Solving Many-Valued SAT Encodings with Local Search

Carlos Ansótegui
Dept. of Computer Science
Universitat de Lleida
E-25001 Lleida, Spain

Felip Manyà
Dept. of Computer Science
Universitat de Lleida
E-25001 Lleida, Spain

Ramón Béjar
Dept. of Computer Science
Universitat de Lleida
E-25001 Lleida, Spain

Carla P. Gomes
Dept. of Computer Science
Cornell University
Ithaca, NY 14853, USA

Abstract

In this paper we present MV-SAT, which is a many-valued constraint programming language that bridges the gap between Boolean Satisfiability and Constraint Satisfaction. Our overall goal is to extend the SAT formalism with many-valued sets and deal with more compact and natural encodings, as in CSP approaches, while retaining the efficiencies of SAT solvers operating on uniform encodings. After some formal definitions, we first discuss the logical and complexity advantages of MV-SAT compared to SAT and other many-valued problem modeling languages. Second, we define MV-SAT encodings, and analyze their complexity, for a number of combinatorial problems: quasigroup with holes completion, graph coloring, all interval series, and sports scheduling. Third, we describe MV-WalkSAT: a local search strategy adapted from the Boolean WalkSAT procedure that we have implemented and that incorporates several heuristics to escape from local minima. Finally, we report on an empirical evaluation that provides experimental evidence of the competitiveness of the MV-SAT problem solving approach.

Introduction

In recent years we have seen an increasing interest in propositional satisfiability encodings. The study of search behavior of random propositional satisfiability (SAT) formulas has provided tremendous insights into the hardness nature of such combinatorial problems, beyond the worst-case notion of NP-completeness. In particular, such studies have uncovered an interesting phase transition behavior between an area in which most instances are solvable and one in which most of the instances are unsolvable (Cheeseman, Kanefsky, & Taylor 1991; Mitchell, Selman, & Levesque 1992) — the critically constrained area, where the hardest instances occur, coincides with the phase transition.

The identification of very hard instances has in turn led to the development of fast SAT solvers, which in turn is making propositional satisfiability a competitive encoding to solve other NP-complete problems. The approach consists of translating a given problem into propositional satisfiability, solving it with a fast SAT solver and mapping the solution back into the original problem. Examples of domains where propositional encodings have been shown effective include hardware verification (Marques-Silva & Guerra 1999; Moskewicz *et al.* 2001), planning (Kautz & Selman 1996), and other benchmark problems such as graph coloring (Selman & Kautz 1993), and the quasigroup completion problem (Gomes & Selman 1997).

The success of using CSP and SAT solvers to tackle hard combinatorial problems has been somewhat counter-intuitive. Traditional wisdom is that designing a search method working directly on the original problem encoding should outperform approaches that require a translation via a generic intermediate format, such as a SAT or a CSP encoding. However, this line of reasoning ignores the fact that generic solvers can benefit from many years of development by a broad research community. It is not easy to duplicate this kind of effort for a particular problem domain. Moreover, in generic problem solving approaches, encodings can compensate for much of the loss due to going to a uniform representation formalism like CSP or SAT. It is well-known that different encodings (e.g. dual encodings, redundant encodings, encodings that break symmetries) of a same combinatorial problem may give raise to very different performance profiles.

Let's consider a concrete example, the problem of finding a completion for a partially defined quasigroup or Latin square. This problem consists of coloring the n^2 cells of an $n \cdot n$ matrix, with n colors, such that there are no repetitions of color in each row and each column. The most natural way of encoding this problem is a CSP formulation with n^2 variables, each variable with a domain of n colors, and O(n) "all-diff" constraints (each constraint stating that n variables, in a row or column, need to be assigned different values). An efficient SAT encoding uses n^3 variables and $O(n^4)$ clauses (Achlioptas *et al.* 2000; Kautz *et al.* 2001). In this domain, sophisticated CSP methods have been explored in detail (e.g., using global consistency measures and randomization) (Gomes & Selman 1997; Kautz *et al.* 2001; Régin 1994; Stergiou & Walsh 1999). Such approaches currently can handle hard completion problems involving Latin squares. Nevertheless, despite the tremendous progress on the CSP side, SAT solvers running on the relatively large SAT encodings defined in (Kautz *et al.* 2001) still often outperform the CSP approaches. As we will show, using the constraint programming language we propose in this paper, we obtain compact and natural encodings of Latin squares like in CSP, and we reduce dramatically the time that SAT solvers need to find a solution.

