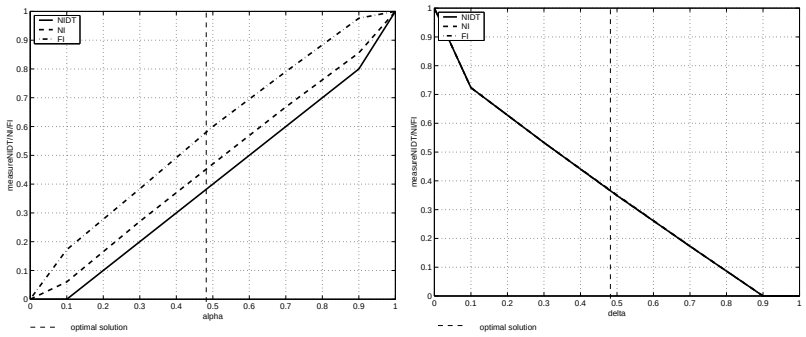


Fig. 6. How the number of α NI, α FI, α NIDT values are varying with α and δ.

Figure 6(b) represents the analysis for ${}^{\delta}$interchangeability. We can see that the graphs almost overlaps. Thus the number of interchangeability values found for ${}^{\delta}$neighbourhood/full interchangeability is almost the same. Thus, in the δ

case the number of interchangeable values found by the Discrimination Tree algorithm is very close or almost the same to the original number found by the definition algorithm.

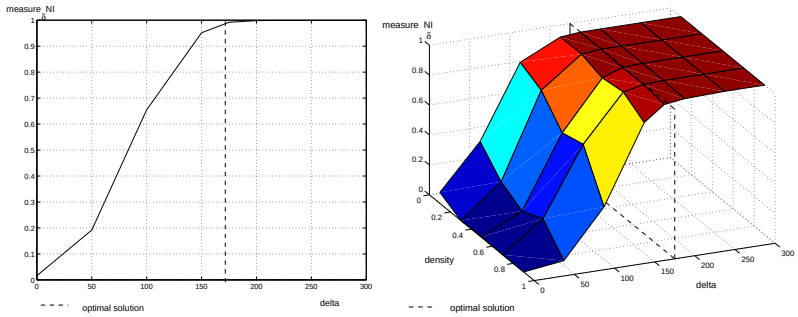
Following these results we can get to the conclusion that full interchangeability in Fuzzy CSPs, and thus for any idempotent soft CSP, can be well approximated by neighbourhood interchangeability computed either with the definition algorithm or with the Discrimination Tree algorithm.

4.2 Weighted CSPs

Weighted CSP represents the important class of Soft CSPs dealing with a non-idempotent times operator. In the following, we evaluate how the neighbourhood interchangeability can approximate full interchangeability.

δNI Occurrency in Weighted CSPs

We present how the occurrence of $^{\delta}$ neighbourhood interchangeable values varies with δ. The tests are not done for $_{\alpha}NI$ because when times is not idempotent we cannot easily compare local and global notions of α interchageabilities.



(a) (b)

Fig. 7. How the number of interchageabilities varies with δ and with CSP density in weighted CSPs.

In Figure 7(a), we see how the number of $^{\delta}$neighbourhood interchangeable increases with δ and how, close to the optimal solution approaches to 1. This means that all the values pairs are interchangeable for high δ (as could be eaily guessed).

In Figure 7(b), we represent how the measure of NI varies w.r.t. δ and the density as well. We can see, as in the Fuzzy case, that the SCSP density does not influence the occurrence of interchangeability too much.

Estimation of NI versus FI for Weighted CSPs

In Figure 8, we can see how the number of neighbourhood and full interchangeability values vary with δ. We can see that the number of neighbourhood and full interchangeability does not differ too much. Thus, we can again approximate full interchangeability with neighbourhood interchangeability.

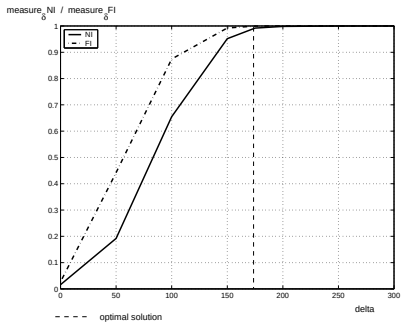


Fig. 8. δNI versus δFI values varying δ for weighted CSPs.

