Modeling and Solving Semiring Constraint Satisfaction Problems by Transformation to Weighted Semiring Max-SAT

Louise Leenen[1], Anbulagan[2], Thomas Meyer[3], and Aditya Ghose[4]

[1] DSL, SCSSE, University of Wollongong, Australia and DPSS, CSIR, South Africa
[2] NICTA* and ANU, Canberra, Australia
[3] Meraka Institute, CSIR, Pretoria, South Africa
[4] DSL, SCSSE, University of Wollongong, Australia

Abstract. We present a variant of the Weighted Maximum Satisfiability Problem (Weighted Max-SAT), which is a modeling of the Semiring Constraint Satisfaction framework. We show how to encode a Semiring Constraint Satisfaction Problem (SCSP) into an instance of a propositional Weighted Max-SAT, and call the encoding Weighted Semiring Max-SAT (WS-Max-SAT). The clauses in our encoding are highly structured and we exploit this feature to develop two algorithms for solving WS-Max-SAT: an incomplete algorithm based on the well-known GSAT algorithm for Max-SAT, and a branch-and-bound algorithm which is complete. Our preliminary experiments show that the translation of SCSP into WS-Max-SAT is feasible, and that our branch-and-bound algorithm performs surprisingly well. We aim in future to combine the natural flexible representation of the SCSP framework with the inherent efficiencies of SAT solvers by adjusting existing SAT solvers to solve WS-Max-SAT.

1 Introduction

The Semiring Constraint Satisfaction Problem (SCSP) framework is an approach to constraint satisfaction and optimization that generalises classical Constraint Satisfaction Problems (CSPs), Partial Constraint Satisfaction Problems, Fuzzy CSPs, Weighted CSPs, Probabilistic CSPs, and others (over finite domains) [1,2,3,4], while also permitting the specification of interesting new instances. Considerable interest has been shown in solving SCSPs. Complete methods exist [5,6,7] but are likely to require exponential time. Bistarelli *et al.* [8] present a prototype SCSP solver based on an incomplete local search method. Wilson [9] shows how to use decision diagrams to solve SCSPs but provides no implementation.

A significant amount of work has been devoted to solving propositional satisfiability (SAT) problems, and specifically the well-known maximum satisfiability problem (Max-SAT) [10,11]. There is continuing interest in translations between

* NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

M.A. Orgun and J. Thornton (Eds.): AI 2007, LNAI 4830, pp. 202–212, 2007.
© Springer-Verlag Berlin Heidelberg 2007

CSPs and SAT problems [12,13,14]. This prompted us to explore the application of methods for solving Max-SAT to SCSPs. We modify the support encoding of CSP to SAT by Gent [12] in order to translate SCSPs as a variant of the Weighted Max-SAT Problem, and call this encoding WS-Max-SAT. Our encoding results in propositional clauses whose structure can be exploited in our algorithms to solve WS-Max-SAT: a local search algorithm that is a modification of the GSAT algorithm for solving Max-SAT [15], and a branch-and-bound (BnB) algorithm.

We show how to formulate a SCSP as a WS-Max-SAT, present a local search and a BnB algorithm, and some experimental results. A comparison of our BnB algorithm with CONFLEX [3], a fuzzy CSP solver, shows that our algorithm performs surprisingly well, and is significantly faster than CONFLEX.

2 Semiring Constraint Satisfaction Problems

The SCSP framework is a popular approach for the representation of constraint satisfaction and in particular, partial constraint satisfaction problems [1].

Definition 1. *A c-semiring is a tuple* $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ *such that*

i) A *is a set with* $\mathbf{0}, \mathbf{1} \in A$*;*
ii) $+$ *is defined over (possibly infinite) sets of elements of* A *as follows a) for all* $a \in A$, $\sum(\{a\}) = a$*; b)* $\sum(\emptyset) = \mathbf{0}$ *and* $\sum(A) = \mathbf{1}$*; c)* $\sum(\bigcup A_i, i \in I) = \sum(\{\sum(A_i), i \in I\})$ *for all sets of indices* I *(flattening property);*
iii) $\times$ *is a commutative, associative, and binary operation with* $\mathbf{1}$ *as unit element and* $\mathbf{0}$ *as absorbing element; iv)* $\times$ *distributes over* $+$ *(i.e., for any* $a \in A$ *and* $B \subseteq A$, $a \times \sum(B) = \sum(\{a \times b, b \in B\})$*).*

The elements of the set A are the preference values to be assigned to tuples of values of the domains of constraints. A partial ordering $\leqslant_S$ over the set A can be defined: $\alpha \leqslant_S \beta$ iff $\alpha + \beta = \beta$.

