WCSP Integration of Soft Neighborhood Substitutability

Christophe Lecoutre, Olivier Roussel, Djamel-Eddine Dehani

WCSP Integration of Soft Neighborhood Substitutability

Christophe Lecoutre, Olivier Roussel, and Djamel E. Dehani

Université Lille-Nord de France, Artois
CRIL-CNRS UMR 8188
F-62307 Lens
{lecoutre,roussel,dehani}@cril.fr

Abstract. WCSP is an optimization problem for which many forms of soft local (arc) consistencies have been proposed such as, for example, existential directional arc consistency (EDAC) and virtual arc consistency (VAC). In this paper, we adopt a different perspective by revisiting the well-known property of (soft) substitutability. First, we provide a clear picture of the relationships existing between soft neighborhood substitutability (SNS) and a tractable property called $pcost$ which allows us to compare the cost of two values (through the use of so-called cost pairs). We prove that under certain assumptions, $pcost$ is equivalent to SNS but weaker than SNS in the general case since we show that SNS is coNP-hard. We also show that SNS preserves the property VAC but not the property EDAC. Finally, we introduce an algorithm to enforce $pcost$ that benefits from several optimizations (early breaks, residues, timestamping). The practical interest of maintaining $pcost$ together with AC*, FDAC or EDAC, during search. is shown on various series of WCSP instances.

1 Introduction

The Valued Constraint Satisfaction Problem (VCSP) [18] is a general optimization framework used to handle soft constraints, which has been successfully applied to many applications in Artificial Intelligence and Operations Research. A problem instance is modeled in this framework by means of a set of variables and a set of cost functions defined over a valuation structure. Each cost function determines a violation degree for each possible instantiation of a subset of variables. These degrees (or costs) can then be combined using the operator $\oplus$ of the valuation structure in order to obtain the overall cost of any complete instantiation. One can broadly classify the different VCSP instantiations according to the properties of the operator $\oplus$: those where $\oplus$ is idempotent (e.g., min) and those where $\oplus$ is monotonic (e.g., +).

Interchangeability is a general property of constraint networks introduced in [10]. Two values a and b for a variable x are interchangeable if for every solution I where x is assigned b, $I_{x=a}$ is also a solution, where $I_{x=a}$ means I with x set to a. Full interchangeability has been refined into several weaker forms such as neighborhood interchangeability, k-interchangeability, partial interchangeability, relational interchangeability and substitutability. Interchangeability and substitutability have been used in many contexts; see e.g. [11, 3, 7, 14]. A partial taxonomy of these two properties can be found in [12].

A generalization of interchangeability and substitutability for soft constraints has been given in [1]. For example, a value a for a variable x is soft substitutable for another value b in the domain of x if for every complete instantiation I involving (x, a), the cost of I is less than or equal to the cost of $I_{x=b}$. Observation of this property can be used to delete value b for x whereas preserving optimality. Identifying full substitutability is not tractable, but is is known [1] that neighborhood substitutability, a limited form of substitutability where only constraints involving a given variable are considered, can be computed in polynomial time when $\oplus$ is idempotent (provided that the arity of constraints be bounded). However, when $\oplus$ is monotonic, as it is the case for the VCSP specialization called WCSP (Weighted CSP), there is no clear picture although dominance rules related to soft neighborhood substitutability have been used [13, 8].

In this paper, we focus on soft neighborhood substitutability (SNS) for WCSP. We introduce a property based on cost pairs, called $pcost$, that allows us to identify efficiently soft substitutable values. We prove that under certain assumptions, $pcost$ is equivalent to SNS. However, in the general case, and especially when the WCSP forbidden cost k is not ∞, $pcost$ is weaker than SNS. Actually, identifying a soft neighborhood value is coNP-hard. We also study the relationships between SNS and known soft arc consistency properties such as Existential Directional Arc Consistency (EDAC) [9] and Virtual Arc Consistency (VAC) [5]. We prove that SNS preserves VAC but not necessarily EDAC. Finally, we develop a $pcost$ algorithm that benefits from a very moderate best-case time complexity, and we show experimentally that it can be successfully combined with EDAC during search.

2 Technical Background

A *constraint network* (CN) P is a pair $(\mathscr{X}, \mathscr{C})$ where $\mathscr{X}$ is a finite set of n variables, also denoted by $vars(P)$, and $\mathscr{C}$ is a finite set of e constraints. Each variable x has a (current) domain, denoted by $dom(x)$, which is the finite set of values that can be (currently) assigned to x; the initial domain of x is denoted by $dom^{init}(x)$. The largest domain size will be denoted by d. Each constraint c_S involves an ordered set S of variables, called the *scope* of c_S, and represents a relation capturing the set of tuples allowed for the variables in S. A *unary* (resp., *binary*) constraint involves 1 (resp., 2) variable(s), and a *non-binary* one strictly more than 2 variables. An *instantiation* I of a set $X = \{x_1, \ldots, x_p\}$ of p variables is a set $\{(x_1, a_1), \ldots, (x_p, a_p)\}$ such that $\forall i \in 1..p, a_i \in dom^{init}(x_i)$; X is denoted by $vars(I)$ and each a_i is denoted by $I[x_i]$. An instantiation I *on* a CN P is an instantiation of a set $X \subseteq vars(P)$; it is *complete* if $vars(I) = vars(P)$. I is *valid* on P iff $\forall (x, a) \in I, a \in dom(x)$. I *covers* a constraint c_S iff $S \subseteq vars(I)$. I *satisfies* a constraint c_S with $S = \{x_1, \ldots, x_r\}$ iff (i) I covers c_S and (ii) the tuple $(I[x_1], \ldots, I[x_r]) \in c_S$. An instantiation I on a CN P is *locally consistent* iff (i) I is valid on P and (ii) every constraint of P covered by I is satisfied by I. A *solution* of P is a complete locally consistent instantiation on P; $sols(P)$ denotes the set of solutions of P.

A *weighted constraint network* (WCN) P is a triplet $(\mathscr{X}, \mathscr{W}, k)$ where $\mathscr{X}$ is a finite set of n variables, as for CSP, $\mathscr{W}$ is a finite set of e weighted constraints, also denoted by $cons(P)$, and $k > 0$ is a natural integer or ∞. Each weighted constraint

$w_S \in \mathscr{W}$ involves an ordered set S of variables (its scope) and is defined as a cost function from $l(S)$ to $\{0, \ldots, k\}$ where $l(S)$ is the set of possible instantiations of S. When a constraint w_S assigns the cost k to an instantiation of S, it means that w_S forbids this instantiation. Otherwise, it is permitted with the corresponding cost (0 is completely satisfactory). Costs are combined with the specific operators $\oplus$ defined as: $\forall a, b \in \{0, \ldots, k\}, a \oplus b = min(k, a + b)$.

