Semiring-Induced Propositional Logic: Definition and Basic Algorithms * †

Javier Larrosa, Albert Oliveras and Enric Rodríguez-Carbonell
Technical University of Catalonia
Barcelona, Spain
larrosa,oliveras,erodri@lsi.upc.edu

Abstract

In this paper we introduce an extension of propositional logic that allows clauses to be weighted with values from a generic semiring. The main interest of this extension is that different instantiations of the semiring model different interesting computational problems such as *finding a model*, *counting the number of models*, *finding the best model* with respect to an objective function, *finding the best model* with respect to several independent objective functions, or *finding the set of pareto-optimal models* with respect to several objective functions.

Then we show that this framework unifies several solving techniques and, even more importantly, rephrases them from an algorithmic language to a logical language. As a result, several solving techniques can be trivially and elegantly transferred from one computational problem to another. As an example, we extend the basic DPLL algorithm to our framework producing an algorithm that we call SDPLL. Then we enhance the basic SDPLL in order to incorporate the three features that are common in all modern SAT solvers: *backjumping*, *learning* and *restarts*.

As a result, we obtain an extremely simple algorithm that captures, unifies and extends in a well-defined logical language several techniques that are valid for arbitrary semirings.

Keywords: semiring, marginalization problem, DPLL

1 Introduction

The importance of semirings to unify apparently unrelated combinatorial computational problems has been known and studied for a long time [28, 27, 5, 19, 1]. Some well-known unifiable problems occur in (soft) *constraint networks*, *probabilistic networks* or *relational databases*, each of them having many real-life domains of application.

There are many advantages for semiring unification. On the one hand, it provides a very general formalism for algorithmic development: instead of re-discovering the same technique for each particular type of problem, it can be formulated in an abstract form and immediately applied to any problem that fits into the framework (e.g. Adaptive Consistency [12], Directional Resolution [11], Nonserial Dynamic Programming [3], the basic pseudo-boolean method [9], and many others [1] are essentially independent developments of the same algorithm). On the other hand, the unification provides a convenient formalism for algorithmic generalization (see e.g. [8] and [29]).

In this paper we study this idea in the context of propositional logic. First of all, we extend propositional logic by incorporating a generic semiring and allowing boolean formulas to be weighted with semiring values. We define its semantics and extend classical notions such as logical implication ($\models$) or equivalence ($\equiv$).

Then we show that semiring-induced propositional logic can model in a natural way very important combinatorial problems over boolean variables such as *finding a model* (SAT), *counting the number of*

*Partially supported by the Spanish Ministry of Science and Innovation through the projects TIN2006-15387-C03-02 and TIN2007-68093-C02-01.

†A version of this paper extended with proofs is available at http://www.lsi.upc.edu/~erodri/lpar16ex.pdf

semiring	A	$\oplus$	$\otimes$	$\mathbf{0}$	$\mathbf{1}$	applic.
$\mathscr{A}_{\mathtt{bool}}$	$\{0,1\}$	$\vee$	$\wedge$	0	1	SAT
$\mathscr{A}_{\mathtt{count}}$	$\mathbb{R}^+$	$+$	$\times$	0	1	#SAT
$\mathscr{A}_{\mathtt{max}\times}$	$\mathbb{R}^+$	max	$\times$	0	1	Max-SAT
$\mathscr{A}_{\mathtt{min}+}$	$\mathbb{R}^+ \cup \{\infty\}$	min	$+$	∞	0	Max-SAT
$\mathscr{A}^n$	A^n	$\oplus^n$	$\otimes^n$	$(\mathbf{0},\ldots,\mathbf{0})$	$(\mathbf{1},\ldots,\mathbf{1})$	
$\mathscr{A}^f$	A^f	$\oplus^f$	$\otimes^f$	$\{\mathbf{0}\}$	$\{\mathbf{1}\}$	

Figure 1: The first four rows show four different semirings with immediate application. The last two rows show two different semiring constructors (multidimensional and frontier extensions) which are relevant to model multi-criteria problems.

models (#SAT), *finding the best model with respect to an objective function*, *finding the best model with respect to several independent objective functions* or *finding the set of pareto-optimal models with respect to several objective functions*.

The principal advantage of the extension is conceptual because techniques for these problems are mostly defined in a procedural way and it is difficult to see the logic that is behind the execution of the procedure (see e.g. [7, 13, 30]). With our approach, solving techniques can be explained in logical terms. As an example, we extend the basic DPLL algorithm [10] to semiring-induced logic producing an algorithm that we call SDPLL. When SDPLL is instantiated with different semirings to model, for example, SAT, #SAT or Max-SAT, it is faithful to the simplest algorithms for each problem [10, 4, 7]. Therefore, we show that these algorithms were in fact the same algorithm *modulo the corresponding semiring*. Additionally, it immediately provides basic enumeration algorithms to not-so-studied problems such as multi-objective model optimization. Then, we enhance the basic SDPLL with three features that are present in all modern SAT solvers [22]: backjumping, learning and restarts. Thus, we show that they are also valid in our much more general framework.

2 Semirings

In this paper, a semiring $\mathscr{A} = (A, \oplus, \otimes)$ consists of a non-empty set A together with two binary operations $\oplus$ and $\otimes$ such that both operations are *commutative* and *associative*, and $\otimes$ *distributes over* $\oplus$[1].

If there is an element $\mathbf{0} \in A$ such that $\mathbf{0} \oplus a = a$ and $\mathbf{0} \otimes a = \mathbf{0}$ for all $a \in A$ then A is a semiring with *zero* element. Similarly, if there is an element $\mathbf{1} \in A$ such that $\mathbf{1} \otimes a = a$ for all $a \in A$ then A is a semiring with *unit* element. It can be assumed without loss of generality that a semiring has a zero element, as noted in [19]. Semirings admit at most one zero and one unit element.

Given a semiring $\mathscr{A}$, a binary relation $\leq_{\mathscr{A}}$ can be defined as follows: for any $a, b \in A$, $a \leq_{\mathscr{A}} b$ holds if there exists $c \in A$ such that $a \oplus c = b$. This relation can be shown to be a pre-order [19]; i.e., *i*) for all $a \in A$, $a \leq_{\mathscr{A}} a$ (reflexivity), and *ii*) if $a \leq_{\mathscr{A}} b$ and $b \leq_{\mathscr{A}} c$ then $a \leq_{\mathscr{A}} c$ (transitivity). In this paper we will restrict ourselves to semirings with zero and unit elements, noted $\mathscr{A} = (A, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, whose pre-order is a partial order (i.e., it holds that $a \leq_{\mathscr{A}} b$ and $b \leq_{\mathscr{A}} a$ implies $a = b$).

