# Optimal soft arc consistency

Martin Cooper, Simon de Givry, Thomas Schiex

# Optimal Soft Arc Consistency

**M.C. Cooper**
IRIT, Univ. Toulouse III, France
cooper@irit.fr

**S. de Givry** and **T. Schiex**
INRA, Applied Math. & CS Dept., France
{degivry,tschiex}@toulouse.inra.fr

## Abstract

The Valued (VCSP) framework is a generic optimization framework with a wide range of applications. Soft arc consistency operations transform a VCSP into an equivalent problem by shifting weights between cost functions. The principal aim is to produce a good lower bound on the cost of solutions, an essential ingredient of a branch and bound search.

But soft AC is much more complex than traditional AC: there may be several closures (fixpoints) and finding the closure with a maximum lower bound has been shown to be NP-hard for integer costs [Cooper and Schiex, 2004].

We introduce a relaxed variant of soft arc consistency using rational costs. In this case, an optimal closure can be found in polynomial time. Furthermore, for finite rational costs, the associated lower bound is shown to provide an optimal arc consistent reformulation of the initial problem.

Preliminary experiments on random and structured problems are reported, showing the strength of the lower bound produced.

## 1 Introduction

The Valued Constraint Satisfaction Problem [Schiex *et al.*, 1995] is a generic cost function optimization framework with many applications. A VCSP is defined by a set of variables with finite domains and a set of local cost functions (soft constraints) which associate costs to tuples. The goal is then to find an assignment to all variables with a minimum combined cost. Costs from different constraints are combined with a domain dependent operator $\oplus$. In this paper, we focus on Weighted CSP (WCSP) where costs are natural numbers, combined by addition. This VCSP sub-case has been shown to capture the essential complexity of non-idempotent VCSP in [Cooper, 2005] and has a lot of direct applications in domains such as *resource allocation*, *combinatorial auctions*, *bioinformatics*, *probabilistic reasoning*, *graph theory*...

Following the initial definition of arc consistency for non idempotent operators in [Schiex, 2000], local consistency is now considered as a crucial mechanism for solving WCSPs. Based on iterated integer cost movements between cost functions of different arities which preserve problem equivalence, local consistency offers all the services of local consistency in classical CSP. It is specifically capable of producing strong incrementally maintained lower bounds which are crucial for efficient branch and bound search. It is however plagued by the absence of uniqueness of the fixpoint (so-called arc consistent closure) and by the fact that finding a closure which maximizes the lower bound is an NP-hard problem [Cooper and Schiex, 2004]. This has led to the definition of a large number of variants of arc consistency (directional [Cooper, 2003], existential [Larrosa *et al.*, 2005], cyclic [Cooper, 2004]...), each using different strategies to try to get closer to an optimal closure.

In this paper, we show that by relaxing the condition that costs be integers and by allowing simultaneous cost movements between cost functions, it is possible to define a new class of local consistency such that finding an optimal closure is a polynomial time problem. On a large class of problems, we show that the lower bound produced is always better than the previous integer arc consistency variants and provides an optimal reformulation of the initial problem.

Beyond a better understanding of local consistency in WCSPs, preliminary experiments of Optimal Soft Arc Consistency on random and structured WCSPs show that it may also be used in practice as a preprocessing algorithm.

## 2 Preliminaries

*Valued* CSP extends the CSP framework by associating *costs* to tuples [Schiex *et al.*, 1995]. In general, costs are specified by means of a so-called *valuation structure* defined as a triple $S = (E, \oplus, \succeq)$, where $E$ is the set of costs totally ordered by $\succeq$. The maximum and a minimum costs are noted $\top$ and $\bot$, respectively. $\oplus$ is a commutative, associative and monotonic operation on $E$ used to combine costs. $\bot$ is the identity element and $\top$ is absorbing.

Weighted CSPs (WCSP) form a specific subclass of valued CSP that relies on a specific valuation structure $S(k)$.

**Definition 2.1** *$S(k)$ is a triple $(\{0, 1, \ldots, k\}, \oplus, \geq)$ where,*

- *$k \in \{1, \ldots, \infty\}$.*
- *$\oplus$ is defined as $a \oplus b = \min\{k, a + b\}$*
- *$\geq$ is the standard order among numbers.*

Observe that in $S(k)$, we have $0 = \bot$ and $k = \top$. $k$ may be either finite or infinite.

