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 CSP (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 arc consistent 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 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

Valued constraint satisfaction problems [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 of 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 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* [Cabon *et al.*, 1999], *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 WCSP. 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 usual 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 lead 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 get closer to an optimal closure.

In this paper, we show that by relaxing the integrity of cost movements 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 problem, we show that the lower bound produced is always better than the previous integer arc consistencies variants and provides a form of optimal reformulation of the initial problem. Beyond a better understanding of local consistencies in VCSP, these results also bring to light an interesting connection between WCSP (and more specifically weighted MAX-2SAT) and the *quadratic* pseudo-boolean function optimization.

Finally, we report preliminary experiments on the quality and usefulness of Optimal Soft Arc Consistency on random and structured WCSP.

2 Preliminaries

Valued CSP extend 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 CSP (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 set of variables and domains, as in 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 $\top$ 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 $\top$ and optimal is there are no complete assignment with a lesser cost. The problem is to find a complete optimal assignment (NP-hard).

We say that two WCSP 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.

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 two following notions:

Definition 2.2 *The* subproblem *of a WCSP (X, D, C, k) on $W \subseteq X$ is the problem WCSP(W) $= (W, D_W, C_W, k)$, where $D_w = \{d_i : i \in w\}$ and $C_w = \{c_S \in C : S \subseteq W\}$.*

Definition 2.3 *For a VCSP $\langle N, D, C, S\rangle$, an* equivalence preserving transformation *(EPT) on $J \subseteq N$ is an operation which transforms the subproblem VCSP(J) into an equivalent VCSP.*

3 Node and arc consistencies

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

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

Although we restrict our presentation to WCSP for the sake of simplicity, please remember that such a pseudo-difference operator exists in a large class of VCSP (so-called fair VCSP, see [Cooper and Schiex, 2004]).

Algorithm 1 gives two elementary equivalence preserving transformations. Project works on a subproblem defined by the scope of one constraint c_S. Project moves an amount of cost α from c_S to a unary cost function $c_i, i \in S$, for value $a \in d_i$. If the cost α is negative, cost moves from the unary cost function c_i to the cost function c_S. Similarly, ProjectUnary works on a subproblem defined just by one variable $i \in X$. It moves costs from the unary cost function c_i to the nullary cost function $c_\emptyset$.

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

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

Procedure UnaryProject($i, \alpha$)
    ; Precond: $-c_\varnothing \leq \alpha \leq \min\{c_i(a) \ : \ a \in d_i\}$;
    $c_\varnothing \leftarrow c_\varnothing \oplus \alpha$;
    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 $a \in d_i$, $c_i(a) \oplus c_\varnothing \neq \top$, 2) there exists a value $b \in d_i$ such that $c_i(b) = 0$. A WCSP is node consistent (NC) is 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$(\{1,2\}, 1, b, 1)$. 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$(\{1,2\}, 2, a, 1)$, we get the problem 1c. It is immediately AC, but we get no increase in the lower bound $c_\varnothing$.

Given an initial WCSP P, any WCSP obtained from P

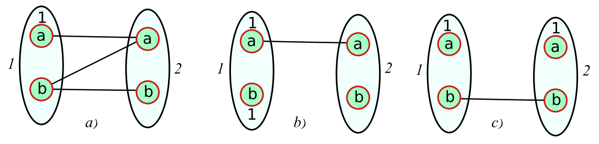


Figure 1: A simple WCSP and two arc consistent closures.

by repeated calls to Project and UnaryProject with positive integer α until a fixpoint is reached is called an arc consistent closure of P. An AC closure of a WCSP P is optimal is 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.

Thus, different alternative definitions of AC have been proposed. They exploit the notion of *full support*. b is a full support if $c_{ij}(a,b) \oplus c_j(b) = 0$. Full supports are support, but because of termination issues [Schiex, 2000], full supports cannot replace supports. This has lead to different proposals:

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 [F. Heras and J. 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 clean heuristics trying to get closer to the optimal closure, but without any guarantee.

4 Optimal arc consistency

We will now relax the definition of an arc consistent closure by shifting from integer costs to rational costs and by allowing to simultaneously apply several equivalence preserving operations.

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

A WCSP is said to be valid if its cost functions remain in the valuation structure. Thus, a negative cost is not valid. We now relax the integrity constraint by explicitly authorizing rational values. 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. 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 $0 \leq w - ij \leq 1$) and suggested determining optimal w_{ij}. [H. Bennaceur and A. 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, as the original Project operator of [Schiex, 2000] allowed, 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_i^S be the sum of all the weights projected from c_S to $c_i(a)$ (remember that weights moved from $c_i(a)$ to c_S are counted as negative in Project). Let u_i be the sum of the weights 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.

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

$$\forall c_S \in C \text{ s.t. } |S| > 1,\ \forall t \in \ell(S) c_S(t) - \sum_{i \in S} p_i^S(t[i]) \geq 0$$

All inequalities for which $c_i(a) = \top$ 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 3 variables subproblems, is actually the basis of the algorithm enforcing 3-cyclic consistency [Cooper, 2004]. In our case, a problem will be said Optimal Soft Arc Consistent 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 however.