Definition 2. *A constraint system is a 3-tuple* $CS = \langle S_p, D, V\rangle$*, where* $S_p = \langle A_p, +_p, \times_p, \mathbf{0}, \mathbf{1}\rangle$ *is a c-semiring,* V *is an ordered finite set of variables, and* D *is a finite set containing the allowed values for the variables in* V*. Given* $CS = \langle S_p, D, V\rangle$*, where* $S_p = \langle A_p, +_p, \times_p, \mathbf{0}, \mathbf{1}\rangle$*, a constraint over* CS *is a pair* $c = \langle def_c^p, con_c\rangle$ *where* $con_c \subseteq V$ *is called the signature of the constraint, and* $def_c^p : D^k \to A_p$ *(*k *is the cardinality of* con_c*) is called the value of the constraint.*

Definition 3. *Given* $CS = \langle S_p, D, V\rangle$*, a SCSP over* CS *is a pair* $P = \langle C, con\rangle$ *where* C *is a finite set of constraints over* CS *and* $con = \bigcup_{c \in C} con_c$*. Assume* $\langle def_{c_1}^p, con_c\rangle \in C$ *and* $\langle def_{c_2}^p, con_c\rangle \in C$ *implies* $def_{c_1}^p = def_{c_2}^p$*.*

Definition 4. *Given* $CS = \langle S_p, D, V\rangle$ *with* V *totally ordered via* $\preceq$*, consider any* k*-tuple* $t = \langle t_1, t_2, \ldots, t_k\rangle$ *of values of* D *and two sets* $W = \{w_1, \ldots, w_k\}$ *and* $W' = \{w'_1, \ldots, w'_m\}$ *such that* $W' \subseteq W \subseteq V$ *and* $w_i \preceq w_j$ *if* $i \leq j$ *and* $w'_i \preceq w'_j$ *if* $i \leq j$*. Then the projection of* t *from* W *to* W'*, written* $t \downarrow_{W'}^{W}$*, is defined as the tuple* $t' = \langle t'_1, \ldots, t'_m\rangle$ *with* $t'_i = t_j$ *iff* $w'_i = w_j$*.*

For $CS = \langle S_p, D, V\rangle$, $S_p = \langle A_p, +_p, \times_p, \mathbf{0}, \mathbf{1}\rangle$, *and two constraints* $c_1 = \langle def^p_{c_1}, con_{c_1}\rangle$ *and* $c_2 = \langle def^p_{c_2}, con_{c_2}\rangle$ *over CS, their combination,* $c_1 \otimes c_2$, *is the constraint* $c = \langle def^p_c, con_c\rangle$ *with* $con_c = con_{c_1} \cup con_{c_2}$ *and* $def^p_c(t) = def^p_{c_1}(t \downarrow^{con_c}_{con_{c_1}}) \times_p def^p_{c_2}(t \downarrow^{con_c}_{con_{c_2}})$. *Let* $\bigotimes C$ *denote* $c_1 \otimes ... \otimes c_m$ *with* $C = \{c_1, ..., c_m\}$.

Definition 5. *Given* $CS = \langle S_p, D, V\rangle$, *with* $S_p = \langle A_p, +_p, \times_p, \mathbf{0}, \mathbf{1}\rangle$, *a constraint* $c = \langle def^p_c, con_c\rangle$ *over CS, and a set I of variables* ($I \subseteq V$), *the projection of c over I, written* $c \Downarrow I$, *is the constraint* $c' = \langle def^p_{c'}, con_{c'}\rangle$ *over CS with* $con_{c'} = I \cap con_c$ *and* $def^p_{c'}(t') = \sum_{\{t | t\downarrow^{con_c}_{I\cap con_c} = t'\}} def^p_c(t)$. *Given a SCSP* $P = \langle C, con\rangle$ *over CS, the solution of P is a constraint defined as* $Sol(P) = \bigotimes C$.