A WCSP $P$ is defined by $P = (X, D, C, k)$. The valuation structure is $S(k)$. $X$ and $D$ are sets of variables and domains, as in a standard CSP. For a set of variables $S \subset X$, we note $\ell(S)$ the set of tuples over $S$. $C$ is a set of cost functions. Each cost function (or soft constraint) $c_S$ in $C$ is defined on a set of variables $S$ called its scope. A cost function $c_S$ assigns costs to assignments of the variables in $S$ i.e.: $c_S : \ell(S) \rightarrow \{0, \ldots, k\}$. For simplicity, we assume that every constraint has a different scope. For binary and unary constraints, we use simplified notations: a binary soft constraint between variables $i$ and $j$ is denoted $c_{ij}$. A unary constraint on variable $i$ is denoted $c_i$. We assume the existence of a unary constraint $c_i$ for every variable, and a *zero*-arity constraint, noted $c_\varnothing$ (if no such constraint is defined, we can always define *dummy* ones $c_i(a) = 0, \forall a \in D_i$ and $c_\varnothing = 0$).

When a constraint $c_S$ assigns cost $\top = k$ to a tuple $t$, it means that $c_S$ forbids $t$, otherwise $t$ is permitted by $c_S$ with the corresponding cost. The *cost* of a complete tuple $t$ in a WCSP $P$, noted $\mathcal{V}_P(t)$, is the sum of all costs:

$$\mathcal{V}_P(t) = \sum_{c_S \in C} c_S(t[S])$$

where $t[S]$ denotes the usual projection of a tuple on the set of variables $S$. A complete assignment is feasible if it has a cost less than $k$ and optimal if there are no complete assignment with a lower cost. The problem is to find a complete optimal assignment (NP-hard).

We say that two WCSPs $P$ and $P'$ defined over the same variables are equivalent if they define the same global cost function, i.e. $\mathcal{V}_P$ and $\mathcal{V}_{P'}$ are identical.

## 3 Node and arc consistencies

Local consistency algorithms are characterized by the fact that they reason at a subproblem level and that they preserve equivalence. This is captured by the following notion:

**Definition 3.1** *For a valued CSP with a set of variables $X$, an* equivalence preserving transformation *(EPT) on $W \subseteq X$ is an operation which transforms cost functions whose scope is included in $W$ to produce an equivalent valued CSP.*

In WCSP, basic EPT move costs between cost functions in order to preserve the equivalence of the problem. To move costs, it is necessary to be able to subtract costs from its origin. This is done using the $\ominus$ operation defined as:

$$a \ominus b = \begin{cases} a - b & : \quad a \neq k \\ k & : \quad a = k \end{cases}$$

Although we restrict our presentation to WCSPs for the sake of simplicity, we remind the reader that such a pseudo-difference operator exists in a large class of VCSPs (fair VCSP [Cooper and Schiex, 2004]).

Algorithm 1 gives two elementary EPT. Project works in the scope of a single constraint $c_S$. It moves an amount of cost $\alpha$ from $c_S$ to a unary cost function $c_i, i \in S$, for a value $a \in d_i$. If the cost $\alpha$ is negative, cost moves from the unary cost function $c_i$ to the cost function $c_S$. To guarantee that the problem obtained after each application is valid, one must guarantee that its cost functions remain in the valuation structure. To avoid negative costs, one must have $-c_i(a) \leq \alpha \leq \min_{t \in \ell(S), t[\{i\}]=a}\{c_S(t)\}$. Similarly, ProjectUnary works on a subproblem defined just by one variable $i \in X$. It moves a cost $\alpha$ from the unary cost function $c_i$ to the nullary cost function $c_\emptyset$ (with $-c_\varnothing \leq \alpha \leq \min_{a \in d_i}\{c_i(a)\}$ in order to get a valid WCSP).

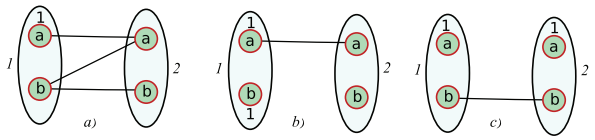



Figure 1: A simple WCSP and two arc consistent closures.