The semiring order also has the properties that *iii*) $a \leq_{\mathscr{A}} b$ and $a' \leq_{\mathscr{A}} b'$ imply $a \oplus a' \leq_{\mathscr{A}} b \oplus b'$ and $a \otimes a' \leq_{\mathscr{A}} b \otimes b'$; and *iv*) for all $a \in A$, $\mathbf{0} \leq_{\mathscr{A}} a$. As a consequence, $\oplus$ increases monotonically (i.e., $a \leq_{\mathscr{A}} a \oplus b$); and when applied to values smaller than or equal to $\mathbf{1}$, then $\otimes$ decreases monotonically (i.e., if $a, b \leq_{\mathscr{A}} \mathbf{1}$ then $a \otimes b \leq_{\mathscr{A}} a$). As usual, $a \neq b$ and $a \leq_{\mathscr{A}} b$ will be noted as $a <_{\mathscr{A}} b$.

[1] This definition corresponds to what is called a *commutative semiring* elsewhere [15].

The first four rows of Figure 1 summarize well-known semirings that will be used to highlight the expressivity of semiring-induced logic. The first column indicates the semiring name, columns $2-6$ show their components and column 7 indicates their paradigmatic application (to be seen in Section 4). In all of them the induced order $\leq_{\mathscr{A}}$ is the usual (total) order, except for $\mathscr{A}_{\texttt{min+}}$ where it is reversed (e.g. $5 \leq_{\mathscr{A}_{\texttt{min+}}} 2$).

Sometimes it is useful to derive a new semiring from an already existing one. In the following we consider two useful extensions (they are summarized in the last two rows of Figure 1). The *multidimensional extension* generates a new semiring whose values are vectors of semiring values.

Definition 1. *Let $\mathscr{A} = (A, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ be a semiring. Its* multidimensional extension *[19] is $\mathscr{A}^n = (A^n, \oplus^n, \otimes^n, \mathbf{0}^n, \mathbf{1}^n)$ where*

- $A^n = A \times \ldots \times A$
- $(a_1, \ldots, a_n) \oplus^n (b_1, \ldots, b_n) = (a_1 \oplus b_1, \ldots, a_n \oplus b_n)$
- $(a_1, \ldots, a_n) \otimes^n (b_1, \ldots, b_n) = (a_1 \otimes b_1, \ldots, a_n \otimes b_n)$
- $\mathbf{0}^n = (\mathbf{0}, \ldots, \mathbf{0})$
- $\mathbf{1}^n = (\mathbf{1}, \ldots, \mathbf{1})$

In this case, $(a_1, \ldots, a_n) \leq_{\mathscr{A}^n} (b_1, \ldots, b_n)$ if and only if $\forall_{1 \leq j \leq n},\ a_j \leq_{\mathscr{A}} b_j$. Observe that if $\leq_{\mathscr{A}}$ is a total order then $\leq_{\mathscr{A}^n}$ is the usual product order in a product of posets. If $a \leq_{\mathscr{A}^n} b$ one says that b *dominates* a.

Given a semiring, the *frontier extension* [6] generates a new semiring whose values are sets of non-dominated values from the original semiring.

Definition 2. *Let $\mathscr{A} = (A, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ be a semiring. The set of non-dominated elements of $S \subseteq A$ is defined as*

$$||S|| = \{v \in S \mid \forall_{w \in S}\ v \not< w\}\ .$$

The frontier extension *of $\mathscr{A}$ is $\mathscr{A}^f = (A^f, \oplus^f, \otimes^f, \mathbf{0}^f, \mathbf{1}^f)$, where*

- $A^f = \{||S|| \mid S \subseteq A\}$
- $S \oplus^f R = ||S \cup R||$
- $S \otimes^f R = ||\{a \otimes b \mid a \in S, b \in R\}||$
- $\mathbf{0}^f = \{\mathbf{0}\}$
- $\mathbf{1}^f = \{\mathbf{1}\}$

In this case $S \leq_{\mathscr{A}^f} R$ holds if and only if for all $a \in S$ there is $b \in R$ such that $a \leq_{\mathscr{A}} b$. This is the so-called *frontier order* widely used in multi-objective optimization.

Example 1. *Consider semiring $\mathscr{A}_{\texttt{bool}} = (\{0,1\}, \vee, \wedge, 0, 1)$. Its bidimensional extension $\mathscr{A}^2_{\texttt{bool}}$ is the set of two-dimensional bit vectors such as $a = (0,1)$ and $b = (1,1)$. Note that $a \leq_{\mathscr{A}^2_{\texttt{bool}}} b$ as $0 \leq_{\mathscr{A}_{\texttt{bool}}} 1$ and $1 \leq_{\mathscr{A}_{\texttt{bool}}} 1$, $a \vee^2 b = (0 \vee 1, 1 \vee 1) = (1,1)$ and $a \wedge^2 b = (0 \wedge 1, 1 \wedge 1) = (0,1)$. The frontier extension of $\mathscr{A}^2_{\texttt{bool}}$ is $(\mathscr{A}^2_{\texttt{bool}})^f$. Its values are sets of non-dominated two-dimensional bit vectors such as $a = \{(0,1),(1,0)\}$ or $b = \{(1,1)\}$. The set $\{(0,1),(1,1)\}$ does not belong to the semiring as $(1,1)$ dominates $(0,1)$. Note that $a \leq_{(\mathscr{A}^2_{\texttt{bool}})^f} b$ as every element of a is dominated by an element of b. Furthermore, $a \vee^{2f} b = ||\{(0,1),(1,0),(1,1)\}|| = \{(1,1)\}$ and $a \wedge^{2f} b = \{(0,1),(1,0)\}$.*

3 Semiring-Induced Propositional Logic

3.1 Syntax

Let P be a finite set of propositional symbols that will remain fixed throughout the paper. If $p \in P$, then p and $\neg p$ are *literals*. The *negation* of a literal l, written $\neg l$, denotes $\neg p$ if l is p, and p if l is $\neg p$. A *clause* C is a (possibly empty) finite disjunction of literals. A *unit clause* consists of a single literal. The *empty clause* is noted $\square$.

A (partial truth) *assignment* M is a set of literals such that if l is in M, then $\neg l$ is not. A literal l is *true* in M if $l \in M$, is *false* in M if $\neg l \in M$, and is *undefined* in M otherwise. The assignment M is *total* if every symbol of P is defined in M. The set of total assignments is noted $\mathscr{M}$.

An assignment M satisfies a clause C if at least one of its literals is true in M. It falsifies C if all the literals of C are false in M. Otherwise, C is undefined in M. Note that the empty clause is falsified by any M.