For any instantiation I and any set of variables X, let $I_{\downarrow X} = \{(x, a) \mid (x, a) \in I \wedge x \in X\}$ be the projection of I on X. We denote by $I_{x=a}$ the instantiation $I_{\downarrow X \setminus \{x\}} \cup \{(x, a)\}$, which is the instantiation obtained from I either by replacing the value assigned to x in I by a, or by extending I with (x, a). The set of neighbor constraints of x is denoted by $\Gamma(x) = \{c_S \in cons(P) \mid x \in S\}$. When $\Gamma(x)$ does not contain two constraints sharing at least two variables, we say that $\Gamma(x)$ is *separable*. If c_S is a (weighted) constraint and I is an instantiation of a set $X \supseteq S$, then $c_S(I)$ will be considered to be equal to $c_S(I_{\downarrow S})$ (in other words, projections will be implicit). If C is a set of constraints, then $vars(C) = \cup_{c_S} S$ is the set of variables involved in C; if I is an instantiation such that $vars(C) \subseteq vars(I)$, then $cost_C(I) = \oplus_{c_S \in C} c_S(I)$ is the cost of I obtained by considering all constraints in C. For a WCN P and a complete instantiation I of P, the cost of I is then $cost_{cons(P)}(I)$ which will be simplified into $cost(I)$. The usual (NP-hard) task of Weighted Constraint Satisfaction Problem (WCSP) is, for a given WCN, to find a complete instantiation with a minimal cost. CSP can be seen as specialization of WCSP (when only costs 0 and k are used) and WCSP [16] can be seen as a specialization of the generic framework of valued/semiring-based constraints [2].

Many forms of soft arc consistency have been proposed during the last decade. We briefly introduce them in the context of binary WCNs. Without any loss of generality, the existence of a zero-arity constraint $c_\emptyset$ (a constant) as well as the presence of a unary constraint c_x for every variable x is assumed. A variable x is node-consistent (NC*) iff $\forall a \in dom(x), c_\emptyset \oplus c_x(a) < k$ and $\exists b \in dom(x) \mid c_x(b) = 0$. A variable x is arc-consistent (AC*) iff x is NC* and $\forall a \in dom(x), \forall c_{xy} \in \Gamma(x), \exists b \in dom(y) \mid c_{xy}(a, b) = 0$ (b is called a simple support of a). A WCN is AC* iff each of its variable is AC* [15, 16]. A WCN is full directional arc-consistent (FDAC) [4] with respect to an order $<$ on the variables if it is AC* and $\forall c_{xy} \mid x < y, \forall a \in dom(x), \exists b \in dom(y) \mid c_{xy}(a, b) = c_y(b) = 0$ (b is called a full support of a). A WCN is existential arc-consistent (EAC) [9] if it is NC* and $\forall x \in vars(P), \exists a \in dom(x) \mid c_x(a) = 0 \wedge \forall c_{xy} \in \Gamma(x), \exists b \in dom(y) \mid c_{xy}(a, b) = c_y(b) = 0$ (a is called the existential support of x). A WCN is existential directional arc-consistent (EDAC) with respect to an order $<$ on the variables if it is EAC and FDAC with respect to $<$. For any WCN P, we can derive an associated CN $Bool(P)$ by considering constraint relations that only allow tuples with cost zero in P. P is virtual arc-consistent (VAC) [5] if the arc consistency closure of $Bool(P)$ does not involve a variable with an empty domain, denoted by $AC(Bool(P)) \neq \bot$. A WCN is optimal soft arc-consistent (OSAC) if no SAC transformation (see [6]) applied to it increases $c_\emptyset$. OSAC is stronger than VAC, which itself is stronger than EDAC, when comparing the values of $c_\emptyset$. We shall note $\phi(P)$ the enforcement of property ϕ (e.g., AC, EDAC, ...) on the (W)CN P.

3 Soft Substitutability

In this section, we introduce soft neighborhood substitutability. Initially, *interchangeability* and *substitutability* are properties that have been introduced for CSP [10]. From now on, we consider given a (W)CN P.

Definition 1. *Let $x \in vars(P)$ and $\{a, b\} \subseteq dom(x)$,*

- *(x, a) is (fully)* substitutable *for (x, b) on P iff for every solution $I_{x=b}$ of P, $I_{x=a}$ is also a solution of P;*
- *(x, a) is (fully)* interchangeable *with (x, b) on P iff (x, a) is substitutable for (x, b) and (x, b) is substitutable for (x, a).*

For example, consider a CN P such that $vars(P) = \{x, y, z\}$ and $sols(P) = \{(a, a, a), (a, b, b), (b, a, a), (c, a, a), (c, b, b)\}$. The values (x, a) and (x, c) are interchangeable and both are substitutable for (x, b). When only a single solution is sought, we can remove a value that is interchangeable with another value (or for which a value is substitutable). Such removal preserves the satisfiability of the problem instance but not the full set of solutions.

From now on, we shall focus on substitutability that has been generalized [1] for WCSP as follows:

Definition 2. *Let $x \in vars(P)$ and $\{a, b\} \subseteq dom(x)$, (x, a) is* soft substitutable *for (x, b) on P iff for every complete instantiation I of P, $cost(I_{x=a}) \leq cost(I_{x=b})$.*

When (x, a) is soft substitutable for (x, b), b can be removed from $dom(x)$ without changing the cost of the optimal solution(s) of P. Indeed, possible solutions of P with (x, b) are lost, but it is guaranteed that solutions with (x, a) are at least as good.

Because identifying substitutable values involves handling complete instantiations, this is subject to combinatorial explosion. However, there is a form of local substitutability, called neighborhood substitutability [10, 1], that may be helpful.

Definition 3. *Let $x \in vars(P)$ and $\{a, b\} \subseteq dom(x)$, (x, a) is* soft neighborhood substitutable *for (x, b) on P iff for every complete instantiation I of P, $cost_{\Gamma(x)}(I_{x=a}) \leq cost_{\Gamma(x)}(I_{x=b})$.*

We shall say that (x, b) is SNS-eliminable (on P) when there exists a value (x, a) such that (x, a) is soft neighborhood substitutable for (x, b). It is rather immediate that soft neighborhood substitutability implies soft (full) substitutability (but the reverse is not true). Interestingly enough, soft (neighborhood) substitutability allows compensation between constraint costs. Such compensation is made possible by the presence of all intermediate costs in the valuation structure, that is to say, the costs different from 0 and k. A simple illustration is given by Figure 1. There are two binary constraints c_{xy} and c_{xz} and three trivial unary constraints (all unary costs are equal to 0). Binary costs are depicted as labeled edges, and zero costs are not shown. Note that (x, a) is soft substitutable for (x, b).