Definition 6. *Given a SCSP problem* $P = \langle C, con\rangle$, *consider* $Sol(P) = \langle def^p_c, con\rangle$. *The abstract solution of P is the set* $ASol(P) = \{\langle t, v\rangle \mid def^p_c(t) = v$ *and there is no* t' *such that* $v <_{S_p} def^p_c(t')\}$. *Let* $ASolV(P) = \{v \mid \langle t, v\rangle \in ASol(P)\}$.

3 Max-SAT and Weighted Max-SAT

The SAT problem is to determine whether a given propositional formula has a model (is satisfiable) or is unsatisfiable. The maximum satisfiability problem is the optimisation variant of SAT: given a set of clauses, the problem is to find a variable assignment that maximises the number of satisfied clauses. In *Weighted* Max-SAT a weight is assigned to each clause and the goal becomes to maximise the total weight of the satisfied clauses. More formally, let $C = \{C_1, C_2, \ldots, C_m\}$ be the set of clauses that involve n Boolean variables $x_1, x_2, \ldots, x_n$ that can take the value *true* or *false*, and w_i with $i = 1, \ldots, m$, is the weight assigned to each clause. The objective function is defined as $f(x) = \sum_{i=1}^{m} w_i \cdot I(C_i)$, where $I(C_i)$ is one if and only if the clause C_i is satisfied, and otherwise is zero. When we refer to Max-SAT in the remainder of this paper we assume the weighted version.

Some of the best exact Max-SAT solvers [16,10,11,17,18] implement variants of a BnB algorithm: given a CNF formula θ, a BnB algorithm explores the search tree representing all possible assignments for θ in a depth-first manner. The lower bound (LB) of a node is the sum of the weights of all clauses satisfied by the current partial assignment plus an over-estimation of the sum of the weigths of all clauses that will become satisfied if the current assignment is completed. If LB $\leq$ UB (UB is the upper bound, or best solution found *so far*), the subtree below the current node is pruned, and the algorithm backtracks to a node at a higher level in the tree. If LB $>$ UB, the algorithm tries to find a better solution by extending the current partial solution by instantiating one more variable. The solution is the value of UB after the whole tree has been explored.

4 Weighted Semiring Max-SAT

In this section we show how to encode a SCSP into a WS-Max-SAT. Our encoding of a SCSP into a WS-Max-SAT is based on the support encoding of CSPs

into Boolean SAT [12]. Every propositional clause in our encoding will receive a semiring value from the semiring associated with the SCSP as a weight.

For the remainder of this paper let $P = \langle C, con \rangle$ be a SCSP over a constraint system $CS = \langle S_p, D, V \rangle$, such that every variable $i \in V$ has $D = \{d_1, \ldots, d_d\}$ as its domain, and $S_p = \langle A_p, +_p, \times_p, \mathbf{0}, \mathbf{1} \rangle$. Assume the cardinality of V is n, the cardinality of C is m, and that there are s semiring values in the set A_p. Every constraint $c \in C$, $c = \langle def_c^p, con_c \rangle$ with r the cardinality of con_c and $con_c \subseteq V$.

For every SCSP variable $i \in V$, we have d WS-Max-SAT variables: the WS-Max-SAT variable $X_{i,j}$ is true if the SCSP variable i has the value $d_j \in D$ with $j = 1...d$. Our *WS-Max-SAT encoding* (encoding of P) has three kinds of clauses:

1. *at-least-one* clauses to ensure that each SCSP variable is assigned at least one value from its domain;
2. *at-most-one* clauses to ensure that each SCSP variable is assigned at most one value from its domain;
3. *support* clauses to represent all the possible value pairs (tuples) with their associated semiring values for every constraint. For every constraint we have at most s such clauses, and every clause receives the associated semiring value as its weight.