Let $\mathscr{A} = (A, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ be a semiring. A *weighted clause* is a pair (C, w) such that C is a clause and $w \in A$ with $w <_{\mathscr{A}} \mathbf{1}$ denotes its weight. A *semiring-induced propositional formula* is a set of weighted clauses $F = \{(C_1, w_1), \ldots, (C_e, w_e)\}$[2].

3.2 Semantics

Definition 3. *Consider a formula $F = \{(C_1, w_1), \ldots, (C_e, w_e)\}$. Each weighted clause (C_i, w_i) defines a function over total assignments,*

$$\phi_i(M) = \begin{cases} w_i & : \quad M \text{ falsifies } C_i \\ \mathbf{1} & : \quad \text{otherwise} \end{cases}$$

The formula F defines an evaluation function *as*

$$\phi_{\mathscr{A},F}(M) = \bigotimes_{i=1}^{e} \phi_i(M) \ ,$$

where M is a total assignment and ϕ_i is the function induced by (C_i, w_i). A total assignment M such that $\phi_{\mathscr{A},F}(M) > \mathbf{0}$ is called a model *of F.*

Definition 4. *The pair $(\mathscr{A}, F)$ defines the* marginalization problem *consisting in finding*

$$mrg(\mathscr{A}, F) = \bigoplus_{M \in \mathscr{M}} \phi_{\mathscr{A},F}(M)$$

As we will see, *mrg* is a very general computational problem. Note that, since $\forall_{a \in A}\ \mathbf{0} \oplus a = a$, only models of F contribute to *mrg*.

The following two definitions present two important relations among formulas. The effect of the α value on the definitions will be clear in Section 5. For the moment, it is just fine to ignore it or, what is equivalent, assume $\alpha = \mathbf{0}$.

Definition 5. *Let F and F' be two formulas over a common set of propositional symbols, and α a semiring value. We say that F $\mathscr{A}$-implies F' subject to threshold α, noted $F \models^{\mathscr{A}}_{\alpha} F'$, if for all total assignment M, $\alpha \oplus \phi_{\mathscr{A},F}(M) \leq \alpha \oplus \phi_{\mathscr{A},F'}(M)$.*

[2] For the sake of simplicity, we will restrict ourselves to clausal form formulas. The extension to the general case is direct: just let each C_i be an arbitrary boolean expression.

Definition 6. *When $F \models^{\mathscr{A}}_{\alpha} F'$ and $F' \models^{\mathscr{A}}_{\alpha} F$ we say that F and F' are $\mathscr{A}$-equivalent subject to threshold α and we note it $F \equiv^{\mathscr{A}}_{\alpha} F'$.*

If $F \models^{\mathscr{A}}_{\alpha} \{(\square, \mathbf{0})\}$ we say that F is an *α-contradiction*. In Section 5 we will take advantage of the following property.

Property 1. *If $F \models^{\mathscr{A}}_{\alpha} \{(\square, \mathbf{0})\}$ and M is a total assignment, then $\alpha \oplus \phi_{\mathscr{A},F}(M) = \alpha$.*

Zero-weighted clauses (i.e, of the form $(C, \mathbf{0})$) are called *hard* clauses. Note that, since $\forall_{a \in A}\ \mathbf{0} \otimes a = \mathbf{0}$, if M is a total assignment that falsifies a hard clause, then M is not a model. If we restrict ourselves to hard clauses and assume $\alpha = \mathbf{0}$, implication and equivalence correspond to the usual definitions in classical propositional logic.

For simplicity we will drop the semiring superscript when there is no ambiguity.

4 Applications

This section is devoted to illustrate the richness of semiring-induced propositional logic. It can be considered semi-tutorial because similar applications have been already identified in different contexts (e.g. [29, 27, 5, 19, 6]). Consider a formula $F = \{(C_1, w_1), \ldots, (C_e, w_e)\}$ defined over the symbols P where weights belong to a semiring $\mathscr{A}$.

4.1 Decision Problems ($\mathscr{A}_{\mathtt{bool}}$)

If we consider semiring $\mathscr{A}_{\mathtt{bool}}$, the weight of all clauses must be zero as $A = \{0, 1\}$ and, by definition, weights are smaller than 1. The corresponding evaluation function is $\phi_F(M) = \phi_1(M) \wedge \ldots \wedge \phi_e(M)$. It is easy to see that $\phi_F(M) = 1$ iff M satisfies every clause in F. Moreover, the marginalization problem $mrg(F) = \vee_{M \in \mathscr{M}}\ \phi_F(M)$ is 1 iff F is satisfiable. In other words, $mrg(\mathscr{A}_{\mathtt{bool}}, F)$ is equivalent to the boolean satisfiability problem (SAT) [24]. As logical consequence and logical equivalence can be reduced to SAT testing, the logic induced by semiring $\mathscr{A}_{\mathtt{bool}}$ can also be used to model such problems.

Example 2. *Consider a set of three boolean variables $P = \{x_1, x_2, x_3\}$, and the problem of assigning them in such a way that $x_1 = x_2$ and $x_2 \neq x_3$. If we want to know if the problem has any solution (i.e. it is satisfiable) we can use semiring $\mathscr{A}_{\mathtt{bool}}$. This problem is encoded in the formula $F = \{(x_1 \vee \neg x_2, 0), (\neg x_1 \vee x_2, 0), (x_2 \vee x_3, 0), (\neg x_2 \vee \neg x_3, 0)\}$. The first column in Figure 2 shows the set of total assignments $\mathscr{M}$. The second column shows, for each assignment M, the value $\phi_F(M)$ of the evaluation function. For instance $\phi_F(x_1, \neg x_2, x_3) = 0$ since this assignment does not satisfy clause $\neg x_1 \vee x_2$. The bottom row shows the result of the marginalization problem $mrg(F)$. In this case it is the logical OR of all $\phi_F(M)$ values. It is 1 as there are assignments for which the evaluation function is 1.*

4.2 Summation Problems ($\mathscr{A}_{\mathtt{count}}$)

If we consider semiring $\mathscr{A}_{\mathtt{count}}$, the corresponding evaluation function and the marginalization problem are $\phi_F(M) = \prod_{i=1}^{e} \phi_i(M)$ and $mrg(F) = \sum_{M \in \mathscr{M}} \phi_F(M)$, respectively. If the weight of all clauses is $w_i = 0 \in \mathbb{R}^+$, then $\phi_F(M) = 1$ iff M satisfies all clauses. So computing $mrg(F)$ is equivalent to the model counting problem, #SAT [4].