Definition 3 requires to consider each instantiation of $vars(\Gamma(x))$, which has a high computational cost. Considering each constraint individually allows to reduce this cost,

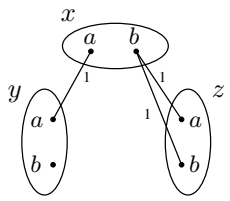

Fig. 1. (x, a) is soft substitutable for (x, b)

but in this case, it is highly desirable to be able to identify cost compensations between constraints. One straightforward way to do this is to compute a sum of minimal cost differences over all constraints, as mentioned in [13, 8]. Unfortunately, these differences of costs introduce subtle problems when $k \neq \infty$. This is illustrated as follows:

Example 1. Consider the two families of constraints $C_i = \{c_i \mid i \in 1..n\}$ and $C'_i = \{c'_i \mid i \in 1..n'\}$ defined as:

x	y_i	c_i
a	c	0
b	c	1

x	z_i	c'_i
a	d	1
b	d	0

When $n = k$ and $n' = k+1$, (x, a) and (x, b) are interchangeable because both are forbidden (i.e., have maximum cost k). However, $\forall c_i \in C_i, cost_{c_i}(I_{x=b}) - cost_{c_i}(I_{x=a}) = 1$ and $\forall c'_i \in C'_i, cost_{c'_i}(I_{x=b}) - cost_{c'_i}(I_{x=a}) = -1$. In the summation over the constraints in $\Gamma(x) = C_i \cup C'_i$, the resulting value is -1 which would indicate that b has globally a lower cost than a, which is false since both a and b have a cost of k. To identify correctly this case of substitution when $k \neq \infty$, it is necessary to use a non commutative operator, which prevents us from using the usual -.

4 Computing Soft Substitutability

We now focus on soft neighborhood substitutability, and more precisely on the complexity of identifying SNS-eliminable values. We start with some related work. For the general framework VCSP, efficient algorithms for computing neighborhood substitutability exist [1] when the aggregation operator of the VCSP valuation structure is idempotent. For the framework FCSP (Fuzzy CSP), the notion of fuzzy neighborhood substitutability is proposed in [4]: it is shown that fuzzy neighborhood substitutable values can be identified efficiently when the aggregation operator of the FCSP valuation structure is strictly monotonic or when it is the operator max. More recently, the possibility of computing dominance forms weaker than soft neighborhood substitutability has been proposed in [13]. However, no qualitative study was led. This is what we propose now for WCSP.

First, we introduce cost pairs as our basic computation mechanism (this is related to what has been proposed in [4] for FCSP). Indeed, one way to circumvent the aforementioned problems with subtraction is to use only addition defined on pairs of costs. This method is analogous to the construction of integers as equivalence classes of

ordered pairs of natural numbers where a pair (β, α) represents the integer $\beta - \alpha$. We define $+$ (addition) on pairs of costs by $(\beta, \alpha) + (\beta', \alpha') = (\beta + \beta', \alpha + \alpha')$ (this is the regular + and not $\oplus$) and the comparison of a pair of costs with zero by $(\beta, \alpha) \geq 0 \Leftrightarrow \beta \geq \alpha$. Pairs are ordered by the relation $\leq$ defined as $(\beta, \alpha) \leq (\beta', \alpha') \Leftrightarrow \beta - \alpha < \beta' - \alpha' \vee (\beta - \alpha = \beta' - \alpha' \wedge \alpha < \alpha')$. In a sense, the pair (β, α) conveys the difference $\beta - \alpha$ but also the information $min(\beta, \alpha)$ which is lost when a simple subtraction is used. Computing cost pairs on each constraint separately is sufficient for identifying certain soft neighborhood substitutable values: it suffices to reason from (sum) minimum differences of costs.

Definition 4. *Let $x \in vars(P)$ and $\{a, b\} \subseteq dom(x)$,*

- *the* cost pair *of (x, b) w.r.t. (x, a) on $c_S \in \Gamma(x)$ is defined as $pcost(c_S, x : a \rightarrow b) = min_{I \in l(S)}\{(c_S(I_{x=b}), c_S(I_{x=a}))\}$;*
- *the* cost pair *of (x, b) with respect to (x, a) on P is defined as $pcost(x : a \rightarrow b) = \sum_{c_S \in \Gamma(x)} pcost(c_S, x : a \rightarrow b)$.*

Proposition 1. *Let $x \in vars(P)$ and $\{a, b\} \subseteq dom(x)$. If $pcost(x : a \rightarrow b) \geq 0$ then (x, a) is soft neighborhood substitutable for (x, b) on P.*

Proof. For a constraint c_S, let I^{c_s} be the instantiation of $S - \{x\}$ which yields the minimal cost pair in $min_{I \in l(S)}\{(c_S(I_{x=b}), c_S(I_{x=a}))\}$. By definition, $pcost(c_S, x : a \rightarrow b) = (c_S(I^{c_S}_{x=b}), c_S(I^{c_S}_{x=a}))$. By definition of min on cost pairs, we have $\forall I, \forall c_S \in \Gamma(x)$, $(c_S(I_{x=b}), c_S(I_{x=a})) \geq (c_S(I^{c_S}_{x=b}), c_S(I^{c_S}_{x=a}))$. By summing up, we obtain $\forall I$, $\sum_{c_S \in \Gamma(x)}(c_S(I_{x=b}), c_S(I_{x=a})) \geq \sum_{c_S \in \Gamma(x)}(c_S(I^{c_S}_{x=b}), c_S(I^{c_S}_{x=a}))$. By hypothesis, we have $pcost(x : a \rightarrow b) \geq 0$, so $\sum_{c_S \in \Gamma(x)}(c_S(I^{c_S}_{x=b}), c_S(I^{c_S}_{x=a})) \geq 0$, and consequently $\forall I, \sum_{c_S \in \Gamma(x)}(c_S(I_{x=b}), c_S(I_{x=a})) \geq 0$. From definition of $+$ and $\leq$ on cost pairs, we can derive $\forall I, \sum_{c_S \in \Gamma(x)} c_S(I_{x=b}) \geq \sum_{c_S \in \Gamma(x)} c_S(I_{x=a})$ which implies $\forall I, min(k, \sum_{c_S \in \Gamma(x)} c_S(I_{x=b})) \geq min(k, \sum_{c_S \in \Gamma(x)} c_S(I_{x=a}))$. Since $\forall a_i \in \{0, \ldots, k\}$, $a_1 \oplus \ldots \oplus a_n = min(k, a_1 + \ldots a_n\}$, we can conclude that $\forall I, \bigoplus_{c_S \in \Gamma(x)} c_S(I_{x=b})) \geq \bigoplus_{c_S \in \Gamma(x)} c_S(I_{x=a}))$. Hence, (x, a) is soft neighborhood substitutable for (x, b) on P. □

The converse of Proposition 1 is not true in the general case. One first case where it is false is when the scope of non-binary constraints intersect on more than one variable (non separable neighborhood). In this situation, constraints cannot be considered individually.