**Algorithm 1**: Project and UnaryProject for soft arc and node consistency enforcing

**Procedure** Project($c_S, i, a, \alpha$)
  $c_i(a) \leftarrow c_i(a) \oplus \alpha$;
  **foreach** ($t \in \ell(S)$ such that $t[\{i\}] = a$) **do**
    $c_S(t) \leftarrow c_S(t) \ominus \alpha$;

**Procedure** UnaryProject($i, \alpha$)
  $c_\varnothing \leftarrow c_\varnothing \oplus \alpha$;
  **foreach** ($a \in d_i$) **do**
    $c_i(a) \leftarrow c_i(a) \ominus \alpha$;

A variable $i$ is said to be node consistent [Larrosa, 2002] if 1) for all values $a \in d_i$, $c_i(a) \oplus c_\varnothing \neq k$, 2) there exists a value $b \in d_i$ such that $c_i(b) = 0$. A WCSP is node consistent (NC) if every variable is node consistent.

Stronger arc level consistencies rely on the notion of *support*. For simplicity, we introduce these notions for binary WCSP but most have been generalized to arbitrary arities (see [Cooper and Schiex, 2004]). A value $b \in d_j$ is a support for a value $a \in d_i$ along $c_{ij}$ if $c_{ij}(a, b) = 0$.

Variable $i$ is arc consistent if every value $a \in d_i$ has a support in every constraint $c_{ij} \in C$. A WCSP is arc consistent (AC) if every variable is arc and node consistent.

When a value $(i, a)$ has no support on a constraint, one can create one using the Project operation. This is exemplified in Figure 1a. This WCSP has two variables with 2 values $a$ and $b$. Vertices, representing values, can be weighted by unary costs. An edge connecting two values represents a cost of 1. If two values are not connected, it means that the corresponding cost is 0. Notice that value $(1, b)$ has no support on variable 2. We can apply Project($c_{12}, 1, b, 1$). This creates a unary cost of 1 for $(1, b)$ which is now AC. Applying UnaryProject$(1, 1)$ on the resulting WCSP (Figure 1b) makes the problem AC and increases $c_\varnothing$ by 1.

But a different sequencing may lead to a different result. If we first note that value $(2, a)$ has no support on $c_{12}$ and apply Project($c_{12}, 2, a, 1$), we get the problem in Figure 1c. It is now AC, but the lower bound $c_\varnothing$ is 0.

Given an initial WCSP $P$, any WCSP obtained from $P$ by iterated applications of Project and UnaryProject with a positive integer $\alpha$ until a fixpoint is reached is called an arc consistent closure of $P$. An AC closure of a WCSP $P$ is optimal if it has the maximum lower bound $c_\varnothing$ among all AC closures of $P$. [Cooper and Schiex, 2004] showed that finding an optimal AC closure for a WCSP is an NP-hard problem.

Alternative definitions of AC have therefore been proposed. They exploit the notion of *full support*. $b$ is a full support for $a$ on $c_{ij}$ if $c_{ij}(a,b) \oplus c_j(b) = 0$. Replacing the notion of support by the stronger notion of full supports in AC is attractive but the corresponding property cannot be enforced [Schiex, 2000]. Restricted versions must be used.

Given a variable ordering $<$, variable $i$ is directional arc consistent (DAC) if every value $a \in d_i$ has a full support in every constraint $c_{ij} \in C$ such that $j > i$. A WCSP is DAC if every variable is DAC and node consistent. It is full directional arc consistent (FDAC) if it is AC and DAC. Several heuristics for ordering variables in FDAC have been tried [Heras and Larrosa, 2006a].

Variable $i$ is existential arc consistent [Larrosa *et al.*, 2005] if there exists $a \in d_i$ such that $c_i(a) = 0$ and $a$ has a full support on every constraint $c_{ij}$. A WCSP is existential arc consistent (EAC) if every variable is EAC and node consistent. It is existential directional arc consistent (EDAC) if it is EAC, and FDAC. All these definitions can be seen as well defined polynomial time heuristics trying to get closer to the optimal closure, but without any guarantee.

## 4 Optimal arc consistency

From now on, we relax the definition of WCSP by allowing rational costs. This mean we use the new valuation structure $S_{\mathbb{Q}}(k) = ([0,k], \oplus, \geq)$ where $[0,k]$ denotes the set of rational numbers (elements of $\mathbb{Q}$) between (and including) 0 and $k$.