Example 3. *Consider the same problem as in the previous example. If we want to know the number of solutions we should use semiring $\mathscr{A}_{\mathtt{count}}$. The encoding of the problem is the same as before. The third column in Figure 2 shows the values of the evaluation function $\phi_F(M)$. As $\wedge$ and $\times$ are equivalent when restricted to $\{0, 1\}$, the evaluations do not change from the previous example. The last cell of the column shows the result of computing $mrg(F)$. It is 2 as there are two assignments whose evaluation is 1.*

Alternatively, if we allow different clauses to have different weights, *mrg* can model important problems such as the computation of marginals in Bayesian networks [26].

4.3 Optimization Problems ($\mathscr{A}_{\mathtt{min}+}$, $\mathscr{A}_{\mathtt{max}\times}$)

If we consider semiring $\mathscr{A}_{\mathtt{min}+}$, the corresponding evaluation function and marginalization problem are $\phi_F(M) = \sum_{i=1}^{e} \phi_i(M)$ and $mrg(F) = \min_{M \in \mathscr{M}}\{\phi_F(M)\}$, respectively. If the weight of all clauses is $w_i = 1 \in \mathbb{R}^+$, then $\phi_F(M) =$ "number of clauses falsified by M". Therefore computing $mrg(F)$ is equivalent to the problem of maximizing the number of satisfied clauses (Max-SAT) [24].

If we allow different clauses to have different weights, *mrg* is equivalent to the *partial weighted* Max-SAT problem [24, 20], which models a variety of interesting additive optimization problems with applications in bioinformatics, circuit design, electronic markets, resource allocation, etc. [16].

Example 4. *Consider the same problem as in the previous examples. Let $\mathscr{P}$ be the set of models of the formula F. Suppose we want to find the model with the least number of variables set to true, $\min_{M \in \mathscr{P}} \sum_{1 \leq i \leq 3} x_i$. We can express this problem with semiring $\mathscr{A}_{\mathtt{min}+}$. The clauses already existing in the previous example should be made hard and for each symbol x_i a unit clause $(\neg x_i, 1)$ should be added. The fourth column in Figure 2 shows the values of the resulting evaluation function $\phi_F(M)$. For instance, $\phi_F(\{x_1, x_2, \neg x_3\}) = 2$ as $\{x_1, x_2, \neg x_3\}$ falsifies clauses $(\neg x_1, 1)$ and $(\neg x_2, 1)$. The bottom cell of the column shows $mrg(F)$. In this case $mrg(F) = 1$, the minimum over all evaluations of total assignments.*

Semirings $\mathscr{A}_{\mathtt{max}\times}$ and $\mathscr{A}_{\mathtt{min}+}$ are isomorphic since we can transform the former into the latter via a logarithmic mapping [25]. Therefore, they have the same expressive power. Nevertheless, semiring $\mathscr{A}_{\mathtt{max}\times}$ seems to be a more natural choice for modeling probabilistic problems such as the most probable explanation problem (MPE) in Bayesian networks [26] or the MAP inference problem in Markov random fields [18] with applications in diagnosis, vision, signal encoding, etc.

4.4 Multi-criteria Optimization Problems ($\mathscr{A}^n$, $\mathscr{A}^f$)

Consider the multidimensional extension $\mathscr{A}^n$ of a semiring $\mathscr{A} = (A, \oplus, \otimes, \mathbf{0}, \mathbf{1})$. Weights are now n-dimension vectors, $w_i = (w_i^1, \ldots, w_i^n)$. It is easy to see that the resulting evaluation function satisfies $\phi_{\mathscr{A}^n,F}(M) = (\phi_{\mathscr{A},F^1}(M), \ldots, \phi_{\mathscr{A},F^n}(M))$, where $F^j = \{(C_1, w_1^j), \ldots, (C_e, w_e^j)\}$ is the projection of F onto the j-th vector dimension. In words, the evaluation function treats each dimension independently from the others. Further, the marginalization satisfies $mrg(\mathscr{A}^n, F) = (mrg(\mathscr{A}, F^1), \ldots, mrg(\mathscr{A}, F^n))$, which corresponds to the independent marginalization problem of each dimension. Therefore, one can use $\mathscr{A}^n$ to model (and, as we will see in Section 5, solve) in one shot n independent problems over the same set of symbols.

Example 5. *Consider again our running example, now with two objectives. The first one, as in the previous example, is to minimize the variables set to true. The second one is to minimize the number of variable pairs simultaneously set to false. Formally,*

$$\min_{M \in \mathscr{P}} \Big(\sum_{1 \leq i \leq 3} x_i, \sum_{1 \leq i < j \leq 3} (1 - x_i)(1 - x_j) \Big) .$$

We can model it using semiring $\mathscr{A}^2_{\mathtt{min}+}$. Note that its zero and unit elements are (∞, ∞) and $(0, 0)$, respectively. Hard clauses must have the new zero element,

$$\{(x_1 \vee \neg x_2, (\infty, \infty)), (\neg x_1 \vee x_2, (\infty, \infty)), (x_2 \vee x_3, (\infty, \infty)), (\neg x_2 \vee \neg x_3, (\infty, \infty))\} .$$

M	$\mathscr{A}_{\texttt{bool}}$	$\mathscr{A}_{\texttt{count}}$	$\mathscr{A}_{\texttt{min}+}$	$\mathscr{A}^2_{\texttt{min}+}$	$(\mathscr{A}^2_{\texttt{min}+})^f$
$\{x_1,x_2,x_3\}$	0	0	∞	(∞,∞)	$\{(\infty,\infty)\}$
$\{x_1,x_2,\neg x_3\}$	1	1	2	$(2,0)$	$\{(2,0)\}$
$\{x_1,\neg x_2,x_3\}$	0	0	∞	(∞,∞)	$\{(\infty,\infty)\}$
$\{x_1,\neg x_2,\neg x_3\}$	0	0	∞	(∞,∞)	$\{(\infty,\infty)\}$
$\{\neg x_1,x_2,x_3\}$	0	0	∞	(∞,∞)	$\{(\infty,\infty)\}$
$\{\neg x_1,x_2,\neg x_3\}$	0	0	∞	(∞,∞)	$\{(\infty,\infty)\}$
$\{\neg x_1,\neg x_2,x_3\}$	1	1	1	$(1,1)$	$\{(1,1)\}$
$\{\neg x_1,\neg x_2,\neg x_3\}$	0	0	∞	(∞,∞)	$\{(\infty,\infty)\}$
$mrg(\mathscr{A},F)$	1	2	1	$(1,0)$	$\{(2,0),(1,1)\}$

Figure 2: Evaluation functions and marginalization problems induced by different semirings. The set of symbols is $P = \{x_1,x_2,x_3\}$. The set of clauses changes from one case to another.