– For every variable $i \in V$ we have one *at-least-one* clause:
$X_{i,1} \vee X_{i,2} \vee \ldots \vee X_{i,d}$.
– For every variable $i \in V$ we have $0.5[d(d-1)]$ *at-most-one* clauses: one clause, $\neg X_{i,v} \vee \neg X_{i,w}$, for every pair (d_v, d_w), where $1 \leq v < w \leq d$.
– For every constraint, we have at most s support clauses: one clause to represent all the tuples that have the same associated semiring value. Suppose $A_p = \{p_1, p_2, \ldots, p_s\}$, then every constraint $c \in C$ has at most s support clauses, $sup_c^{p_t}$ with $t = 1, \ldots, s$. A support clause $sup_c^{p_t}$ for a constraint $c = \langle def_c^p, con_c \rangle$ is a disjunction of the representation of all tuples $\langle X_{c_1,v_1}, \ldots, X_{c_r,v_r} \rangle$ (the conjunction of all these literals) that has the associated semiring value p_t, $c_i \in con_c$, and $v_i \in D$ for $i = 1 \ldots r$.

All the clauses except the support clauses have to be satisfied and we call these clauses the *hard* clauses. The minimum semiring value, $\mathbf{0}$, is assigned as the weight of the negation of each hard clause. If a hard clause is not satisfied by an assignment, this unsatisfied clause contributes the minimum semiring value to the total weight. Note that at most one support clause for each constraint can be satisfied by a truth assignment if we assume that the *at-most-one* and *at-least-one* clauses are all satisfied. We regard the support clauses as soft clauses: the weight assigned to each support clause is the combined semiring value of the tuples represented in that support clause.

Definition 7. *Given a SCSP P and a WS-Max-SAT encoding of P, let the set Sup contain the support clauses in the encoding and α be some truth assignment for the clauses in Sup. Then $t_{sol}^{\alpha} = \langle X_{1,val_1}, \ldots, X_{n,val_n} \rangle$ is a solution tuple for the encoding iff the following holds:*

– *for every constraint $c \in C$ exactly one of its support clauses, sup_c^k, ($1 \leq k \leq p_s$) is satisfied by α. This support clause represents a disjunction of tuples*

of this constraint, each as a conjunction of WS-Max-SAT variables, with exactly one tuple, $t_c = \langle X_{c_1}, \ldots, X_{c_r} \rangle$ *with* $c_i \in con_c$, $v_i \in D$, $i = 1 \ldots r$, *and such that all* $X_{c_i,v_i} = \text{true}$, *and*

- *if* c_i *is equal to* j $(j = 1, \ldots, n)$ *then* $v_i = val_j$.

The semiring value associated with t^{α}_{sol} *is* $sr^{\alpha}_{sol} = sp_1 \times_p \ldots \times_p sp_m$ *where* $sp_c = k$ *and* sup^k_c *is the support clause satisfied by assignment* α *for a constraint* c.

Definition 8. *Given a SCSP* P, *let* Cl *be the set of clauses that represents the WS-Max-SAT encoding of* P. *Let* $Cl_h = \langle cl_1, \ldots cl_h \} \subseteq Cl$ *be the clauses that are satisfied by some truth assignment* α. *The WS-Max-SAT problem is to find max* $f(\alpha) = \bigotimes_{i=1}^{h} w_i$, *where* $w_i \in A_p$ *is the weight assigned to clause* $i \in Cl_h$, $f(\alpha)$ *is maximal with respect to the partial order* $<_{S_p}$, *and* $t^{\alpha}_{sol} = \langle X_{1,val_1}, \ldots, X_{n,val_n} \rangle$ *is the solution tuple for the encoding* Cl. *The ordered pair* $\langle \langle val_1, \ldots, val_n \rangle, f(\alpha) \rangle \in ASol(P)$.

Example 1. Suppose we have a SCSP $P = \langle \{c_1, c_2, c_3\}, V \rangle$ over $\langle S_p, D, V \rangle$, with $V = \{A, B, C\}$ and $D = \{1, 2, 3\}$. The three constraints are explicitly defined by $c_1 : A < B$, $c_2 : B < C$, and $c_3 : C < A$. The semiring is $S_p = \langle \{0, 5, 10\}, max, min, 0, 10 \rangle$. See the constraint definitions in Table 1. Note