Our research program is aimed at bridging the gap between propositional satisfiability encodings and constraint satisfaction formalisms. The challenge is to combine the inherent efficiencies of SAT solvers operating on uniform satisfiability encodings with the much more compact and natural representations, and more sophisticated propagation techniques of CSP formalisms. Our starting point to achieving our research objectives is our experience with many-valued propositional logics and, specifically, with developing encodings of combinatorial problems and satisfiability solvers for many-valued clausal forms (see, e.g. (Beckert, Hähnle, & Manyà 1999; 2000; Béjar 2000; Béjar *et al.* 2001; Béjar, Hähnle, & Manyà 2001; Béjar & Manyà 1999a; 1999b; 1999c)). Based on this experience, in this paper we propose MV-SAT as a new constraint programming language between CSP and SAT.

MV-SAT is the problem of deciding the satisfiability of many-valued CNF Formulas. A *many-valued CNF formula* is a classical propositional conjunctive clause form based on a generalized notion of literal, called many-valued literal. Given a truth value set T ($|T| \geq 2$) equipped with a total ordering $\leq$, a *many-valued literal* is an expression of the form $S : p$, where p is a propositional variable and S is a subset of T which is of the form $\{i\}$, of the form $\uparrow i = \{j \in T \,|\, j \geq i\}$, or of the form $\downarrow i = \{j \in T \,|\, j \leq i\}$ for some $i \in T$. The informal meaning of $S : p$ is "p is constrained to the values in S".

In this paper we first formally define the syntax and semantics of many-valued CNF formulas, as well as the many-valued clausal forms so-called signed, regular and monosigned CNF formulas. Second, we discuss the logical and complexity advantages of MV-SAT compared to SAT and other many-valued problem modeling languages. Third, we present MV-SAT encodings for a number of combinatorial problems: quasigroup with holes completion (Achlioptas *et al.* 2000), graph coloring (Selman & Kautz 1993), all interval series (Hoos 1998), and sports scheduling (Gomes *et al.* 1998). We show that some of these problems have a much concise and natural encoding in our many-valued formalism than in the Boolean formalism. For example, while an efficient SAT encoding of a quasigroup uses n^3 variables and $O(n^4)$ clauses (Achlioptas *et al.* 2000; Kautz *et al.* 2001), an MV-SAT encoding only uses n^2 variables and $O(n^2)$ clauses. Fourth, we present MV-WalkSAT: a local search strategy adapted from the Boolean WalkSAT procedure that incorporates several heuristics to escape from local minima (Basic, Novelty, R-Novelty, R-Novelty+, G); this algorithms builds on our previous implementations of Regular-WalkSAT (Béjar 2000; Béjar *et al.* 2001; Béjar & Manyà 1999c), which is a local search procedure for the subclass of regular CNF formulas. Nevertheless, and due to the improvements performed in the current implementation, MV-WalkSAT is slightly faster than Regular-WalkSAT even when MV-WalkSAT only considers regular CNF formulas. Fifth, we present detailed experiments on a representative sample of computationally difficult benchmarks. Our results provide experimental evidence of the competitiveness of the MV-SAT problem solving approach. Finally, we give some conclusions and concluding remarks.

Many-Valued CNF Formulas

Definition 1 *A* truth value set, *or* domain, *is a non-empty set $T = \{i_1, i_2, \ldots, i_n\}$, equipped with a total ordering $\leq$. A* sign *is a set $S \subseteq T$ of truth values. For each element i of the truth value set T, let $\uparrow i$ denote the sign $\{j \in T \mid j \geq i\}$, and let $\downarrow i$ denote the sign $\{j \in T \mid j \leq i\}$. A sign S is* monosigned *if it is identical to $\{i\}$ for some $i \in T$. A sign S is* regular *if it is identical either to $\uparrow i$ (positive) or to $\downarrow i$ (negative) for some $i \in T$. A sign S is* many-valued *if it is monosigned or regular.*