The first objective can be encoded with the following set of clauses,

$$\{(\neg x_1,(1,0)),(\neg x_2,(1,0)),(\neg x_3,(1,0))\}\ .$$

The second objective can be encoded with the following set of clauses,

$$\{(x_1 \vee x_2,(0,1)),(x_1 \vee x_3,(0,1)),(x_2 \vee x_3,(0,1))\}\ .$$

The fifth column in Figure 2 shows the values of the corresponding evaluation function. For instance, $\phi_F(\{\neg x_1,\neg x_2,x_3\}) = (1,1)$ *as it falsifies clauses* $(\neg x_3,(1,0))$ *and* $(x_1 \vee x_2,(0,1))$*, and* $(1,0) +^2 (0,1) = (1,1)$*. The last cell of the column shows the result of the marginalization problem. In this case it is the point-wise minimum over all the column entries.*

Given a formula $F = \{(C_1,w_1),\ldots,(C_e,w_e)\}$ its frontier extension is $F' = \{(C_1,w'_1),\ldots,(C_e,w'_e)\}$, where $w'_j = \{w_j\}$. It can be proved [6] that $\phi_{\mathscr{A}^f,F'}(M) = \{\phi_{\mathscr{A},F}(M)\}$ and

$$mrg(\mathscr{A}^f,F') = \{\phi_{\mathscr{A},F}(M) \mid \forall_{N\in\mathscr{M}}, \phi_{\mathscr{A},F}(M) \not\leq \phi_{\mathscr{A},F}(N)\},$$

which is the set of maximal evaluations of $\phi_{\mathscr{A},F}(M)$.

An immediate application is to model a multi-objective problem with n objectives with semiring $(\mathscr{A}^n)^f$. Each objective is encoded in one dimension of $\mathscr{A}^n$. The marginalization problem corresponds to the so-called *efficient frontier* of the problem, which is the most general notion of optimality in multi-objective optimization.

Example 6. *If we want to compute the efficient frontier of the previous bi-objective problem, we can use the frontier extension of the previous semiring,* $(\mathscr{A}^2_{\texttt{min}+})^f$*, and replace vector weights by singleton vector weights. The sixth column in Figure 2 shows the values of the corresponding evaluation function. The last cell of the column shows the result of the corresponding marginalization problem. In this case it is the union of all the values followed by the removal of the non optimal elements, which is the efficient frontier of the original bi-objective optimization problem.*

5 A DPLL Algorithm for Semiring-Induced Propositional Logic

In this section we show how DPLL for satisfiability testing and its most prominent enhancements can be naturally generalized to compute the marginalization problem of a semiring-induced propositional logic formula. Following [23], we describe the algorithm using a transition system.

5.1 Transition Systems

We will model our semiring-induced DPLL procedures by means of a set of *states* together with a binary relation $\Rightarrow$ over these states, called the *transition relation*. If $S \Rightarrow S'$ we say that there is a *transition* from S to S'. We call any sequence of transitions of the form $S_0 \Rightarrow S_1$, $S_1 \Rightarrow S_2$, ... a *derivation*, and denote it by $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots$ We call any subsequence of a derivation a *subderivation*.

In what follows, transition relations will be defined by means of conditional *transition rules*. For a given state S, a transition rule precisely defines whether there is a transition from S by this rule and, if so, to which state S'. Such a transition is called an *application step* of the rule.

A *transition system* is a set of transition rules defined over some given set of states. Note that if more than one transition is possible from S, any option is valid. If there is no transition from S, we will say that S is *final*.

5.2 States in SDPLL Transition Systems

Semiring-induced DPLL (SDPLL) can be fully described by simply considering that a state of the procedure is of the form (α, M, F), where F is a formula, M is a (partial) assignment and $\alpha \in A$ is a semiring element. Additionally, we define states of the form $(\alpha, \mathit{done}, F)$ to represent final states.

More precisely, M is a *sequence* of literals, never containing both a literal and its negation. Each literal has an *annotation*, a bit that marks it as a *decision* literal or not. Essentially, a decision literal is a literal that is added in the context of a split case and, at some point, its negation needs to be considered. The concatenation of two such sequences will be denoted by simple juxtaposition. When we want to emphasize that a literal l is a decision literal we will write it as l^{d}. We will denote the empty sequence of literals (or the empty assignment) by $\emptyset$.

Adding a literal l to M means that we are conditioning the formula F with l. This is equivalent to adding a unit hard clause $(l, \mathbf{0})$ to F. Accordingly, we will frequently consider M as a set of unit hard clauses, ignoring the annotations, the order between its elements and assuming an implicit zero weight. Therefore, $M \cup F$ with $M = l_1 \ldots l_n$ will be a shorthand for $\{(l_1, \mathbf{0}), \ldots (l_n, \mathbf{0})\} \cup F$.

5.3 The Basic SDPLL Procedure

A basic backtracking-based algorithm that enumerates all total assignments can be defined with the following three rules. Since none of them changes the input formula F, we do not include it in the state descriptions.

Definition 7. *Backtracking-based enumeration rules.*

$$\textit{Decide}: \quad (\alpha, M) \;\Rightarrow\; (\alpha, M l^{\mathsf{d}}) \quad \textbf{if} \quad \{\ l \text{ is undefined in } M$$

$$\textit{BacktrackSuccess}: \quad (\alpha, M l^{\mathsf{d}} N) \;\Rightarrow\; (\alpha', M \neg l) \quad \textbf{if} \quad \begin{cases} \alpha' = \alpha \oplus \phi_F(M\ l^{\mathsf{d}}\ N) \\ M\ l^{\mathsf{d}}\ N \text{ is total} \\ N \text{ has no decision literals} \end{cases}$$

$$\textit{DoneSuccess}: \quad (\alpha, M) \;\Rightarrow\; (\alpha', \mathit{done}) \quad \textbf{if} \quad \begin{cases} \alpha' = \alpha \oplus \phi_F(M) \\ M \quad \text{is total} \\ M \text{ has no decision literals} \end{cases}$$

One can use the previous system for solving a semiring-induced marginalization problem $\mathit{mrg}(F)$ by simply generating an arbitrary derivation $(\mathbf{0}, \emptyset) \Rightarrow \ldots \Rightarrow (\alpha, \mathit{done})$. The α value of the final state is the

solution of the marginalization problem. The algorithm generates all total assignments M and adds their $\phi_F(M)$ contribution to the semiring value.