**Definition 4.1** *Given a WCSP $P$, a Soft Arc Consistent (SAC) Transformation is a set of soft arc consistency operations* Project *and* UnaryProject *which, when applied simultaneously, transform $P$ into a valid WCSP.*

Note that in this definition there is no precondition on the costs moved individually by each EPT application. We just want the final equivalent WCSP to be valid (with positive rational costs): this also guarantees that $c_\varnothing$ is a lower bound of the optimal cost. Previously, [Affane and Bennaceur, 1998] proposed to split integer costs by propagating a fraction $w_{ij}$ of the binary constraint $c_{ij}$ towards variable $i$ and the remaining fraction $1 - w_{ij}$ towards $j$ (where $0 \leq w_{ij} \leq 1$), suggested determining optimal $w_{ij}$ but just used $w_{ij} = \frac{1}{2}$. [Bennaceur and Osamni, 2003] further suggested to use different weights $w_{iajb}$ for every pair of values $(a,b) \in d_i \times d_j$. It turns out that using one weight for each triple $(i,j,a)$ where $a \in d_i$, allows us to find optimal weights in polynomial time.

**Theorem 4.2** *If the valuation structure $S_{\mathbb{Q}}(\infty)$ is used, then it is possible to find in polynomial time a SAC transformation of $P$ which maximizes the lower bound $c_\varnothing$ provided the arity of the constraints in $P$ is bounded.*

**Proof:** Assume first as in [Cooper, 2003] that all infinite costs have been propagated using standard generalized AC [Mohr and Masini, 1988]. Then only finite costs can be further propagated by Project and UnaryProject. We then want to determine a set of SAC operations which, when applied simultaneously, maximize the increase in $c_\varnothing$. For each $c_S \in C$ with $|S| > 1$, and for every variable $i \in S$, let $p^S_{i,a}$ be the amount of cost projected from $c_S$ to $c_i(a)$ (remember that costs moved from $c_i(a)$ to $c_S$ are counted as negative in Project). Let $u_i \geq 0$ be the amount of cost projected by UnaryProject from $c_i$ to $c_\varnothing$. Thus the problem is to maximize $\sum_i u_i$ while keeping the WCSP valid (no negative cost appears) i.e.:

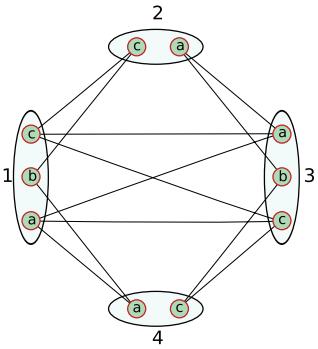
$$\forall i \in X, \forall a \in d_i, \quad c_i(a) - u_i + \sum_{(c_S \in C),(i \in S)} p^S_{i,a} \geq 0$$

$$\forall c_S \in C, |S| > 1, \forall t \in \ell(S) \quad c_S(t) - \sum_{i \in S} p^S_{i,t[\{i\}]} \geq 0$$

All inequalities for which $c_i(a) = \top = \infty$ or $c_S(t) = \top$ can be ignored since they are necessarily satisfied. The remaining inequalities define a linear programming problem over $\mathbb{Q}$ with $O(ed + n)$ variables which can be solved in polynomial time [Karmarkar, 1984]. ∎

A local version of the above theorem, limited to subproblems of 3 variables, is actually the basis of the algorithm enforcing 3-cyclic consistency [Cooper, 2004]. In our case, a problem will be called Optimal Soft Arc Consistent (OSAC) when $c_\varnothing$ cannot be improved as above. Obviously, in $S_{\mathbb{Q}}(\infty)$, OSAC is stronger than AC, DAC, FDAC or EDAC.

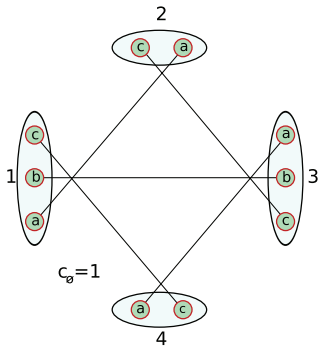
Note that if $k < \infty$, things get more involved since an increase in $c_\varnothing$ can, through node consistency, lead to the deletion of values, thus requiring a new optimization. Because value deletion cannot occur more than $nd$ times, this is still polynomial time but the proof of optimality of the final closure is an open question.