Definition 2 *A* signed literal *is an expression of the form $S : p$, where S is an arbitrary subset of T and p is a propositional variable. A* many-valued literal *is an expression of the form $S : p$, where S is a many-valued sign and p is a propositional variable; in case S is monosigned (regular) we say that $S : p$ is monosigned (regular). A* signed clause *is a disjunction of signed literals. A* many-valued clause *is a disjunction of many-valued literals; in case all the literals are monosigned (regular) we say that the clause is monosigned (regular), and in case there is either at most one regular positive literal or at most one monosigned literal we say that the clause is Horn. A* signed CNF formula *is a conjunction of signed clauses. A* many-valued CNF formula *is a conjunction of many-valued clauses; in case all the clauses are monosigned (regular, Horn) we say that the CNF formula is monosigned (regular, Horn).*

Definition 3 *An* interpretation *is a mapping that assigns to every propositional variable an element of the truth value set. An interpretation I satisfies a many-valued (signed) literal $S : p$ iff $I(p) \in S$. An interpretation satisfies a many-valued (signed) clause iff it satisfies at least one of its literals. A many-valued (signed) CNF formula Γ is* satisfiable *iff there exists at least one interpretation that satisfies all the clauses in Γ. A many-valued (signed) CNF formula that is not satisfiable is* unsatisfiable.

MV-SAT is formally defined as the problem of deciding the satisfiability of many-valued CNF formulas. Indeed, many-valued CNF formulas can be seen as the union of regular and monosigned CNF formulas, which are the most widely investigated many-valued problem modeling languages (see, e.g. (Baaz & Fermüller 1995; Béjar 2000; Béjar *et al.* 2001; Béjar, Hähnle, & Manyà 2001; Béjar & Manyà 1999a; 1999b; 1999c; Frisch & Peugniez 2001))

Logical and Complexity Aspects of MV-SAT

In this section we present the logical and complexity results that led us to choose many-valued CNF formulas as our problem modeling language. In particular, we show that the Horn-SAT and 2-SAT problems of many-valued CNF formulas belong to the same complexity class as the Horn-SAT and 2-SAT problems of the less powerful languages of monosigned and regular CNF formulas; observe that not all binary (Horn) many-valued CNF formula can be expressed either as a pure monosigned or as a pure regular binary (Horn) CNF formula.[1] On the other hand, Horn-SAT is NP-complete for monosigned CNF formulas with negation (i.e. formed by monosigned literals and negated monosigned literals), and 2-SAT is NP-complete for both signed and monosigned with negation CNF formulas. Interestingly, any of such CNF formulas is logically equivalent to a many-valued CNF formula that has exactly the same variables and the same number of clauses.

We have recently shown in (Ansótegui *et al.* 2002) that, given a monosigned or a regular positive unit clause $S : p$ and a many-valued Horn clause $S' : p \vee D$, the following inference rule (*many-valued positive unit resolution rule*)

$$\frac{S : p \quad S' : p \vee D}{D} \quad \text{if } S \cap S' = \emptyset \tag{1}$$

is refutation complete for many-valued Horn CNF formulas. Since the number of possible resolvents that can be produced by applying rule (1) to a many-valued Horn CNF formula is polynomial, we have that many-valued Horn-SAT is polynomially solvable. But we can prove that Horn-SAT is NP-complete for monosigned with negation CNF formulas: NP-containment is straightforward to show. The 3-colorability for an undirected graph $G = (V, E)$ is polynomially reducible to our Horn-SAT problem by taking $T = \{1, 2, 3\}$ and defining the following clauses

$$\begin{array}{l} (\neg\{1\} : u \vee \neg\{1\} : v) \wedge \\ (\neg\{2\} : u \vee \neg\{2\} : v) \wedge \\ (\neg\{3\} : u \vee \neg\{3\} : v) \end{array} \tag{2}$$

for each edge $[u, v] \in E$.