Example 2. Let's consider four variables x, y, z t such that $dom(x) = \{a, b\}, dom(y) = \{c, d\}, dom(z) = dom(t) = \{e\}$, and two ternary constraints c_{xyz}, c_{xyt} defined by the following cost table:

x	y	z	t	c_{xyz}	c_{xyt}
a	c	e	e	1	0
b	c	e	e	0	1
a	d	e	e	0	1
b	d	e	e	1	0

It is easily seen that (x, a) is soft neighborhood substitutable for (x, b) but $pcost(c_{xyz}, x, a \to b) = pcost(c_{xyt}, x, a \to b) = (0, 1)$ and therefore $pcost(x, a \to b) = (0, 2) \not\geq 0$.

So a first condition for the converse of Proposition 1 to hold it that $\Gamma(x)$ be separable (which is the case of binary normalized networks).

Another case where the converse of Proposition 1 is false is when $k \neq \infty$. Considering the WCN of example 1 with $n = k$ and $n' = k + 1$, we can observe that $pcost(x : a \to b) = (n, n') = (k, k + 1) \not\geq 0$. However, both (x, a) and (x, b) imply cost k and therefore (x, a) is substitutable for (x, b) (and conversely). On this example, it might seem a good idea to use $\oplus$ instead of $+$ in the definition of the addition of pairs. However, Example 3 shows that this would lead to the incorrect identification of substitutable values.

Example 3. Consider the unary constraint c_x and the family of binary constraints $C_i = \{c_i \mid i \in 1..n\}$ defined by:

x	y_i	c_i
a	a	2
a	b	0
b	a	1
b	b	0

x	c_x
a	1
b	0

Clearly, (x, a) is not substitutable for (x, b). With the sum of pairs defined with $+$ and $n = k$, $pcost(x, a \to b) = (n, 2n + 1) \not\geq 0$. If the sum of pairs was defined with $\oplus$, we would obtain $(k, k) \geq 0$. Note that even if (k, k) was interpreted as $k - k = 0$ or as $k - k = k$ (absorbing element), the problem would remain: in both cases, we would have $(k, k) \geq 0$.

Interestingly, there are some situations where, even when $k \neq \infty$, using cost pairs allows us to identify exactly neighborhood substitutable values.

Proposition 2. *Let $x \in vars(P)$ and $\{a, b\} \subseteq dom(x)$ such that $\Gamma(x)$ is separable and $pcost(x, a \to b) = (\beta, \alpha)$ with $\alpha < k$. If (x, a) is soft neighborhood substitutable for (x, b) on P then $pcost(x : a \to b) \geq 0$.*

Proof. Since by hypothesis $\Gamma(x)$ is separable, one can define the instantiation I^{min} on $vars(\Gamma(x)) \setminus \{x\}$ as the union for each constraint $C_s \in \Gamma(x)$ of the instantiations I^{c_S} defined in the proof of Proposition 1. I^{min} is such that $pcost(c_S, x : a \to b) = (c_S(I^{min}_{x=b}), c_S(I^{min}_{x=a}))$.

By hypothesis, $\forall I, cost_{\Gamma(x)}(I_{x=b}) \geq cost_{\Gamma(x)}(I_{x=a})$ which can be rewritten as $\forall I, \bigoplus_{c_S \in \Gamma(x)} c_S(I_{x=b}) \geq \bigoplus_{c_S \in \Gamma(x)} c_S(I_{x=a})$. This is especially true for $I = I^{min}$ therefore $\bigoplus_{c_S \in \Gamma(x)} c_S(I^{min}_{x=b}) \geq \bigoplus_{c_S \in \Gamma(x)} c_S(I^{min}_{x=a})$, giving $min(k, \beta) \geq min(k, \alpha)$ where $\beta = \sum_{c_S \in \Gamma(x)} c_S(I^{min}_{x=b}))$ and $\alpha = \sum_{c_S \in \Gamma(x)} c_S(I^{min}_{x=a})$. By definition, $pcost(x : a \to b) = (\beta, \alpha)$ and therefore $pcost(x : a \to b) \geq 0$ iff $\beta \geq \alpha$. Now, if $\alpha < k$, $min(k, \alpha) = \alpha$ and $min(k, \beta) \geq min(k, \alpha) \Rightarrow \beta \geq \alpha \Rightarrow pcost(x : a \to b) \geq 0$ (this is true for both $\beta < k$ and $\beta \geq k$). Note that when $\alpha \geq k$, $min(k, \beta) \geq min(k, \alpha) \not\Rightarrow \beta \geq \alpha$, a counter example being $\beta = k$ and $\alpha = k + 1$. □

Corollary 1. *Let $x \in vars(P)$ and $\{a, b\} \subseteq dom(x)$ such that $\Gamma(x)$ is separable and $pcost(x : a \to b) = (\beta, \alpha)$ with $\alpha < k$. If $(\beta, \alpha) < 0$ then (x, a) is not soft neighborhood substitutable for (x, b) on P.*

When $pcost(x : a \to b) = (\beta, \alpha)$ with $\alpha \geq k$, deciding if (x, a) is soft neighborhood substitutable for (x, b) is much harder. Indeed, this problem is co-NP hard.

To prove this, we introduce the Multiple-choice Double Cost Problem (MCDP). We show that MCDP is NP-complete and exhibit a polynomial reduction of the MCDP problem to soft neighborhood substitutability.