As in the tests for fuzzy CSPs, we computed the ratio $ratioNIFI$, and in Figure 9 we display the obtained results. We can see how the ratio between ${}^{\delta}NI$ and ${}^{\delta}FI$ varies with delta and CSP density. The ratio is always between 0.8 and 1 and this lead us to the conclusion that neighbourhood interchangeability can approximate fairly full interchangeability for weighted CSP if we can accept a degradation of δ.

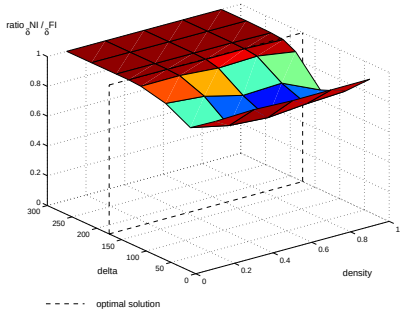


Fig. 9. The figure represents how ratio δNI/δFI is varying with δ and CSP density for weighted CSPs.

5 Conclusions and Future Work

Interchangeability is an important technique as it can be used as a preprocessing technique for finding equivalences among variables values, and thus enhancing search. It can also be used as a postprocessing technique for solution adaptation. In this paper we used our java based implementation to compute interchangeability for Soft CSPs. 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, on the CSP parameters and also how local NI relies with FI. The experimental facts show also that there is high occurrence of ${}_{\alpha}NI$ and ${}^{\delta}NI$ interchangeability close to the optimal solutions in fuzzy CSPs and weighted CSPs as well. Thus, these algorithms can be used successfully in solution update applications. Moreover, we showed that NI interchangeability can well approximate FI interchangeability. We studied also how the discrimination tree algorithm can be used to compute a subset of interchangeabilities.

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.

References

1. B.W. Benson and E. Freuder. Interchangeability preprocessing can improve forward checking search. In *Proc. of the 10th ECAI*, 1992.
2. S. Bistarelli, B. Faltings, and N. Neagu. A definition of interchangeability for soft csps. In *Proc. ERCIM/CologNet Workshop on Constraint - Selected Papers*, LNAI. Springer-Verlag, 2002. to appear.
3. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *J. ACM*, 44(2), 1997.
4. Berthe Choueiry Boi Faltings, Rainer Weigel. Abstraction by Interchangeability in Resource Allocation. In *Proc. of the 14 th IJCAI-95*, pages 1694–1701, Montreal, Canada, 1995.
5. Eric Bruchez and Marc Torrens. Java constraint library. http://liawww.epfl.ch/ torrens/Project/JCL/, 1996.
6. Berthe Y. Choueiry. *Abstraction Methods for Resource Allocation*. PhD thesis, EPFL PhD Thesis no 1292, 1994.
7. D. Dubois, H. Fargier, and H. Prade. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In *Proc. IEEE International Conference on Fuzzy Systems*, 1993.
8. E.C. Freuder and R.J. Wallace. Partial constraint satisfaction. *AI Journal*, 58, 1992.
9. Eugene C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proc. of AAAI-91*, 1991.
10. A. Haselbock. Exploiting interchangeabilities in constraint satisfaction problems. In *Proc. of the 13th IJCAI*, 1993.
11. N. Neagu, S. Bistarelli, and B. Faltings. On the computation of local interchangeability in soft constraint satisfaction problems. In *Proc. of the 16th International FLAIRS Conference - Special Track on Constraint Solving and Programming*, St. Augustine, Florida, USA, 2003. AAAI Press.

12. Nicoleta Neagu and Boi Faltings. Constraint Satisfaction for Case Adaptation. In *In Proc. of the workshop sesion (ICCBR99)*, 1999.
13. Zs. Ruttkay. Fuzzy constraint satisfaction. In *Proc. 3rd IEEE International Conference on Fuzzy Systems*, 1994.
14. T. Schiex, H. Fargier, and G. Verfaille. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proc. IJCAI95*, 1995.
15. R. Weigel and B. Faltings. Interchangeability for case adaptation in configuration problems. In *Proc. of the AAAI98 Spring Symposium on Multimodal Reasoning, Stanford, CA*, 1998. TR SS-98-04.
16. R. Weigel and B. Faltings. Compiling constraint satisfaction problems. *Artificial Intelligence*, 115, 1999.