Observe that formula (2) is formed by Horn clauses which are also binary clauses, and therefore we can conclude that 2-SAT is NP-complete for monosigned with negation CNF formulas. Moreover, since formula (2) is logically equivalent to the signed CNF formula $(\{2,3\} : u \vee \{2,3\} : v) \wedge (\{1,3\} : u \vee \{1,3\} : v) \wedge (\{1,2\} : u \vee \{1,2\} : v)$, we can also conclude that 2-SAT is NP-complete for signed CNF formulas.

In (Ansótegui *et al.* 2002) we have shown that, given two many-valued clauses $S : p \vee D_1$ and $S' : p \vee D_2$, the *many-valued binary resolution rule*

$$\frac{S : p \vee D_1 \quad S' : p \vee D_2}{D_1 \vee D_2} \quad \text{if } S \cap S' = \emptyset \tag{3}$$

is refutation complete for many-valued CNF formulas (and, of course, for the subclasses of regular and monosigned CNF formulas). Since the number of resolvents is polynomial when both parent clauses are binary, we have that 2-SAT is polynomially solvable for many-valued, regular and monosigned CNF formulas.

[1]Since monosigned CNF formulas have no notion of polarity, it makes no sense to define monosigned Horn CNF formulas. The same happens for arbitrary signed CNF formulas. We have Horn CNF formulas when we extend monosigned CNF formulas with negation; in this case, a clause is Horn if it contains at most one non-negated (positive) monosigned literal.

Rule (3) it is not complete (i) for signed CNF formulas, and (ii) for monosigned CNF formulas with negation. For instance, given the truth value set $T = \{1, 2, 3\}$, the signed CNF formula $\{1, 2\} : p \wedge \{1, 3\} : p \wedge \{2, 3\} : p$ is clearly unsatisfiable but cannot be proved with that rule. The same happens if we incorporate negation to monosigned literals: Observe that $\{1, 2\} : p \wedge \{1, 3\} : p \wedge \{2, 3\} : p$ is equivalent to $\neg\{3\} : p \wedge \neg\{2\} : p \wedge \neg\{1\} : p$. For obtaining a complete calculus for that sort of formulas we need the following inference rules (Murray & Rosenthal 1993):

$$\frac{S_1 : p \vee D_1 \qquad S_2 : p \vee D_2}{(S_1 \cap S_2) : p \vee D_1 \vee D_2} \qquad \frac{\emptyset : p \vee D}{D} \tag{4}$$

Note that, unlike many-valued binary resolution, the literal resolved upon does not necessarily vanish and a so-called *residue* remains. Unfortunately, computing and propagating residues complicates considerably the data structures, and increases the time needed to apply resolution rules. For instance, unit propagation can be naturally extended to many-valued CNF formulas. However, unit propagation —which is the main constraint propagation technique in satisfiability solvers— needs to consider residues for signed and monosigned with negation CNF formulas.

The above complexity results remain valid in case the domain is continuous and all the signs are intervals. They are also valid in case the domain, say $T = \{i_1, i_2, \ldots, i_n\}$, is discrete and signs are subsets of T of the form $\{i_j, i_{j+1}, i_{j+2}, \ldots, i_{j+k}\}$.

MV-SAT Encodings

In this section we define MV-SAT encodings, and analyze their complexity, for a number of combinatorial problems: quasigroup with holes completion (Achlioptas *et al.* 2000), graph coloring (Selman & Kautz 1993), all interval series (Hoos 1998), and round robin scheduling (Gomes *et al.* 1998). As we will see, MV-SAT encodings for some of those problems are much compact than SAT encodings. On the other hand, MV-SAT encodings are much natural and closer to CSP encodings; in the sense of their capability to succinctly express different types of constraints, dealing with domains, and using a similar number of variables. For instance, suppose we want to encode the constraint $p_1 = 1 \vee p_2 = 3 \vee p_3 \neq 4 \vee p_4 \neq 100$ and that the domain of values is $T = \{1, \ldots, 100\}$. It can be easily encoded as a many-valued CNF formula as follows: $\{1\} : p_1 \vee \{3\} : p_2 \vee {\downarrow}3 : p_3 \vee {\uparrow}5 : p_3 \vee {\downarrow}99 : p_4$.