Multiple-choice Double Cost Problem (MDCP) Given m sets E_1, E_2, ..., E_m of objects such that each object $o_j \in E_i$ has a cost value $r_{ij} \in \mathbb{Z}^+$ as well as a secondary cost value $s_{ij} \in \mathbb{Z}^+$. Given a maximal cost $C \in \mathbb{Z}^+$, the MDCP problem consists in deciding if it is possible to choose one object from each set such that the sum of the costs of these selected objects does not exceed C and does not exceed the sum of the secondary costs as well. This problem may be formulated as:

$$\sum_{i=1}^{m} \sum_{j \in E_i} r_{ij} x_{ij} \leq C,$$
$$\sum_{i=1}^{m} \sum_{j \in E_i} r_{ij} x_{ij} \leq \sum_{i=1}^{m} \sum_{j \in E_i} s_{ij} x_{ij},$$
$$\sum_{j \in E_i} x_{ij} = 1, i = 1, \ldots, m,$$
$$x_{ij} \in \{0, 1\}, i = 1, \ldots, m, j \in E_i.$$

Proposition 3. *The multiple-choice double cost problem is NP-complete.*

Proof. Membership to NP is immediate. For NP-hardness, we reduce Multiple-choice Knapsack problem (MCKP known to be NP-hard [17]) to Multiple-choice Double Cost Problem. For MCKP, we have also m sets, and each object is given a profit $p_{ij} \in \mathbb{Z}^+$ as well as a weight $w_{ij} \in \mathbb{Z}^+$. Given a minimal profit $P \in \mathbb{Z}^+$ and a maximal weight $W \in \mathbb{Z}^+$, the MCKP decision problem is formulated as:

$$\sum_{i=1}^{m} \sum_{j \in E_i} w_{ij} x_{ij} \leq W,$$
$$\sum_{i=1}^{m} \sum_{j \in E_i} p_{ij} x_{ij} \geq P,$$
$$\sum_{j \in E_i} x_{ij} = 1, i = 1, \ldots, m,$$
$$x_{ij} \in \{0, 1\}, i = 1, \ldots, m, j \in E_i.$$

To encode a MCKP instance into a MDCP instance, we keep the same structure (sets) and define:

$$C = qW - mP,$$
$$r_{ij} = qw_{ij} - P, i = 1, \ldots, m, j \in E_i,$$
$$s_{ij} = mp_{ij} + r_{ij} - P, i = 1, \ldots, m, j \in E_i.$$

with $q = 2mP$. With such a value for q, one can show that all values C, r_{ij} and s_{ij} belong to $\mathbb{Z}^+$. The first MDCP equation can be transformed as follows:
$\sum_{i=1}^{m} \sum_{j \in E_i} r_{ij} x_{ij} \leq C$
$\Rightarrow \sum_{i=1}^{m} \sum_{j \in E_i} (qw_{ij} - P) x_{ij} \leq qW - mP$
$\Rightarrow \sum_{i=1}^{m} \sum_{j \in E_i} qw_{ij} x_{ij} - mP \leq qW - mP$ because exactly m variables x_{ij} are set to 1

$\Rightarrow \sum_{i=1}^{m} \sum_{j \in E_i} w_{ij} x_{ij} \leq W$.
The second MDCP equation can be transformed as follows:
$\sum_{i=1}^{m} \sum_{j \in E_i} r_{ij} x_{ij} \leq \sum_{i=1}^{m} \sum_{j \in E_i} s_{ij} x_{ij}$,
$\Rightarrow 0 \leq \sum_{i=1}^{m} \sum_{j \in E_i} (mp_{ij} - P) x_{ij}$ by simplifying r_{ij} from both sides,
$\Rightarrow 0 \leq \sum_{i=1}^{m} \sum_{j \in E_i} mp_{ij} x_{ij} - mP$ because exactly m variables x_{ij} are set to 1,
$\Rightarrow P \leq \sum_{i=1}^{m} \sum_{j \in E_i} p_{ij} x_{ij}$. □

Proposition 2 states that $pcost$ allows us to exactly identify soft neighborhood substitutable values when two conditions are verified: $\gamma(x)$ is separable and $pcost(x, a \rightarrow b) = (\beta, \alpha)$ with $\alpha < k$. In the following proposition, by construction, we deal with instances such that $\gamma(x)$ is separable. However, nothing is imposed on α, with the consequence that some SNS-eliminable values cannot be detected in polynomial time.

Proposition 4. *Deciding if a value is soft neighborhood substitutable for another on a WCN $(\mathscr{X}, \mathscr{W}, k)$ where $k \neq \infty$ is coNP-hard.*

Proof. Any MDCP instance can be reduced to the problem of deciding whether a value (x, a) is **not** soft substitutable for a value (x, b) on a WCN P. From the MDCP instance, we build the WCN P as follows. $vars(P)$ contains a variable x such that $dom(x) = \{a, b\}$ and a variable y_i for each set E_i ; the domain of y_i contains the objects $o_{i1}, o_{i2}, \ldots$ of E_i. $cons(P)$ contains exactly m binary soft constraints w_{xy_i}: we have $w_{xy_i}(\{(x, a), (y_i, o_{ij})\}) = s_{ij}$ and $w_{xy_i}(\{(x, b), (y_i, o_{ij})\}) = r_{ij}$. k is set to C. Determining if (x, a) is not soft substitutable for (x, b) on P is equivalent to finding an instantiation I of $vars(\Gamma(x))$ such that $cost_{\Gamma(x)}(I_{x=a}) > cost_{\Gamma(x)}(I_{x=b})$ which is equivalent to $\sum_{c_S \in \Gamma(x)} c_s(I_{x=b}) < k \wedge \sum_{c_S \in \Gamma(x)} c_s(I_{x=a}) > \sum_{c_S \in \Gamma(x)} c_s(I_{x=b})$. The first (resp. second) condition encodes the first (resp. second) inequality of the MDCP. Since variables x_{ij} of the MDCP instance correspond to the assignment of variables y_i in the WCSP ($x_{ij} = 1 \Leftrightarrow y_i = o_{ij}$), the third equation of the MDCP instance is directly encoded in the WCSP instance. □

5 Relationships with Soft Arc Consistency

After introducing soft neighborhood substitutability closure, this section presents some results relating soft neighborhood substitutability to various forms of soft arc consistency.

Definition 5. *The soft neighborhood substitutability closure (or SNS-closure) of a WCN P, denoted by $SNS(P)$, is any WCN obtained after iteratively removing SNS-eliminable values until convergence.*

Since this operation is not confluent, $SNS(P)$ is not unique. When we use the $pcost$ approach to identify SNS-eliminable values, we note $PSNS(P)$.