Table 1. Constraint definitions

t	$c_1 : A < B$	$c_2 : B < C$	$c_3 : C < A$
$\langle 1,1 \rangle$	5	5	5
$\langle 1,2 \rangle$	10	10	10
$\langle 1,3 \rangle$	10	10	10
$\langle 2,1 \rangle$	0	0	0
$\langle 2,2 \rangle$	5	5	5

t	$c_1 : A < B$	$c_2 : B < C$	$c_3 : C < A$
$\langle 2,3 \rangle$	10	10	10
$\langle 3,1 \rangle$	0	0	0
$\langle 3,2 \rangle$	0	0	0
$\langle 3,3 \rangle$	5	5	5

that the first coordinate reflects the value of the first variable in a constraint and the second coordinate reflects the value of the second variable. For example, the preference value associated with the tuple $\langle 1, 3 \rangle$ for constraint c_1 is when A has the value 1 and B has the value 3. Tuples that satisfy the constraints get the highest preference value. Among the tuples that do not satisfy the constraints, we prefer those where the first coordinate equals the second coordinate. These tuples get a preference value of 5, and all the remaining tuples get the worst value, 0. The WS-Max-SAT encoding of the problem follows below.

At-least-one clauses: $A_1 \vee A_2 \vee A_3$, $B_1 \vee B_2 \vee B_3$, and $C_1 \vee C_2 \vee C_3$
At-most-one clauses: $\neg A_1 \vee \neg A_2$, $\neg A_2 \vee \neg A_3$, $\neg A_1 \vee \neg A_3$, $\neg B_1 \vee \neg B_2$, $\neg B_2 \vee \neg B_3$, $\neg B_1 \vee \neg B_3$, $\neg C_1 \vee \neg C_2$, $\neg C_2 \vee \neg C_3$, and $\neg C_1 \vee \neg C_3$
Support clauses:
$c_1^{10} : [(A_1 \wedge B_2) \vee (A_1 \wedge B_3) \vee (A_2 \wedge B_3)]$, $c_1^{5} : [(A_1 \wedge B_1) \vee (A_2 \wedge B_2) \vee (A_3 \wedge B_3)]$,
$c_1^{0} : [(A_2 \wedge B_1) \vee (A_3 \wedge B_1) \vee (A_3 \wedge B_2)]$, $c_2^{10} : [(B_1 \wedge C_2) \vee (B_1 \wedge C_3) \vee (B_2 \wedge C_3)]$,
$c_2^{5} : [(B_1 \wedge C_1) \vee (B_2 \wedge C_2) \vee (B_3 \wedge C_3)]$, $c_2^{0} : [(B_2 \wedge C_1) \vee (B_3 \wedge C_1) \vee (B_3 \wedge C_2)]$,
$c_3^{10} : [(C_1 \wedge A_2) \vee (C_1 \wedge A_3) \vee (C_2 \wedge A_3)]$,

$c_3^5 : [(C_1 \wedge A_1) \vee (C_2 \wedge A_2) \vee (C_3 \wedge A_3)]$,
$c_3^0 : [(C_2 \wedge A_1) \vee (C_3 \wedge A_1) \vee (C_3 \wedge A_2)]$

In our example, each support clause has its associated semiring value as its weight. The negation of all the remaining clauses are each given a weight of 0:

1. $\neg A_1 \wedge \neg A_2 \wedge \neg A_3$	2. $\neg B_1 \wedge \neg B_2 \wedge \neg B_3$	3. $\neg C_1 \wedge \neg C_2 \wedge \neg C_3$
4. $A_1 \wedge A_2$	5. $A_2 \wedge A_3$	6. $A_1 \wedge A_3$
7. $B_1 \wedge B_2$	8. $B_2 \wedge B_3$	9. $B_1 \wedge B_3$
10. $C_1 \wedge C_2$	11. $C_2 \wedge C_3$	12. $C_1 \wedge C_3$

We want to find an assignment that does not satisfy any of the (hard) clauses numbered 1 to 12. A truth assignment that satisfies one or more of the support clauses with an associated semiring value of 10 is preferred over one with lower semiring values. We can prove that our encoding is correct.