Observe that for representing negation we use one or two regular literals, but if we only consider monosigned literals, we need disjunctions of $|N| - 1$ monosigned literals. Moreover, monosigned literals— unlike regular literals— are not well-suited for dealing with continuous domains. On the other hand, if we only consider regular literals, we cannot represent in a compact way disjunctions of positive constraints, because a positive constraint, using only regular literals, has to be represented as the conjunction of two regular literals. So, encoding a disjunction of positive constraints using only regular literals produces a CNF formula with an exponential number of clauses.

Finally, observe that we obtain certain computational advantages from having an ordering on our truth values. The reason for this is that we can use data structures that keep track of *intervals* or *ranges* of truth values by storing their end-points, instead of manipulating only discrete lists of truth values.

Graph Coloring Encoding

Given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, the k-colorability problem is the problem of deciding if there is a function $c : V \rightarrow \{1, \dots, k\}$ such that for each edge $[u, v] \in E$ we have $c(u) \neq c(v)$.

Our MV-SAT encoding of the k-colorability problem for a graph $G = (V, E)$ employs $|V|$ propositional variables (SAT encodings use $k \cdot |V|$ variables) and a domain $T = \{1, \dots, k\}$ with k elements: there is one variable for each vertex, and each element of the domain represents one color. Then, the MV-SAT instance is formed by k many-valued clauses for each edge $[u, v] \in E$:

$$\begin{array}{l}
\uparrow 2 : u \vee \uparrow 2 : v \\
\{1\} : u \vee \uparrow 3 : u \vee \{1\} : v \vee \uparrow 3 : v \\
\qquad\qquad \vdots \\
\downarrow (i-1) : u \vee \uparrow (i+1) : u \vee \downarrow (i-1) : v \vee \uparrow (i+1) : v \\
\qquad\qquad \vdots \\
\downarrow (k-2) : u \vee \{k\} : u \vee \downarrow (k-2) : v \vee \{k\} : v \\
\downarrow (k-1) : u \vee \downarrow (k-1) : v
\end{array}$$

The intended meaning of the first clause is that vertex u and vertex v do not have both color 1, of the second that do not have both color 2, and so on. Observe that from the definition of interpretation we can ensure that every vertex is colored exactly with one color. The number of clauses of the many-valued encoding is in $O(|T| \cdot |E|)$, where $|T|$ is the number of colors and $|E|$ is the number of edges. The simplest SAT encoding for that problem is in $O(|V| + |T| \cdot |E|)$, where $|V|$ is the number of vertices.

Quasigroups With Holes (QWH) Encoding

Given n colors, a quasigroup, or Latin square, is defined by an n by n table, where each entry has a color and where there are no repeated colors in any row or any column; n is called the *order* of the quasigroup.

The problem of whether a partially colored quasigroup can be completed into a full quasigroup by assigning colors to the open entries of the table is called the quasigroup with holes problem in case the partially colored quasigroup is obtained from a full quasigroup by "punching" holes into it (cf. (Kautz *et al.* 2001) for further details). This way of generating instances is particularly useful for testing incomplete SAT solvers because all the instances generated are satisfiable, and has the advantage that the hardness can be finely controlled by the number of open entries in the table.

Our MV-SAT encodings of quasigroup of order n employ n^2 propositional variables (like CSP encodings) and a domain with n elements: there is one variable per cell, and each element of the domain represents one of the colors that can be assigned to the cell. Then, the MV-SAT instance is formed by a set of clauses that state that each row and each column is a permutation of n colors.

We consider two encodings of permutation, one is monosigned (i.e. literals only contain monosigned signs) and the other regular (i.e. literals only contain regular signs). Given a vector $v = (v_1, \dots, v_n)$, the first basic encoding (P1) that represents that v is a permutation of n elements is defined as follows:
For each $t \in \{1, \cdots, n\}$ we define the clause

$$\{t\} : v_1 \vee \cdots \vee \{t\} : v_n;$$

and the second basic encoding (P2) is defined as follows:
For each $i, j \ (1 \leq i, j \leq n)$ such that $i < j$, and for each $t \in \{1, \cdots, n\}$, we define the clause

$$\downarrow (t-1) : v_i \vee \uparrow (t+1) : v_i \vee \downarrow (t-1) : v_j \vee \uparrow (t+1) : v_j$$

The quasigroup SAT encodings defined in (Kautz *et al.* 2001) employ n^3 propositional variables while our MV-SAT encodings only employ n^2 variables; the number of clauses is in $O(n^4)$ for the SAT encoding and the MV-SAT encoding that uses P2 for representing permutation, but the MV-SAT encoding that uses encoding P1 reduces dramatically the number of clauses: it only employs $2n^2$ clauses. We call the encoding with P1 monosigned encoding, the one with P2 regular encoding, and the redundant encoding with both P1 and P2 the hybrid encoding.