To better understand how OSAC works, consider the 4-variable WCSP on the left. All unary costs are equal to 0. All edges represent a unit cost, omitted for clarity. $c_\varnothing$ is assumed to be 0. None of the basic EPTs can transform the problem into a valid one. However, we may simultaneously perform the following operations:

1. Project($c_{23}, 2, c, -1$): we move (a virtual) cost of 1 from value $(2,c)$ to 3 pairs inside $c_{23}$.
2. Project($c_{23}, 3, a, 1$), Project($c_{23}, 3, b, 1$): this moves two unary costs of 1 to $(3,a)$ and $(3,b)$.
3. Project($c_{34}, 3, a, -1$), Project($c_{31}, 3, b, -1$): these two unary costs are moved inside $c_{34}$ and $c_{31}$ respectively.
4. Project($c_{34}, 4, c, 1$): this moves a unary cost of 1 to $(4,c)$.
5. Project($c_{31}, 1, a, 1$), Project($c_{31}, 1, c, 1$): this moves two unary costs of 1 to $(1,c)$ and $(1,a)$.

6. Project$(c_{12}, 1, a, -1)$, Project$(c_{12}, 2, c, 1)$: we reimburse our initial loan on value $(c, 2)$.
7. Project$(c_{14}, 1, c, -1)$, Project$(c_{14}, 4, a, 1)$: we send a unary cost to value $(4, a)$.




Finally, the application of UnaryProject$(4, 1)$ yields the problem above with a lower bound $c_\emptyset = 1$. This set of EPT corresponds to a solution of the previous linear programming problem where $p^{23}_{2c} = p^{34}_{3a} = p^{31}_{3b} = p^{12}_{1a} = p^{14}_{1c} = -1$ and $p^{23}_{3a} = p^{23}_{3b} = p^{34}_{4c} = p^{31}_{1a} = p^{31}_{1c} = p^{12}_{2c} = p^{14}_{4a} = u_4 = 1$ (all other variables being equal to 0).

## 5 Properties

Theorem 4.2 shows that on $S_{\mathbb{Q}}(\infty)$, OSAC always provides an optimal lower bound using only a set of arc level preserving operations. Actually, one can prove that for a large class of WCSP the final problem obtained is an optimal reformulation of the problem using the initial set of scopes.

**Definition 5.1** *A WCSP $P$ is* in-scope $c_\varnothing$-irreducible *if there is no equivalent WCSP $Q$ with the same set of constraint scopes as $P$ and such that $c^Q_\varnothing > c^P_\varnothing$ (where $c^P_\varnothing$, $c^Q_\varnothing$ are the nullary constraints in $P$, $Q$).*

**Theorem 5.2** *Let $P$ be a WCSP over $S_{\mathbb{Q}}(\infty)$ using only finite costs. If no SAC transformation applied to $P$ produces a WCSP $Q$ with $c^Q_\varnothing > c^P_\varnothing$, then $P$ is in-scope $c_\varnothing$-irreducible.*

**Proof:** This is a direct consequence of Lemma 5.2 in [Cooper, 2004]. ∎

Thus, for finite rational costs, OSAC can be used to establish in-scope $c_\varnothing$-irreducibility. This naturally does not work when infinite costs (hard constraints) exist. The 3-clique 2-coloring problem is inconsistent and is therefore equivalent to the same problem with just $c_\varnothing$ set to $\top$ but no SAC transformation can be applied to this WCSP.

Naturally, higher-order consistencies which may change scopes could provide possibly stronger lower bounds: soft 3-consistency [Cooper, 2005] or soft path inverse consistency [Heras and Larrosa, 2006b] applied to the previous 2-coloring problem would detect infeasibility.

## 6 Experiments

We applied OSAC during preprocessing. The linear programming problem defined by OSAC was solved using ILOG CPLEX version 9.1.3 (using the barrier algorithm). The lower bound produced is then compared to the lower bound produced by EDAC.

The first set of instances processed are random Max-CSP created by the *random_vcsp* generator[1] using the usual four parameters model. The aim here is to find an assignment that minimizes the number of violated constraints. Four different categories of problems with domain size 10 were generated following the same protocol as in [Larrosa *et al.*, 2005]: sparse loose (SL, 40 var.), sparse tight (ST, 25 var.), dense loose (DL, 30 var.) and dense tight (DT, 25 var.) (see the Benchmarks section of [de Givry *et al.*, 2006b]).

| | SL | ST | DL | DT |
|---|---|---|---|---|
| Optimum | 2.84 | 19.68 | 2.22 | 29.62 |
| EDAC lb. | 0 | 4.26 | 0 | 9.96 |
| OSAC lb. | 0 | 12.30 | 0 | 19.80 |

Samples have 50 instances. The table above shows respectively the average optimum value, the average values of the EDAC lower bound and the average value of the OSAC lower bound. On loose problems, OSAC and EDAC leave the lower bound unchanged. This shows that higher level local consistencies are required here. However for tight problems, OSAC is extremely powerful, providing lower bounds which are sometime three times better than EDAC.