Proposition 5. *Let P be an EDAC-consistent WCN. $SNS(P)$ is not necessarily EDAC-consistent.*

Proof. Consider the WCN P depicted in Figure 2(a). Note that P is EDAC-consistent w.r.t. the order $w > x > z > y$, and that (w, a) and (z, a) are respectively soft neighborhood substitutable for (w, b) and (z, b), since $pcost(w, a \rightarrow b) = (0, 0)$ and $pcost(z, a \rightarrow b) = (0, 0)$. There exists a unique SNS-closure of P, $P' = SNS(P)$, which is depicted in Figure 2(b). Clearly, P' is not EDAC-consistent since (x, b) and (y, b) have no support on c_{xy}. □

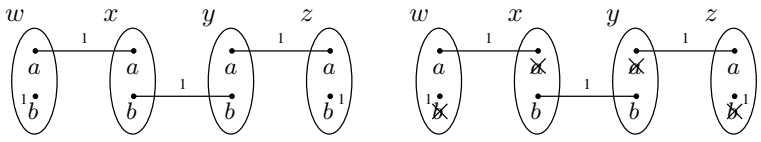

Fig. 2. EDAC versus SNS

Lemma 1. *Let P be a VAC-consistent WCN, $x \in S$ and $\{a, b\} \subseteq dom(x)$ such that (x, b) is AC-consistent in $Bool(P)$. If $pcost(x : a \rightarrow b) \geq 0$ on P then (x, a) is neighborhood substitutable (in the CSP sense [10]) for (x, b) on $Bool(P)$.*

Proof. We suppose that P is VAC-consistent and (x, b) is AC-consistent in $Bool(P)$. Because (x, b) is AC-consistent in $Bool(P)$, we know that for every constraint $c_S \in \Gamma(x)$, there exists an instantiation I of S such that $I[x] = b$ and $c_S(I) = 0$ (by construction of $Bool(P)$). This means that on every such constraint c_S, $pcost(c_S, x : a \rightarrow b) \leq 0$. As by hypothesis $pcost(x : a \rightarrow b) \geq 0$, for every constraint $c_S \in \Gamma(x)$, we have necessarily $pcost(c_S, x : a \rightarrow b) = 0$. We can deduce that for every constraint $c_S \in \Gamma(x)$, for every instantiation I of S such that $I[x] = b$ and $c_S(I) = 0$, the instantiation $I' = I_{x=a}$ is such that $c_S(I') = 0$. Finally, we can conclude that (x, a) is neighborhood substitutable for (x, b) on $Bool(P)$. □

Proposition 6. *Let P be a VAC-consistent WCN, $x \in vars(P)$ and $\{a, b\} \subseteq dom(x)$. If $pcost(x : a \rightarrow b) \geq 0$ on P then $P \setminus \{(x, b)\}$ is VAC-consistent.*

Proof. On the one hand, suppose that (x, b) is AC-consistent in $Bool(P)$. From the previous lemma, we know that (x, a) is neighborhood substitutable for (x, b) on $Bool(P)$, and so we have $AC(Bool(P)) \neq \bot \Leftrightarrow AC(Bool(P \setminus \{(x, b)\})) \neq \bot$. Because P is VAC-consistent, necessarily $P \setminus \{(x, b)\}$ is VAC-consistent. On the other hand, suppose that (x, b) is not AC-consistent in $Bool(P)$. Clearly we have $AC(Bool(P)) = AC(Bool(P \setminus \{(x, b)\}))$, and so $P \setminus \{(x, b)\}$ is VAC-consistent (since P is VAC-consistent). □

Corollary 2. *Let P be a WCN. If P is VAC-consistent then $SNS(P)$ is VAC-consistent.*

The previous corollary also holds for OSAC (because OSAC is stronger than VAC).

6 Algorithms

In this section, we introduce an algorithm to enforce AC*+PSNS (that can be easily adapted to EDAC+PSNS, for example). The main idea is always to start identifying SNS-eliminable values from a WCN that is AC*-consistent. This allows us to reduce the computation effort by using early breaks and residues.

The main procedure is Algorithm 1. As usual, we use a set, denoted by Q, to store the variables whose domain has been recently reduced. Initially, Q is initialized with all variables (line 4). Then, at line 6, a classical AC* algorithm, denoted here by W-AC*, is run (for example, this may be W-AC*2001 [16]), before soliciting a function called PSNS^r. The calls to W-AC* and PSNS^r are interleaved until a fixed point is reached (i.e., $Q = \emptyset$).

Function PSNS^r, Algorithm 2, iterates over all variables in order to collect SNS-eliminable values into a set called Δ. This set is initialized at line 1 and updated at lines 6 and 8. Let us imagine that all SNS-eliminable values (that can be identified by means of pcost) for a variable x have been deleted, and that the domains of all variables in the neighborhood of x remain the same (while possible reductions happen elsewhere). Clearly, there is no need to consider x again for seeking SNS-eliminable values. This is the point of line 3. Here, timestamps are used. By introducing a global counter $time$ and by associating a time-stamp $stamp[x]$ with every variable x as well as a time-stamp $substamp$ with function PSNS^r, it is possible to determine which variables should be considered. The value of $stamp[x]$ indicates at which moment a value was most recently removed from $dom(x)$, while the value of $substamp$ indicates at which moment PSNS^r was most recently called. Variables $time$, $stamp[x]$ for each variable x and $substamp$ are initialized at lines 1 to 3 of Algorithm 1. The value of $time$ is incremented whenever a variable is added to Q (line 13 of Algorithm 2 and this *must* also be performed *inside* W-AC*) and whenever PSNS^r is called (line 9). All SNS-eliminable values collected in Δ are removed while updating Q for the next call to W-AC* (lines 10 to 12).

Algorithm 1: AC*-PSNS(P : WCN)

Output: P, made AC*-consistent and PSNS-closed

1 $time \leftarrow 0$;
2 $substamp \leftarrow -1$;
3 $stamp[x] \leftarrow 0, \forall x \in vars(P)$;
4 $Q \leftarrow vars(P)$;
5 **repeat**
6 | $PSNS^r(W\text{-}AC^*(P, Q))$;
7 **until** $Q \neq \emptyset$;

Algorithm 3 allows us to compute the pcost of (x, b) with respect to (x, a). Because we know that the WCN is currently AC*-consistent, we have the guarantee that the pcost of (x, b) with respect to (x, a) on any non-unary constraint c_S where $x \in S$ is less than or equal to 0. This means that we can never compensate a pair (β, α) s.t.