Theorem 1. *Let SCSP P be a set of clauses C_l containing a WS-Max-SAT encoding for P, and a truth assignment α.*

- *If $t_{sol}^{\alpha} = \langle X_{1,val_1}, \ldots, X_{n,val_n} \rangle$ is a solution tuple for the WS-Max-SAT encoding of P, then $t = \langle X_1, \ldots, X_n \rangle$ with $X_i = val_i$ for $i = 1, \ldots, n$ is a solution tuple for P.*
- *If sr_{sol}^{α} is the semiring value associated with t_{sol}^{α}, then the associated semiring value of t is sr_{sol}^{α}.*

5 Algorithms to Solve WS-Max-SAT

Most SAT and Max-SAT solvers require the propositional clauses to be in conjunctive normal form (CNF). In our encoding, the propositional clauses are highly structured and we do not have to convert them into CNF. We only allow truth assignments where the *at-least-one-variable* and *at-most-one-variable* clauses are satisfied in our implementation. This leaves only the support clauses that have to be checked for satisfiability: exactly one support clause can be satisfied for every constraint. We simply search for the support clause with the best associated semiring value for each constraint under the current truth assignment.

5.1 The GSAT Algorithm for WS-Max-SAT

GSAT [15] is a greedy algorithm that tries to maximise the number of satisfied clauses by selecting different variable assignments based on the *score* of a variable x under the current assignment α: this is defined as the difference between the weight of the clauses *un*satisfied by α and the assignment obtained by flipping x in α. GSAT is not a complete algorithm and can get stuck in local minima. GSAT starts with a random truth assignment. After a maximum number of allowed flips has been performed, a new random truth assignment is generated. This improves the probability to find a solution.

We have adjusted GSAT, as presented in Algorithm 1, to solve a WS-Max-SAT problem for SCSPs with binary constraints. The procedure *chooseVariable*

Algorithm 1. GSAT for WS-Max-SAT

Require: C_l (WS-Max-SAT encoding); W, (set of weights); S_p, (semiring); *MaxFlips*, *MaxSteps*

```
Initialise α_Best = 0;
for i = 1 to MaxSteps do
    Let α = a randomly generated truth assignment for C_l;
    if f(α_BEST) <_{S_p} f(α) then
        α_BEST = α;
    for j = 1 to MaxFlips do
        if f(α_BEST) <_{S_p} f(α) then
            α_BEST = α;
        if f(α) = 1 then
            return α
        else
            x = chooseVariable(C_l,α);
            α = α with truth value of x flipped;
return α_BEST;
```

selects the next variable to be flipped by considering the *score* of each variable x in WS-Max-SAT. The score of a variable x in an assignment α is the f value a truth assignment α_x has if α_x is identical to α except for the truth value assigned to x. There may be more than one variable with a maximum score. In this case any variable with a maximal f value is chosen at random to be flipped. When a WS-Max-SAT variable $X_{i,j}$ is chosen to be flipped, we consider two cases:

- The current value of $X_{i,j} = true$ and has to be flipped to become $false$. Some variable $X_{i,l}$ with $l = 1, \ldots, d$ and $l \neq j$ is chosen at random to become *true*.
- The current value of $X_{i,j} = false$ and has to be flipped to become *true*. Some variable $X_{i,l}$ with $l = 1, \ldots, d$ and $l \neq j$ is currently *true* and is given the value $false$.

Example 2. Suppose we generate the following random truth assignment α_1: $A_1 = A_2 = B_1 = B_2 = C_1 = C_3 = false$ and $A_3 = B_3 = C_2 = true$. The following three clauses are satisfied: c_1^5, c_2^0, and c_3^{10} with $f(\alpha_1) = min(5, 0, 10) = 0$. The procedure *chooseVariable* finds that all flips result in f values of 0, so it randomly selects C_1 to be flipped. Now $C_1 = true$ and $C_2 = false$. In the next iteration it finds that C_3 has a score of 5 and it flips C_3: in this case the three clauses c_1^5, c_2^5, and c_3^5 with $f(\alpha_3) = min(5, 5, 5) = 5$.

5.2 The BnB Algorithm for WS-Max-SAT

We present a BnB algorithm to solve WS-Max-SAT, as sketched in Algorithm 2. The variables are ordered according to their membership in the signatures of the constraints. We place the variables in the signature of the first constraint on a queue, and then check whether these variables appear in the signatures of other constraints. If they do, the unlisted variables in the others constraints' signatures are also placed on the queue. Repeat this step until all constraints' signatures

Algorithm 2. BnB(NoInstantiated,LB)