Consider an arbitrary state (α, M). If M is a partial assignment, rule Decide models the split case. The assignment M is extended with a so far undefined literal l. The literal is annotated as a decision literal to denote that once all the extensions of Ml have been taken into account, the extensions of $M\neg l$ must still be considered. If M is a total assignment, then the contribution of $\phi_F(M)$ must be added to α (i.e, $\alpha' = \alpha \oplus \phi_F(M)$). If M does not contain any decision literals, it means that all total assignments have already been considered, so the value of α is the final result and we can end the application of rules. This is the situation considered by the DoneSuccess rule. Alternatively, if M contains decision literals the algorithm backtracks by replacing the most recent decision literal by its negation and removing all subsequent literals in M. This is the situation considered by the BacktrackSuccess rule. Clearly, the algorithm terminates in a finite number of steps and, in a final state, α contains the result of the marginalization problem. Note that the α value increases monotonically during the execution.

Example 7. *Consider an arbitrary intermediate state (α, M). If the semiring is $\mathscr{A}_{bool}$, then $\alpha = 1$ iff "some total assignment in a previous state was a model". If the semiring is $\mathscr{A}_{count}$, then $\alpha =$"number of models found so far". If the semiring is $\mathscr{A}_{min+}$, then $\alpha =$"evaluation of the best (i.e, minimum cost) model so far". Finally, if the semiring is $\mathscr{A}^{nf}_{min+}$, then $\alpha =$"set of evaluations of pareto-optimal models with respect to already inspected total assignments".*

Obviously, the previous algorithm is extremely inefficient. It can be improved with the addition of *pruning*. We say that state (α, M) is *in a conflict* if $M \cup F$ is an α-contradiction (i.e, $M \cup F \models_\alpha \{(\square, \mathbf{0})\}$). Property 1 indicates that the algorithm can discard (i.e. prune) all the extensions of the assignment M since they will not contribute to the solution. Pruning is specified with the following rules.

Definition 8. *The basic SDPLL algorithm is the system defined by the previous three and the following three pruning rules.*

$$\textit{Propagate}: \quad (\alpha, M) \Rightarrow (\alpha, Ml) \quad \textbf{if} \quad \begin{cases} (M\neg l) \cup F \models_\alpha \{(\square, \mathbf{0})\} \\ l \text{ is undefined in } M \end{cases}$$

$$\textit{BacktrackFail}: \quad (\alpha, Ml^{\mathrm{d}}N) \Rightarrow (\alpha, M\neg l) \quad \textbf{if} \quad \begin{cases} (Ml^{\mathrm{d}}N) \cup F \models_\alpha \{(\square, \mathbf{0})\} \\ N \text{ has no decision literals} \end{cases}$$

$$\textit{DoneFail}: \quad (\alpha, M) \Rightarrow (\alpha, done) \quad \textbf{if} \quad \begin{cases} M \cup F \models_\alpha \{(\square, \mathbf{0})\} \\ M \text{ has no decision literals} \end{cases}$$

Rule DoneFail considers the case in which the current assignment M is in a conflict and does not contain any decision literal. In that case we can end the execution. Rule BacktrackFail considers the case in which M is in a conflict and contains decision literals. In that case the algorithm backtracks by replacing the most recent decision literal by its negation and removing all subsequent literals of M. Rule Propagate considers the case when M is not in a conflict, but $M\neg l$ is. In that case, the algorithm extends M with l. Note that l is not marked as a decision, as its negation needs not to be considered.

Unlike the three earlier rules, the applicability of the pruning rules is not easy to check since detecting conflicts is in general NP-hard. Therefore, practical algorithms rely on sufficient conditions that can be efficiently computed. The following property presents a very naive, but still widely used one.

Property 2. *Consider a transition state (α, M). Let $V \subseteq F$ be the set of clauses falsified by M, and $\beta = \bigotimes_{(C,w)\in V} w$ the product of costs of violated clauses. Then, $\alpha \oplus \beta = \alpha$ is a sufficient condition for $M \cup F \models_\alpha \{(\square, \mathbf{0})\}$.*

Example 8. *Consider the simplest semiring $\mathscr{A}_{bool}$. The previous pruning condition becomes $\alpha = 1 \vee V \neq \varnothing$. It occurs when either a model has already been found or the current assignment M falsifies some clause. With semiring $\mathscr{A}_{\min+}$, the previous pruning condition becomes $\min\{\alpha,\beta\} = \alpha$. It will occur if either $\alpha = 0$ (a solution that cannot be improved has already been found) or $\alpha \leq \beta$ (the current assignment M is already worse than the best solution found so far). Consider the more complex semiring $(\mathscr{A}^2_{\min+})^f$ (i.e, the frontier extension of a two-dimensional optimization semiring). In this case, α is the set of optimal evaluations found so far and $\beta = \{(w_1,w_2)\}$ is a singleton that adds up the violations of M. The pruning condition is $||\alpha \cup \{(w_1,w_2)\}|| = \alpha$, which occurs when (w_1,w_2) is dominated by some element of α.*

Observe that when Propagate can be applied, so is Decide. For efficiency reasons, it is desirable to always chose Propagate.

The SDPLL algorithm using the previous pruning condition is a faithful generalization of several algorithms: If the semiring is $\mathscr{A}_{bool}$ it becomes DPLL [10] for satisfiability testing. If the semiring is $\mathscr{A}_{count}$ it becomes the algorithm in [4] for model counting. If the semiring is $\mathscr{A}_{min+}$ it becomes the algorithm in [7] for Max-SAT. Finally, if the semiring is $(\mathscr{A}^n_{min+})^f$ it becomes essentially equivalent to the algorithm described in [14] for multi-objective optimization.

6 Extending SAT Techniques

In the previous section we showed how our formalism allows one to unify several basic enumeration algorithms by abstracting away algorithmic details. Here we show its convenience for generalizing more sophisticated techniques. We consider three features common to all modern DPLL-based SAT solvers: backjumping, learning and restarts.

6.1 Backjumping, Learning and Restarts

The purpose of *backjumping* is to undo *several* decisions at once, going back to a lower decision level than the previous level and adding some new literal to that lower level. The Backjump rule below models this idea.

$$\textsf{Backjump}: \quad (\alpha, M l^{\mathsf{d}} N) \Rightarrow (\alpha', M l') \textbf{ if } \begin{cases} \text{there is a previous state} \\ (\alpha', M) \text{ such that:} \\ (M\neg l') \cup F \models_{\alpha'} \{(\square, \mathbf{0})\} \\ l' \text{ is undefined in } M \end{cases}$$

It can be seen as a delayed propagation. In words, the rule is triggered if the algorithm detects that a propagation instead of a decision could have been done at an earlier state (α', M). This occurs when the condition for propagation, i.e. $(M\neg l') \cup F \models_{\alpha'} \{(\square, \mathbf{0})\}$, was not detected at the earlier state (recall that simple sufficient conditions are usually used) but can be detected now (possibly, from an analysis of the current state). Note that $\alpha' \leq \alpha$ as α' is taken from a previous state. Therefore with the Backjump rule the semiring value does not grow monotonically anymore during the execution of the algorithm.