Algorithm 2: PSNS^r(P : WCN AC*-consistent)

```
Δ ← ∅ ;
foreach x ∈ vars(P) do
    if ∃y ∈ Γ(x) | stamp[y] > substamp then
        foreach (a, b) ∈ dom(x)^2 | b > a do
            if pcost(x, a → b) ≥ 0 then
                Δ ← Δ ∪ {(x, b)} ;
            else if pcost(x, b → a) ≥ 0 then
                Δ ← Δ ∪ {(x, a)} ;
substamp ← time++ ;
foreach (x, a) ∈ Δ do
    remove (x, a) from dom(x) ;
    Q ← Q ∪ {x} ;
    stamp[x] ← time++ ;
```

Algorithm 3: pcost($x, a \to b$): cost pair

```
pcst ← (c_x(b), c_x(a)) ;
if pcst < 0 then
    return pcst
pcst ← pcst + pcost(residues[x, a, b], x, a → b) ;
if pcst < 0 then
    return pcst
foreach c_S ∈ Γ(x) | c_S ≠ residues[x, a, b] do
    d ← pcost(c_S, x, a → b) ;
    if d < (c_x(a), c_x(b)) then
        residues[x, a, b] ← c_S ;
    pcst ← pcst + d ;
    if pcst < 0 then
        return pcst
return pcst
```

Algorithm 4: pcost($c_S, x : a \to b$): cost pair

```
pcst ← (1, 0) ;
foreach I ∈ l(S \ {x}) do
    if (c_S(I_{x=b}), c_S(I_{x=a})) < pcst then
        pcst ← (c_S(I_{x=b}), c_S(I_{x=a})) ;
return pcst
```

$\alpha > \beta$ with a pair (β', α') s.t. $\alpha' < \beta$ (once the pair of unary costs has been taken into account). A strong benefit of that observation is the possibility of using early breaks

during such computations. This is performed at lines 3, 6 and 13. Residues are another mechanism used to increase the performance of the algorithm. For every variable x, and every pair (a, b) of values in $dom(x)$, we store in $residues[x, a, b]$ the constraint c_S which guarantees that (x, b) is not SNS-eliminable by (x, a), if it exists. The residual constraint takes priority (lines 4 to 6); this way, if it compensates the initial unary cost, we avoid unnecessary work. It is updated at lines 9-10. Note that we can initialize the array $residues$ with any arbitrary constraints (not shown in the algorithm), and that Algorithm 4 necessarily returns a value less than or equal to 0 (this explains the initialization of $pcst$ to $(1, 0)$ at line 1).

We now discuss the complexity of PSNS^r while assuming (for simplicity) that the WCN is binary. The space complexity is $O(nd^2)$ due to the use of the structure $residues$. The time complexity of Algorithm 4 is $O(d)$, and the time complexity of Algorithm 3 is $O(1)$ in the best case (if it is stopped at line 3) and $O(qd)$ in the worst case, where $q = |\Gamma(x)|$. Discarding line 3, the time complexity of Algorithm 2 is $O(nd^2)$ in the best case and $O(nd^3e)$ in the worst-case. Of course, Algorithm 2 can be called several times at line 6 of Algorithm 1, so we obtain a worst-case time complexity in $O(n^2d^4e)$. However, we have observed in our experiments that the number of successive calls to PSNS^r is quite limited in practice (as we were predicting). Besides, imagine now that we call AC*-PSNS after the assignment of a value to a variable x (i.e., during search). In the best case (from a complexity point of view), no removal is made by $W\text{-}AC*$, and so, we just consider the neighbors of x, due to line 3 of Algorithm 2, which gives a best-case time complexity in $O(qd^2)$. This last result leans us toward experimenting maintaining AC*-PSNS during search.

7 Experimental Results

To show the practical value of removing SNS-eliminable values, we have conducted an experimentation using the series of WCSP instances available at `http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/BenchmarkS` and a cluster of Xeon 3.0GHz with 1 GB of RAM under Linux. Our goal is to observe the relative efficiency of solving WCSP instances when *maintaining* AC*, AC*+PSNS, FDAC, FDAC+PSNS, EDAC, and EDAC+PSNS. For variable ordering during search, we use the simple static heuristic *max degree* which is independent of the pruning efficiency of the different algorithms, as in [8] where some experiments have been performed with a partial form of SNS enforced during a preprocessing stage.

Table 1 shows the average results obtained on various series. For each series, the number of considered instances (#inst) is given below the name of the series. We have discarded the instances that have not been solved by at least one of the algorithms, within 1,200 seconds. Here, a solved instance means that an optimal solution has been found and proved. In Table 1, the number (#sol) of instances solved within 1,200 seconds is given as well as the average CPU time over these solved instances. The average number (avg-sub) of SNS-eliminable values deleted during search (at each step) is also given (rounded to the nearest integer). On RLFAP instances (celar, scens, graphs), the benefit of using PSNS is rather erratic, but on planning (driver, mprime), coloring (myciel, geom), spot and warehouse instances, we can see a clear advantage of embedding

Series		AC*	AC* +PSNS	FDAC	FDAC +PSNS	EDAC	EDAC +PSNS
celar	#sol (cpu)	5 (337)	4 (231)	6 (316)	6 (341)	6 (344)	**7** (461)
#inst=7	avg-sub	0	3	0	6	0	4
driver	#sol (cpu)	18 (103)	18 (52)	19 (39.3)	19 (19.7)	19 (68.5)	**19 (56.8)**
#inst=19	avg-sub	0	1	0	1	0	1
geom	#sol (cpu)	5 (37.4)	5 (12.4)	5 (18.8)	**5 (11.0)**	5 (21.0)	5 (12.0)
#inst=5	avg-sub	0	0	0	1	0	0
mprime	#solv (cpu)	4 (9.82)	8 (17.4)	4 (11.6)	**8 (12.0)**	5 (206)	8 (21.0)
#inst=8	avg-sub	0	1	0	1	0	1
myciel	#sol (cpu)	3 (122)	3 (77.0)	3 (72.9)	**3 (34.3)**	3 (80.4)	3 (37.8)
#inst=3	avg-sub	0	2	0	3	0	3
scens+graphs	#sol (cpu)	1 (20.4)	3 (113)	**8** (242)	7 (186)	6 (266)	5 (19.8)
#inst=9	avg-sub	0	8	0	8	0	8
spot5	#sol (cpu)	0	0	3 (20.5)	3 (12.6)	3 (20.1)	**3 (11.4)**
#inst=3	avg-sub	0	0	0	0	0	0
warehouse	#sol (cpu)	12 (286)	12 (46.2)	18 (166)	24 (139)	28 (53.2)	**29** (79.8)
#inst=34	avg-sub	0	4	0	7	0	27
	#solved	48	53	66	75	75	**79**

Table 1. Mean results obtained on various series (a time-out of 1,200 seconds was set per instance).