To better understand how OSAC works, consider the WCSP in Figure 2. It has 4 variables and is vertically and horizontally symmetric. 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 equivalence preserving operations can modify the problem and the problem is EDAC. However, we may simultaneously perform the following operations:

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

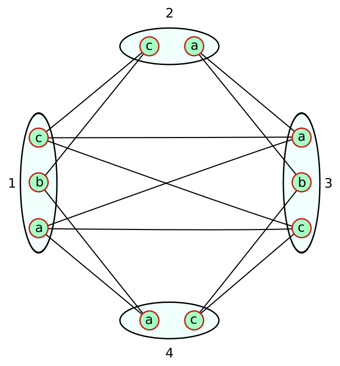


Figure 2: An EDAC WCSP.

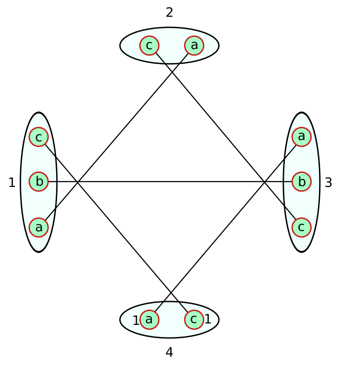


Figure 3: The previous WCSP made OSAC

4. Project$(\{3,4\},4,c,1)$: this moves a unary cost of 1 on $(4,c)$.
5. Project$(\{3,1\},1,a,1)$, Project$(\{3,1\},1,c,1)$: and also two other unary costs of 1 to $(1,c)$ and $(1,a)$.
6. Project$(\{1,2\},1,a,-1)$, Project$(\{1,2\},2,c,1)$: we reimburse our initial loan on value $(c,2)$.
7. Project$(\{1,4\},1,c,-1)$, Project$(\{1,4\},4,a,1)$: we get a second unary cost on value $((4,a)$.

And we get the problem in Figure 3. Just applying UnaryProject$(4,1)$ extracts a lb. $c_\emptyset$ of 1.

5 Properties

The previous theorem 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 $c_\varnothing^Q > c_\varnothing^P$ (where $c_\varnothing^P$, $c_\varnothing^Q$ 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 $c_\varnothing^Q > c_\varnothing^P$, 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, there also exist higher-order consistencies that may change the scopes of problems and provide possibly stronger lower bounds: soft 3-consistency [Cooper, 2005] or soft path inverse consistency [F. Heras and J. Larrosa, 2006b] applied to the previous 2-coloring problem would detect infeasibility.

6 Experiments

The previous OSAC algorithm has been used as a preprocessing algorithm on both random and structured problems. The linear programming problem defined by OSAC is solved using ILOG CPLEX version ?? (using the interior point method). 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 maximizes the number of satisfied constraints. Four different categories of problems with domain size 10 have been generated following the same protocol as in [Larrosa *et al.*, 2005]: sparse loose (SL), sparse tight (ST), dense loose (DL) and dense tight (DT). Samples have 50 instances and we report averages.

	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

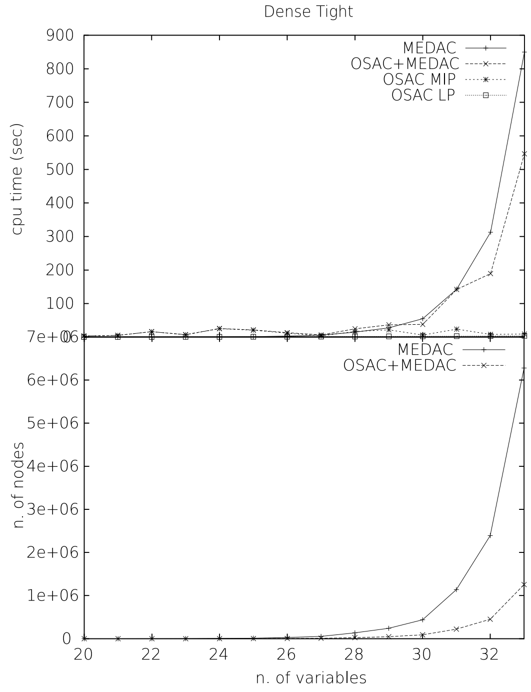
The previous table respectively report the average optimum value, the average values of the EDAC lower bound and the average value of the OSAC lower bound. On loose problems, OSAC and EDAC leave the lower bound unchanged.This shows that higher level local consistencies are required here. However for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

To actually assess the practical interest of OSAC as a preprocessing technique, we tried to solve the preprocessed tight problems. The difficulty here lies in the fact that CPLEX is a floating point solver while our favorite WCSP solver (toolbar[2]) deals with integer costs and cannot read the preprocessed problem. To avoid this we use “fixed point” costs: for all WCSP considered, we first multiply all costs by a large integer constant $\lambda = 1,000$, we then solve the linear programming problem defined by OSAC using integer variables (instead of floating point). The resulting problem

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

[2]http://carlit/cgi-bin/awki.cgi/ToolBarIntro

has integer costs which can be solved by toolbar[3]. Naturally, this means that we shift from a polynomial problem to an NP-hard one. In practice, we found that the problems multiplied by λ usually have a very good linear continuous relaxation and are not too expensive to solve as integer problems. Using a polynomial time rational LP solver would be better.