The second set of benchmarks is defined by open instances of the Radio Link Frequency Assignment Problem of the CELAR [Cabon *et al.*, 1999].[2] These problems have been extensively studied (see http://www.zib.de/fap/problems/CALMA) but the gap between the best upper bound (computed by local search methods) and the best lower bound (computed by exponential time algorithms) is not closed. The problem considered are the `scen0{7,8}reduc.wcsp` and `graph1{1,3}reducmore.wcsp` which have already been through different strong preprocessing (see the Benchmarks section in [de Givry *et al.*, 2006b]).

| | scen07 | scen08 | graph11 | graph13 |
|---|---|---|---|---|
| Total # of values | 4824 | 14194 | 5747 | 13153 |
| Best known ub | 343592 | 262 | 3080 | 10110 |
| Best known lb | 300000 | 216 | 3016 | 9925 |
| EDAC lb | 10000 | 6 | 2710 | 8722 |
| OSAC lb | 31453.1 | 48 | 2957 | 9797.5 |
| Cpu-time | 3530” | 6718” | 492” | 6254” |

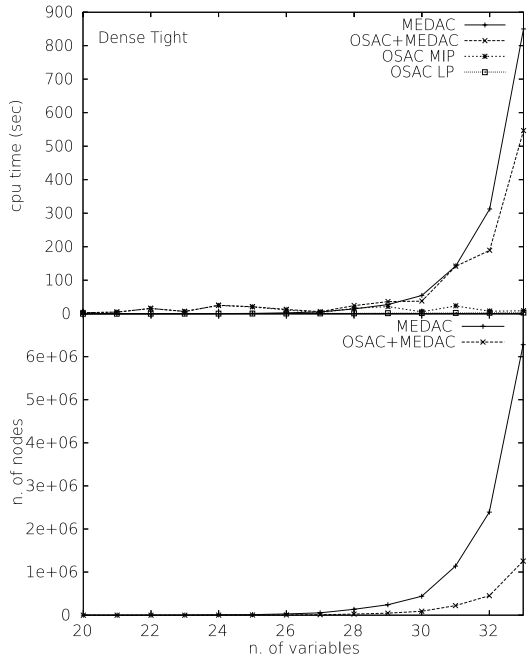
As the table above shows, OSAC offers substantial improvements over EDAC, especially on the graph11 and graph13 instances. For these instances, the optimality gap $\frac{UB-OSAC}{UB}$ is reduced to 4% and 3% respectively. The polynomial time lower bounds obtained by OSAC are actually close to the best known (exponential time) lower bounds. The cpu-time show the cpu-time for computing the OSAC lower bound.

To actually assess the practical interest of OSAC for preprocessing, we tried to solve problems where OSAC was effective: tight problems. The difficulty here lies in the fact

[1] www.inra.fr/mia/T/VCSP

[2] We would like to thank the french Centre Electronique de l'Armement for making these instances available.

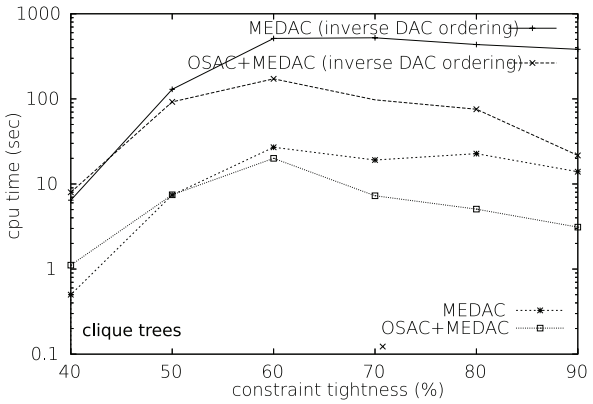
that CPLEX is a floating point solver while the open source WCSP solver used (toolbar, section Algorithms in [de Givry *et al.*, 2006b]) deals with integer costs. To address this issue, we use "fixed point" costs: for all WCSP considered, we first multiply all costs by a large integer constant $\lambda = 1,000$, and then solve the linear programming problem defined by OSAC using integer variables (instead of floating point). The first integer solution found is used. The resulting problem has integer costs and can be tackled by toolbar[3]. This means that we shift from a polynomial problem to an NP-hard one. In practice, we found that the problems obtained have a very good linear continuous relaxation and are not too expensive to solve as integer problems. Using a polytime rational LP solver would allow time complexity to remain polynomial.