The purpose of *learning* is to make explicit in the original formula implicit clauses, because they may help in the future identification of conflicts. Similarly, when a clause seems not to be useful for that purpose according to some measure, it can be removed. The Learn and Forget rules below generalize this idea. Since these rules modify the formula F, we add it to the state description.

$$\textsf{Learn}: \quad (\alpha, M, F) \Rightarrow (\alpha, M, F \cup \{(C,w)\}) \textbf{ if } \{ F \models_{\mathbf{0}} F \cup \{(C,w)\}$$

$$\textsf{Forget}: \quad (\alpha, M, F \cup \{(C,w)\}) \Rightarrow (\alpha, M, F) \textbf{ if } \{ F \models_{\mathbf{0}} F \cup \{(C,w)\}$$

Observe that the Learn and Forget rules allow one to add and remove from the current formula F an arbitrary clause C as long as it is $\mathbf{0}$-entailed by F. The addition and removal is safe, even in combination with Backjump and Restart (to be seen later) which decrease the α value, precisely because the entailment is required with respect to the lowest possible α value.

Finally, it may be useful to *restart* the DPLL procedure whenever the search is not making enough progress according to some measure. The rationale behind this idea is that upon each restart, the additional knowledge of the search space compiled into the newly learned clauses will lead the heuristics for Decide to behave differently, and possibly in a wiser way. The following rule models this idea.

$$\textsf{Restart}: \qquad (\alpha, M, F) \;\Rightarrow\; (\mathbf{0}, \emptyset, F)$$

If Learn and Forget are applied, termination of the procedure can be achieved by avoiding infinite subderivations with only Learn and Forget steps. On the other hand, if the Restart rule is also applied, in order to get termination in practice one periodically increases the minimum number of applications of the other rules between each pair of restart steps. This is formalized below.

Definition 9. *Let us consider a derivation by the basic SDPLL rules together with the Backjump, Learn, Forget and Restart rules. We say that Restart* has increasing periodicity *in the derivation if, for each subderivation $S_i \Rightarrow \ldots \Rightarrow S_j \Rightarrow \ldots \Rightarrow S_k$ where the steps producing S_i, S_j, and S_k are the only Restart steps, the number of steps of the other rules in $S_i \Rightarrow \ldots \Rightarrow S_j$ is strictly smaller than in $S_j \Rightarrow \ldots \Rightarrow S_k$.*

Finally, the following theorem shows how the SDPLL algorithm can be used to effectively compute the marginalization of a given formula.

Theorem 1. *Let us consider the basic SDPLL rules together with the Backjump, Learn, Forget and Restart rules. If infinite subderivations consisting of only Learn and Forget steps are not allowed and Restart has increasing periodicity, any derivation $(\mathbf{0}, \emptyset, F) \Rightarrow \ldots \Rightarrow S$ is finite. Moreover, if S is final then it is of the form $(mrg(F), done, G)$.*

6.2 Conflict-Driven Backjumping and Learning

The previous four rules have been presented in their most general form. In principle, they can be applied independently. Still, the experience from modern SAT solvers is that it is their combination what produces the best results. Besides, their application should be *driven by conflicting states*. In our framework the description of this idea requires the specialization of the Backjump rule as follows.

ConflictDrivenBackjump :

$$(\alpha, Ml^{\mathsf{d}}N) \;\Rightarrow\; (\alpha', Ml') \qquad \textbf{if} \begin{cases} Ml^{\mathsf{d}}N \cup F \models_{\alpha} \{(\Box, \mathbf{0})\} \\ \text{there exists a previous state} (\alpha', M) \text{ and} \\ \text{some clause } l_1 \vee \cdots \vee l_n \vee l' \text{ such that:} \\ \quad F \models_{\alpha'} \{(l_1 \vee \cdots \vee l_n \vee l', \mathbf{0})\} \\ \quad \forall_{1 \le i \le n},\ M \cup F \models_{\alpha'} \{(\neg l_i, \mathbf{0})\} \\ \quad l' \text{ is undefined in } M \end{cases}$$

We call the clause $l_1 \vee \ldots \vee l_n \vee l'$ in ConflictDrivenBackjump a *backjump* clause. This rule is more specific than the previous Backjump because it can be only applied when the current state is a conflict. Furthermore, the analysis of the conflict has to reveal the existence of a backjump clause. It can be easily proved that the two conditions for a backjump clause imply the condition $(M\neg l') \cup F \models_{\alpha'} \{(\Box, \mathbf{0})\}$ of the more general Backjump rule. Conflict-driven-learning restricts the Learn and Forget rules to add and remove only backjump clauses, which in turn restricts the new knowledge after each restart.

Conflict analysis [31] is the efficient detection of useful backjump clauses. It is only well-studied in the SAT case. However, the following example shows that our abstract description provides direct generalizations to other problems.

Example 9. *Consider semiring $\mathscr{A}_{min+}$ and a formula with, among others, the following clauses, $\{(\neg x_1 \vee x_2, \infty), (\neg x_2 \vee x_3, 8), (\neg x_4 \vee x_5, 7), (\neg x_7 \vee x_8, \infty), (\neg x_7 \vee \neg x_8, 7), (\neg x_2, 1), (\neg x_3, 1), (\neg x_5, 1)\}$. Suppose that an execution of SDPLL that uses the pruning condition of Prop. 2 has generated state $(9, M)$ such that M does not violate any clause and $x_1, \ldots, x_8$ are undefined in M. A possible subderivation is*

$$\ldots \Rightarrow (9,M) \Rightarrow (9,Mx_1^d) \Rightarrow (9,Mx_1^d x_2) \Rightarrow (9,Mx_1^d x_2 x_3) \Rightarrow (9,Mx_1^d x_2 x_3 x_4^d) \Rightarrow$$

$$\Rightarrow (9,Mx_1^d x_2 x_3 x_4^d x_5) \Rightarrow (9,Mx_1^d x_2 x_3 x_4^d x_5 x_6^d) \Rightarrow\Rightarrow (9,Mx_1^d x_2 x_3 x_4^d x_5 x_6^d x_7^d) \Rightarrow$$

$$\Rightarrow (9,Mx_1^d x_2 x_3 x_4^d x_5 x_6^d x_7^d x_8)$$

The current state is in a conflict since it satisfies the pruning condition (the sum of weights of clauses falsified by the current assignment is 10 *and* $\min\{9,10\} = 9$*). If we analyze the conflict, we observe that decisions x_4^d and x_6^d are irrelevant for the pruning condition (if we remove their contribution and the contribution of their implications, the state is still in a conflict). From the analysis of the conflict, we can obtain backjump clause $\neg x_2 \vee \neg x_7$. The application of the rule* ConflictDrivenBackjump *produces the following state $(9, Mx_1^d x_2 x_3 \neg x_7)$.*

It is important to note that state-of-the-art Max-SAT solvers [16, 21] are very naive in terms of backjumping (they only backjump when the conflict is exclusively caused by hard clauses). Thus, our backjump rule does not only cover this basic case, but also opens a new perspective for new cases as the one in this example.