The Figure above reports cpu-time (up) and size of the tree search (down) for DT problems of increasing size. For the cpu-time, 4 times are reported: (1) OSAC LP is the time taken by CPLEX to solve the first linear relaxation (2) OSAC MIP is the time taken to get an integer solution, (3) MEDAC is the time taken to solve the original problem by mantaining EDAC [Larrosa *et al.*, 2005], (4) OSAC+MEDAC is the sum of OSAC MIP with the time needed by toolbar to solve the preprocessed problem.

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

The second set of benchmarks chosen is defined by some instances of the Radio Link Frequency Assignment Problem of the CELAR [Cabon *et al.*, 1999].[4] These problems have been extensively studied but some instances are still open (see http://www.zib.de/fap/problems/CALMA). This means that the gap between the best upper bounding techniques (usually local search methods) and the best lower bounding techniques (exponential time algorithms) is still not reduced to 0. Motivated by the good results on large problems, we have tried to apply OSAC on these open instances.

	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”

Table 1: A comparison of the strength of OSAC and EDAC on frequency assignment problems

As Table 1 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. These polynomial time lower bounds obtained by simple preprocessing are actually close to the best known (exponential time) lower bounds known. The cpu-time show the cpu-time for the initial OSAC preprocessing. Despite the powerful preprocessing, all problems remained unsolvable. Simultaneously taking into account the strong structure of the problems as in [S. de Givry *et al.*, 2006] would probably lead to significant speedups.

7 Related works

If one assumes that all constraints are binary over boolean domains, WCSP reduces to the weighted MAX-2SAT problem. It is well-known in the SAT community that MAXSAT can be encoded as a pseudo-boolean *linear* formula [de Givry *et al.*, 2003].

One should note that for real-valued costs, MAX-2SAT is equivalent to quadratic pseudo-boolean function optimization [E. Boros and P. Hammer, 2002]: using 0 and 1 to represent respectively true and false, using $\times$ for disjunction, $(1-x)$ for negation and $+$ for combination of costs, a set of weighted 2-clauses can be transformed in a quadratic function. For example, the set of 2-clauses $\{a \vee b, a \vee \bar{b}\}$ with weights 3 and 5 respectively 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 [A. Goldberg and R.E. Tarjan, 1988] in an appropriately defined network [E. Boros and P. 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 maxflow formulation of [E. Boros and P. Hammer, 2002] would be interesting.

8 Conclusion

OSAC provides a polynomial time optimal arc consistent closure in a large class of WCSP. Despite very preliminary

[3]The code of toolbar has been modified to take this into account. If a solution of cost 2λ is known for example and if the current lb. is 1.1λ then backtrack occurs since all global costs in the original problem are integer and the first integer larger or equal to 1.1 is 2, equal to the upper bound

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

testing and relatively expensive enforcing, OSAC already shows its usefulness as a preprocessing algorithm for sufficiently large and tight problems.

Beyond this immediate 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 and orthogonal to the ongoing trend of using higher level consistencies (such as path consistency [Cooper, 2005] or PIC [F. Heras and J. Larrosa, 2006b]). The extension of OSAC to such higher level consistencies is also an open question.

References

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

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

[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. (see arXiv.org/abs/cs.AI/0111038).

[Cooper, 2003] Martin C. Cooper. Reduction operations in fuzzy or valued constraint satisfactio n. *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.*, 2003] Simon de Givry, Javier Larrosa, Pedro Meseguer, and Thomas Schiex. Solving Max-Sat as weighted CSP. In *Proc. of the Ninth International Conference on Principles and Practice of Constraint Programming*, LNCS, Kinsale, Ireland, October 2003. Springer Verlag.

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

[F. Heras and J. 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.

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

[H. Bennaceur and A. Osamni, 2003] H. Bennaceur and A. Osamni. Computing lower bounds for Max-CSP problems. In *Proc.* $16\{th\}$ *International conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 2003)*, 2003.

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

[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 19th IJCAI*, Edinburgh, Scotland, August 2005.

[Larrosa, 2002] J. Larrosa. On arc and node consistency in weighted CSP. In *Proc. AAAI'02*, pages 48–53, Edmondton, (CA), 2002.

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

[S. de Givry *et al.*, 2006] S. de Givry, T. Schiex, and G. Verfaillie. Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP. In *Proc. of the National Conference on Artificial Intelligence, AAAI-2006*, 2006.

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

[Schiex, 2000] T. Schiex. Arc consistency for soft constraints. In *Principles and Practice of Constraint Programming - CP 2000*, volume 1894 of *LNCS*, pages 411–424, Singapore, September 2000.