The Figure above reports cpu-time (up) and size of the tree search (down) for dense tight problems of increasing size. The time limit was set to 1800". Four cpu-times are reported: (1) OSAC LP: time taken by CPLEX to solve the first linear relaxation (2) OSAC MIP: time taken to get the first integer solution, (3) MEDAC: time taken to solve the original problem by maintaining EDAC [Larrosa *et al.*, 2005] in toolbar with default parameters and a good initial upper bound, (4) OSAC+MEDAC is the sum of OSAC MIP with the time needed by toolbar to solve the OSAC problem (with the same default parameters and upper bound).

Clearly, for small problems (with less than 29 variables), OSAC is more expensive than the resolution itself. But as the problem size increases, OSAC becomes effective and for 33 variables, it divides the overall cpu-time by roughly 2. The number of nodes explored by toolbar in both cases shows the strength of OSAC used as a preprocessing technique (remember that EDAC is maintained during search).

The strength of OSAC compared to local consistencies using full supports such as DAC is that is does not require an initial variable ordering. Indeed, DAC directly solves tree-structured problems but only if the variable ordering used for DAC enforcing is a topological ordering of the tree. To evaluate to what extent OSAC can overcome these limitations, we used random problems structured as binary clique trees as in [de Givry *et al.*, 2006a]. Each clique contains 6 variables with domain size 5, each sharing 2 variables with its parent clique. The overall tree height is 4 leading to a total number of 62 variables, with a graph density of 11%.




The figure above uses a logarithmic scale for cpu-time for different constraint tightnesses (below 40%, problems are satisfiable). On these tree-like problems, two DAC ordering were used. One is compatible with a topological ordering of the binary tree (and should give good lower bounds), the inverse order can be considered as pathological. The cpu-times for MEDAC alone (default toolbar parameters and upper bound) and OSAC+MEDAC (as previously) are shown in each case. Clearly, OSAC leads to drastic (up to 20 fold) improvements when a bad DAC ordering is used. Being used just during preprocessing, it does not totally compensate for the bad ordering. But, even when a good DAC ordering is used, OSAC still allows impressive (up to 4 fold) speedups, especially on tight problems.

Finally, we also tried to solve the challenging open CELAR instances after OSAC preprocessing. Despite the strength of OSAC, all problems remained unsolvable. Simultaneously taking into account the strong structure of the problems as in [de Givry *et al.*, 2006a] is an attractive direction here.

## 7 Related work

As a reviewer pointed out, the LP program optimized for OSAC is the dual of the relaxation of the 01-linear formulation proposed in [Koster, 1999]. The LP formulation of OSAC was also found in the image processing community [Schlesinger, 1976; Werner, 2005].

If all constraints are binary over boolean domains, WCSP reduces to weighted MAX-2SAT. A direct encoding of MAX-2SAT to *quadratic* pseudo-boolean function optimization [Boros and Hammer, 2002] exists: using 0 and 1 to represent respectively true and false, $\times$ for disjunction, $(1-x)$ for

[3]The code of toolbar has been modified accordingly: if a solution of cost $2\lambda$ is known for example and if the current lb. is $1.1\lambda$ then backtrack occurs since all global costs in the original problem are integer and the first integer above 1.1 is 2, the upper bound.

negation and $+$ for combination of costs, a set of weighted 2-clauses can be transformed into a quadratic function. For example, the clauses $\{a \vee b, a \vee \bar{b}\}$ with weights 3 and 5 translates to the function $f(a, b) = 3ab + 5a(1 - b)$.

The problem of producing a so-called "equivalent quadratic posiform representation" with a high constant term (the equivalent of $c_\varnothing$) has been shown to reduce to the computation of a maximum flow [Goldberg and Tarjan, 1988] in an appropriately defined network [Boros and Hammer, 2002]. Given the close connection between maximum flow and linear programming, a fine comparison of the lower bound produced by OSAC on MAX-2SAT problems and by the max-flow formulation of [Boros and Hammer, 2002] would be interesting.

## 8 Conclusion

OSAC provides a polynomial time optimal arc consistent closure in a large class of WCSPs. Despite very preliminary testing and comparatively high computational cost to enforce it, OSAC already shows its usefulness as a preprocessing algorithm for sufficiently large and tight problems.