6.3 Semirings with Idempotent $\oplus$

Semirings with idempotent $\oplus$ include all applications discussed in Section 4, except for counting problems ($\mathscr{A}_{count}$). Although the purpose of this paper is to consider general definitions and techniques, the $\oplus$-idempotent case is still so general that we will mention some algorithmic improvements for it. They are all based on the fact that during the execution of SDPLL, it is always possible to replace the semiring value of a previous state by the higher semiring value of the current state and resume the execution from that earlier state. Formally,

Property 3. *Let F be a formula defined over a $\oplus$-idempotent semiring. Consider an arbitrary derivation of the basic SDPLL (without the enhancements of Section 6.1), $(\mathbf{0}, \varnothing) \Rightarrow (\alpha_1, M_1) \Rightarrow \ldots \Rightarrow (\alpha_j, M_j)$. For any $1 \leq i \leq j$, any derivation from the state (α_j, M_i) to a final state $(\alpha_n, done)$ satisfies that $\alpha_n = mrg(F)$.*

The first implication of this is that in the Backjump (and ConflictDrivenBackjump) rule the occurrences of α' can be replaced by α. Therefore, the rule models a return to an earlier state but preserving the better semiring value of the current state. This allows one to take advantage of the work done so far for, e.g., propagating unassigned literals or detecting new conflicts after backjumping. An interesting feature of the resulting rule is that ConflictDrivenBackjump subsumes chronological backtracking, as the negation of all decision literals of the current assignment M is a backjump clause.

The second implication of Property 3 is that the Restart rule can restart the search preserving the semiring value of the current state. Finally, the Learn and Forget rules can add and remove α-implied clauses where α is not $\mathbf{0}$ but the semiring value of the current state, which broadens the range of clauses that can be used for learning.

7 Conclusions and Future Work

In this paper we introduce semiring-induced propositional logic, which extends propositional logic by allowing clauses to be weighted with semiring values. We show that it provides a convenient formalism for modeling a variety of important computational problems. Further, it serves as an elegant and well-defined presentation of general solving techniques by focusing on the general idea and abstracting away algorithmic details.

In our future work we want to incorporate to the SDPLL algorithm decomposition techniques, which have proven fundamental in counting problems [17, 2].

This paper has focused on enumeration-based algorithmic techniques. We want to investigate which inference-based algorithms can also be unified. In particular, we want to study under which conditions the resolution rule for Max-SAT introduced in [20] can be generalized to arbitrary semirings, while preserving completeness.

References

[1] S. Aji and R. McEliece. The generalized distributive law. *IEEE Trans. on Information Theory*, 46(2):325–343, 2000.

[2] F. Bacchus, S. Dalmao, and T. Pitassi. Algorithms and complexity results for #SAT and Bayesian inference. In *FOCS*, pp. 340–351, 2003.

[3] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.

[4] E. Birnbaum and E. Lozinskii. The good old Davis-Putnam procedure helps counting models. *JAIR*, 10:457–477, 1999.

[5] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *JACM*, 44(2):201–236, March 1997.

[6] S. Bistarelli, M. Gadducci, J. Larrosa, and E. Rollon. A semiring-based approach to multi-objective optimization. In *Proc. of Intl. Workshop on Soft Constraints and Preferences*, 2008.

[7] B. Borchers and J. Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *Journal of Combinatorial Optimization*, 2:299–306, 1999.

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

[9] Y. Crama, P. Hansen, and B. Jaumard. The basic algorithm for pseudo-boolean programming revisited. *Discrete Applied Mathmatics*, 29:171–185, 1990.

[10] M. Davis, G. Logemann, and G. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[11] M. Davis and H. Putnam. A computing procedure for quantification theory. *JACM*, 3(1960).

[12] R. Dechter and J. Pearl. Network-based heuristics for constraint-satisfaction problems. *Artif. Intell.*, 34:1–38, 1988.

[13] Z. Fu and S. Malik. On solving the partial Max-SAT problem. In *SAT*, pp. 252–265, 2006.

[14] M. Gavanelli. An algorithm for multi-criteria optimization in CSPs. *ECAI*, pp. 136–140, 2002.

[15] J. Golan. *Semirings and their applications*. Kluwer Academic Publishers, 1999.

[16] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSAT: An efficient Weighted Max-SAT Solver. *JAIR*, 31:1–32, 2008.

[17] R. Bayardo and J. Pehoushek. Counting models using connected components. In *AAAI/IAAI*, pp. 157–162, 2000.

[18] R. Kindermann and L. Snell. *Markov Random Fields and Their Applications*. AMS, 1980.

[19] J. and N. Wilson. Semiring induced valuation algebras: Exact and approximate local computation algorithms. *Artif. Intell.*, 172(11):1360–1399, 2008.

[20] J. Larrosa, F. Heras, and S. de Givry. A logical approach to efficient Max-SAT solving. *Artif. Intell.*, 172(2-3):204–233, 2008.

[21] C. Li, F. Manyà, and J. Planes. New inference rules for Max-SAT. *JAIR*, 30:321–359, 2007.

[22] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *DAC'01*, pp. 530–535. ACM Press, 2001.

[23] R. Nieuwenhuis, A. Oliveras, C. Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL. *JACM*, 53(6):937–977, 2006.

[24] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, USA, 1994.

[25] J. Park. Using weighted Max-SAT engines to solve MPE. In *AAAI-02*, pp. 682–687.

[26] J. Pearl. *Probabilistic Inference in Intelligent Systems. Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[27] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems. In *IJCAI-95*, pp. 631–637.

[28] G. R. Shafer and P.P. Shenoy. Probability propagation. *Anals of Mathematics and Artificial Intelligence*, 2:327–352, 1990.

[29] P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. In *UAI-88*.

[30] Z. Xing and W. Zhang. Maxsolver: An efficient exact algorithm for (weighted) maximum satisfiability. *Artif. Intell.*, 164(1-2):47–80, 2005.

[31] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik. Efficient Conflict Driven Learning in a Boolean Satisfiability Solver. In *ICCAD'01*, pp. 279–285, 2001.