Require: *Input variables:* C_l, (WS-Max-SAT encoding); W, (set of weights); S_p (a semiring); N (number of variables)
Global variables: Queue (variables in order of instantiation); *UB* (upper bound)
Value parameters NoInstantiated; *LB* (lower bound)

```
1: if (NoInstantiated < N) then
2:   var = pop Queue;
3:   while (var_domain not empty) do
4:     var_value = finds best value from var_domain;
5:     var_domain = var_domain - var_value;
6:     NewLB = lower bound for current node;
7:     NewLB = ×_p(LB, NewLB);
8:     if (UB <_Sp NewLB) then
9:       LB = NewLB;
10:      if (NoInstantiated = N-1) then
11:        UB = LB;
12:        Best-Solution = current assignment;
13:        if (UB = 1) then
14:          return 1;
15:        Prune all tuples with associated semiring value ≤_Sp UB;
16:      if ( BnB(Noinstantiated+1,LB) = 1) then
17:        return 1;
18: return 0;
```

have been checked. Every variable has a *main constraint* which is the constraint where the variable has been identified to be placed on the queue. To find a value for a variable we search among the tuples of its main constraint for a tuple with a maximal associated semiring value under the current partial assignment. From this chosen tuple we get a value to instantiate the current variable.

After a variable has been instantiated, we calculate a lower bound value for the current node in the search tree, i.e. we calculate an estimated semiring value for the variable-value tuple of the current (in most cases, partial) assignment. The lower bound is computed by looking at constraints whose signatures have been entirely instantiated by the current (possibly partial) solution and combining the semiring values assigned to the projection of the current solution to the signatures of these constraints. For all the remaining constraints (i.e with uninstantiated variables), we estimate the maximal semiring value for the purpose of finding a lower bound. The upper bound is initialised with the worst semiring value and the lower bound with the best semiring value. The variable *var* is instantiated with the next variable from *Queue* and *var_domain* contains the domain values for that variable.

6 Experimental Setting and Results

We used an Intel Pentium 4 processor at 2.53GHz with 512MB RAM.

6.1 Results of GSAT-Based Algorithm

We solved three sets of randomly generated binary fuzzy CSPs where each set contains 100 instances. Each instance has 10 domain values. Instances in Set 1 has 80 variables and 10 constraints, in Set 2 100 variables and 10 constraints, and in Set 3 has 120 variables and 20 constraints.

In each set of 100 instances, 50 instances have a tightness of 70% and the other 50 instances have a tightness of 90%. A tightness (T) of x% means that (100-x)% of the possibles tuples have been assigned the maximum semiring value. All the problems have the following semiring values, $A_p = \{0, 0.3, 0.5, 0.8, 1\}$, the comparative operator is *max* and the combination operator is *min*. The results are in Table 2. In the third row (S=1) we show for how many instances the algorithm found a maximal solution (with a solution tuple that has a semiring value of 1). In the next rows we show for how many instances we found solution tuples with semiring values of 0.8, 0.5 and 0.3, respectively.

Table 2. Results for the GSAT-based algorithm

Set	1	1	2	2	3	3
T	70%	90%	70%	90%	70%	90%
S = 1	50	2	50	0	1	0
S = 0.8	0	48	0	49	33	0
S = 0.5	0	0	0	1	16	38
S = 0.3	0	0	0	0	0	12
steps	1.34	2.00	1.28	n/a	1	n/a
flips	452.4	836.5	438.5	n/a	329.0	n/a

There were no instances where a solution with the minimum semiring value was found. The last two rows contain information for those instances where the maximal solution were found: it shows on average when the solution was found. We show during which step, and after how many flips, the algorithm halted. Each step consists of a maximum of 1000 flips.

All the problems instances have maximal solutions apart from three instances in set 3 with tightness 90% (last column). These three instances have a best solution with a combined semiring value of 0.8. Our GSAT algorithm performs reasonably well for a relatively small number of steps and flips.

6.2 Results of BnB Algorithm

We solved two sets of randomly generated binary fuzzy CSPs with CONFLEX [3] and with our BnB algorithm. We used the CONFLEX executable for Windows downloaded from the CONFLEX website.[1] The 40 instances in Set 4 has 10 variables, 10 constraints, 10 domain values (20 instances with tightness 60% and 20 with tightness 80%). The 20 instances in Set 5 has 15 variables, 10

[1] http://www.inra.fr/internet/Departements/MIA/T//conflex/adressesConflex.html

Table 3. Average runtime for Sets 4 and 5 (in seconds)

Set	4	4	5	5
Tightness	60%	80%	60%	80%
CONFLEX	12.34	19.89	154.42	165.14
BnB	0.44	0.29	0.05	0.26

Table 4. Runtime for the BnB algorithm (in seconds)

Set	1	1	2	2	3	3	5
T	70%	90%	70%	90%	70%	90%	90%
Runtime	0.27	0.48	0.35	0.52	0.86	1.52	5.30

constraints, 10 domain values (10 instances with tightness 60% and 10 instances with tightness 80%). Table 3 gives the results.