PSNS. Overall, maintaining PSNS is cost-effective as it usually offers a benefit both in terms of solved instances (see last line of the table) and CPU time.

Table 2 presents the results obtained on some representative instances. It is interesting to note that on the warehouse instances (here, *cap101*, *cap111* and *capmo1*), enforcing PSNS does not entail a reduction of the size of the search tree (see the values of #nodes). However, PSNS permits to significantly reduce the size of the domains, making propagation of soft constraints quicker. On *celar7-sub1*, note that (maintaining) EDAC+PSNS is only about 50% faster than EDAC while the number of nodes has been divided by 10. This means that on such instances, PSNS is rather expensive, which maybe lets room for further improvements.

8 Conclusion

In this paper, we have investigated the property of soft neighborhood substitutability for weighted constraint networks, and have found a sufficient condition on substitutions that can be identified by an algorithm of reasonable complexity (restricted to the neighborhood and polynomial). We have proved that, even in simple cases, when $k \neq \infty$, the problem of deciding whether a value is soft neighborhood substitutable for another is coNP-hard. We have also given some properties linking substitutability and soft arc consistencies. Finally, we have proposed an algorithm that exploits early breaks and residues and shown experimentally that it can be helpful during search.

Instances		AC*	AC* +PSNS	FDAC	FDAC +PSNS	EDAC	EDAC +PSNS
cap101	cpu	232	42.1	1.6	1.62	1.48	**1.12**
	#nodes	242K	242K	835	835	75	75
	avg-sub	0	1	0	2	0	15
cap111	cpu	>1,200	>1,200	633	162	3.03	**2.74**
	#nodes	–	–	72,924	72,924	439	199
	avg-sub	0	2	0	3	0	12
capmo1	cpu	>1,200	>1,200	>1,200	>1,200	>1,200	**984**
	#nodes	–	–	–	–	–	–
	avg-sub	0	15	0	23	0	64
driverlog02ac	cpu	253	49.5	10.6	**7.99**	19.3	13.5
	#nodes	4,729K	701K	19,454	8,412	19,444	8,402
	avg-sub	0	0	0	0	0	0
driverlogs06	cpu	>1,200	>1,200	187	**61.0**	218	80.1
	#nodes	–	–	2,049K	609K	2,049K	609K
	avg-sub	0	0	0	0	0	0
mprime04ac	cpu	>1,200	30.9	>1,200	**15.7**	>1,200	43.7
	#nodes	–	189K	–	22,373	–	20,381
	avg-sub	0	0	0	1	0	1
myciel5g-3	cpu	38.1	19.6	**3.87**	4.18	4.36	4.57
	#nodes	518K	168K	10,046	9,922	6,159	6,128
	avg-sub	0	2	0	3	0	2
celar7-sub1	cpu	925	820	147	145	135	**86.4**
	#nodes	9,078K	1,443K	732K	91,552	796K	70,896
	avg-sub	0	6	0	9	0	6
graph07	cpu	>1,200	261	**3.11**	3.58	3.86	4.3
	#nodes	6,156K	145K	1,112	647	1,796	1,514
	avg-sub	0	23	0	9	0	4
scen06-24	cpu	>1,200	>1,200	**1,061**	>1,200	>1,200	>1,200
	#nodes	–	–	–	375K	–	–
	avg-sub	0	2	0	6	0	7
spot5-29	cpu	>1,200	>1,200	30.1	**18.0**	47.2	22.5
	#nodes	–	–	343K	174K	352K	185K
	avg-sub	0	0	0	0	0	0

Table 2. Illustrative results obtained on some problem instances.

Acknowledgments

This work has been supported by both CNRS and OSEO within the ISI project 'Pajero'.

References

1. S. Bistarelli, B. Faltings, and N. Neagu. Interchangeability in soft CSPs. In *Proceedings of CSCLP'02*, pages 31–46, 2002.
2. S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4(3):199–240, 1999.
3. B.Y. Choueiry, B. Faltings, and R. Weigel. Abstraction by interchangeability in resource allocation. In *Proceedings of IJCAI'95*, pages 1694–1710, 1995.
4. M. Cooper. Reduction operations in fuzzy or valued constraint satisfaction. *Fuzzy Sets and Systems*, 134(3):311–342, 2003.
5. M. Cooper, S. de Givry, M. Sanchez, T. Schiex, and M. Zytnicki. Virtual arc consistency for weighted CSP. In *Proceedings of AAAI'08*, pages 253–258, 2008.
6. M. Cooper, S. de Givry, M. Sanchez, T. Schiex, M. Zytnicki, and T. Werner. Soft arc consistency revisited. *Artificial Intelligence*, 174(7-8):449–478, 2010.
7. M.C. Cooper. Fundamental properties of neighbourhood substitution in constraint satisfaction problems. *Artificial Intelligence*, 90:1–24, 1997.
8. S. de Givry. Singleton consistency and dominance for weighted CSP. In *Proceedings of CP'04 Workshop on Preferences and Soft Constraints*, 2004.
9. S. de Givry, F. Heras, M. Zytnicki, and J. Larrosa. Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In *Proceedings of IJCAI'05*, pages 84–89, 2005.
10. E.C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proceedings of AAAI'91*, pages 227–233, 1991.
11. A. Haselbock. Exploiting interchangeabilities in constraint satisfaction problems. In *Proceedings of IJCAI'93*, pages 282–287, 1993.
12. S. Karakashian, R. Woodward, B. Choueiry, S. Prestwhich, and E. Freuder. A partial taxonomy of substitutability and interchangeability. Technical Report arXiv:1010.4609, CoRR, 2010.
13. A.M. Koster. *Frequency assignment: Models and Algorithms*. PhD thesis, University of Maastricht, The Netherlands, 1999.
14. A. Lal, B.Y. Choueiry, and E.C. Freuder. Neighborhood interchangeability and dynamic bundling for non-binary finite CSPs. In *Proceedings of AAAI'05*, pages 397–404, 2005.
15. J. Larrosa. Node and arc consistency in weighted CSP. In *Proceedings of AAAI'02*, pages 48–53, 2002.
16. J. Larrosa and T. Schiex. Solving weighted CSP by maintaining arc consistency. *Artificial Intelligence*, 159(1-2):1–26, 2004.
17. S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, 1990.
18. T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of IJCAI'95*, pages 631–639, 1995.