Beyond this practical usefulness, we think that OSAC brings to light the fact that, in order to increase their strength, new soft local consistency algorithms should probably not be restricted to the mechanical application of elementary operations but should instead try to identify worthy set of equivalence preserving operations that should be simultaneously applied. If maintaining OSAC during search is an obvious but challenging next step, the construction of simpler limited versions of OSAC should also be considered.

This approach is radically different from the ongoing trend of using higher level consistencies (such as path [Cooper, 2005] or path inverse consistency [Heras and Larrosa, 2006b]). The extension of OSAC to such higher level consistencies is also an open question.

## References


[Affane and Bennaceur, 1998] M. S. Affane and H. Bennaceur. A weighted arc consistency technique for Max-CSP. In *Proc. of the 13$^{th}$ ECAI*, pages 209–213, Brighton, United Kingdom, 1998.

[Bennaceur and Osamni, 2003] H. Bennaceur and A. Osamni. Computing lower bounds for Max-CSP problems. In *Proc. 16$^{th}$ International IEA/AIE conference*, 2003.

[Boros and Hammer, 2002] E. Boros and P. Hammer. Pseudo-Boolean Optimization. *Discrete Appl. Math.*, 123:155–225, 2002.

[Cabon *et al.*, 1999] B. Cabon, S. de Givry, L. Lobjois, T. Schiex, and J.P. Warners. Radio link frequency assignment. *Constraints*, 4:79–89, 1999.

[Cooper and Schiex, 2004] M. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154(1-2):199–227, 2004.

[Cooper, 2003] Martin C. Cooper. Reduction operations in fuzzy or valued constraint satisfaction. *Fuzzy Sets and Systems*, 134(3), 2003.

[Cooper, 2004] M. Cooper. Cyclic consistency: a local reduction operation for binary valued constraints. *Artificial Intelligence*, 155(1-2):69–92, 2004.

[Cooper, 2005] M. Cooper. High-order consistency in Valued Constraint Satisfaction. *Constraints*, 10:283–305, 2005.

[de Givry *et al.*, 2006b] S. de Givry, F. Heras, J. Larrosa, E. Rollon, and T. Schiex. The SoftCSP and Max-SAT benchmarks and algorithms web site. http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/softcsp.

[de Givry *et al.*, 2006a] S. de Givry, T. Schiex, and G. Verfaillie. Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP. In *Proc. of AAAI*, 2006.

[Goldberg and Tarjan, 1988] A. Goldberg and R.E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988.

[Heras and Larrosa, 2006a] F. Heras and J. Larrosa. Intelligent variable orderings and re-orderings in DAC-based solvers for WCSP. *Journal of Heuristics*, 12(4-5):287 – 306, September 2006.

[Heras and Larrosa, 2006b] F. Heras and J. Larrosa. New Inference Rules for Efficient Max-SAT Solving. In *Proc. of AAAI*, 2006.

[Karmarkar, 1984] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[Koster, 1999] A.M.C.A. Koster. Frequency Assignment—Models and Algorithms. PhD. Thesis. UM (Maastricht, The Netherlands), 1999.

[Larrosa *et al.*, 2005] J. Larrosa, S. de Givry, F. Heras, and M. Zytnicki. Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In *Proc. of the 19$^{th}$ IJCAI*, Edinburgh, Scotland, August 2005.

[Larrosa, 2002] J. Larrosa. On arc and node consistency in weighted CSP. In *Proc. of AAAI*, pages 48–53, Edmonton, (CA), 2002.

[Mohr and Masini, 1988] R. Mohr and G. Masini. Good old discrete relaxation. In *Proc. of the 8$^{th}$ ECAI*, pages 651–656, Munchen FRG, 1988.

[Schiex *et al.*, 1995] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems. In *Proc. of the 14$^{th}$ IJCAI*, pages 631–637, Montréal, Canada, August 1995.

[Schiex, 2000] T. Schiex. Arc consistency for soft constraints. In *Proc. of CP'2000*, volume 1894 of *LNCS*, pages 411–424, Singapore, September 2000.

[Schlesinger, 1976] M.I. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4:113–130, 1976. In Russian.

[Werner, 2005] T. Werner. A Linear Programming Approach to Max-sum Problem: A Review. Technical report CTU-CMP-2005-25. Czech Technical University, 2005. http://cmp.felk.cvut.cz/cmp/software/maxsum/