We experienced difficulty in running larger problems on the Windows version of the CONFLEX software. However, it is clear from these experiments that our BnB algorithm is considerably faster than CONFLEX.

We also solved the three sets of problems of Table 2 and a fifth set containing 50 instances, where each instance has 150 variables, 30 constraints and a domain size of 20. All the problems in Set 5 have a maximal solution. See Table 4.

7 Conclusion and Future Work

We define a WS-Max-SAT problem by translating a SCSPs into propositional weighted Max-SAT problem. We present an incomplete GSAT-based algorithm to show that this translation is feasible to solve SCSPs. We also present a BnB algorithm that performs surprisingly well, outperforming the CONFLEX system. A next step is to investigate the adaptation of existing efficient Weighted Max-SAT algorithms to solve WS-Max-SAT. We also plan to test our BnB algorithm by solving benchmark problems.

References

1. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint solving and optimization. Journal of the ACM 44(2), 201–236 (1997)
2. Wilson, M., Borning, A.: Hierarchical constraint logic programming. Journal of Logic Programming 16, 277–318 (1993)
3. Dubois, D., Fargier, H., Prade, H.: The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In: Proc. of IEEE Conf on Fuzzy Syst. (1993)
4. Fargier, H., Lang, J.: Uncertainty in constraint satisfaction problems: a probabilistic approach. In: Moral, S., Kruse, R., Clarke, E. (eds.) ECSQARU 1993. LNCS, vol. 747, Springer, Heidelberg (1993)
5. Georget, Y., Codognet, P.: Compiling semiring-based constraint with clp(FD,s). In: Maher, M.J., Puget, J.F. (eds.) CP 1998. LNCS, vol. 1520, Springer, Heidelberg (1998)

6. Rossi, F., Pilan, I.: Abstracting soft constraints: Some experimental results. In: Proc. of Joint Annual Workshop of ERCIM Working Group on Constraints and the CologNET area on Constraint and Logic Programming (2003)
7. Bistarelli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G.: Semiring-based CSPs and valued CSPs: Basic properties and comparison. Constraints 4, 199–240 (1999)
8. Bistarelli, S., Fung, S., Lee, J., Leung, H.: A local search framework for semiring-based constraint satisfaction problems. In: Soft 2003 (2003)
9. Wilson, N.: Decision diagrams for the computation of semiring valuations. In: IJCAI 2005 (2005)
10. Xing, Z., Zhang, W.: MaxSolver: An efficient exact algorithm for (weighted) maximum satisfiability. Artificial Intelligence 164(1-2), 47–80 (2005)
11. Alsinet, T., Manya, F., Planes, J.: Improved exact solver for weighted Max-SAT. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 371–377. Springer, Heidelberg (2005)
12. Gent, I.P.: Arc consistency in SAT. In: ECAI 2002 (2002)
13. Walsh, T.: SAT v CSP. In: Dechter, R. (ed.) CP 2000. LNCS, vol. 1894, pp. 441–456. Springer, Heidelberg (2000)
14. Bennaceur, H.: The satisfiability problem regarded as a constraint satisfaction problem. In: ECAI 1996, pp. 155–159 (1996)
15. Selman, B., Levesque, H., Mitchell, D.: A new method for solving hard satisfiability problems. In: AAAI 1992, pp. 440–446 (1992)
16. Li, C.M., Manyà, F., Planes, J.: Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In: AAAI 2006 (2006)
17. de Givry, S., Larrosa, J., Meseguer, S.T.: Solving Max-SAT as weighted CSP. In: Rossi, F. (ed.) CP 2003. LNCS, vol. 2833, Springer, Heidelberg (2003)
18. Larossa, J., Heras, F.: Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In: IJCAI 2005 (2005)