All Interval Series Encoding

The All Interval Series (AIS) problem of size n is formulated as follows: find two vectors s and v, such that

1. $s = (s_1, \dots, s_n)$ is a permutation of $\{0, 1, \dots, n-1\}$.
2. $v = (|s_2 - s_1|, |s_3 - s_2|, \dots, |s_n - s_{n-1}|)$ is a permutation of $\{1, 2, \dots, n-1\}$.

Our MV-SAT encoding of the AIS problem of size n uses $2n - 1$ variables (like CSP encodings) and a domain with n elements: there are n $(n-1)$ variables for vector s (v), and each element of the domain represents one of the values that can be assigned to the elements of vectors s and v. Then, an MV-SAT instance is formed by (i) a set of clauses that state that s is a permutation of $\{0, 1, \dots, n-1\}$; (ii) a set of clauses that state that v is a permutation of $\{1, \dots, n-1\}$; and (iii) a set of clauses that state that $v_i = |s_{i+1} - s_i|$ for each i, where $1 \leq i < n$.

The clauses of (iii) are defined as follows: For each two different $x, y \in \{0, 1, \dots, n-1\}$ and for each $i \ (1 \leq i \leq n-1)$, we define the clause

$$\begin{array}{l}
\downarrow (x-1) : s_i \vee \uparrow (x+1) : s_i \vee \\
\downarrow (y-1) : s_{i+1} \vee \uparrow (y+1) : s_{i+1} \vee \{z\} : v_i,
\end{array}$$

where $z = |x - y|$.

We have considered the monosigned encoding (P1) to represent that vectors s and v are permutations of natural number because it allow us to obtain better experimental results. In this case, the MV-SAT encoding uses only $O(n)$ clauses for encoding permutations while the SAT encoding uses $O(n^2)$ clauses.

Round Robin Timetabling Encoding

The round robin timetabling problem for n teams (n even) is the problem of constructing an sports league such that:

1. There are $n/2$ fields, and the season lasts $n-1$ weeks.
2. Every team plays one game in each week of the season.
3. Every two teams play each other exactly once.
4. Each week, every field is scheduled for one game.
5. No team plays more than twice in the same field during the season.

The meeting between two teams is called a game and takes place in a slot; i.e., in a particular field in a particular week. An n-team round robin timetable contains $n(n-1)/2$ slots and slots are filled in with games. A game is represented by a pair of teams (t_1, t_2).

Our MV-SAT encoding of the n-team round robin problem uses $n \cdot (n-1)$ propositional variables and a domain with $n/2$ elements: there is a variable for every combination team–week, and each element of the domain represents one of the fields that can be assigned to a team in a particular week. Then, an MV-SAT instance is defined is as follows:

1. *Every team plays one game in each week of the season.* This constraint is satisfied because an interpretation assigns exactly one field to every combination team–week. Nevertheless, we have to add clauses for ensuring that there are no more than two teams assigned to a particular slot: For each three different teams t_1, t_2, t_3, for each week w, and for every field f, we define the clause
$$\downarrow (f-1) : p^w_{t_1} \vee \uparrow (f+1) : p^w_{t_1} \vee$$
$$\downarrow (f-1) : p^w_{t_2} \vee \uparrow (f+1) : p^w_{t_2} \vee$$
$$\downarrow (f-1) : p^w_{t_3} \vee \uparrow (f+1) : p^w_{t_3}$$
2. *Every two teams play each other exactly once.* For each two teams t_1, t_2 such that $t_1 < t_2$, for each two different weeks w_1, w_2, and for each two fields f_1, f_2, we define the clause
$$\downarrow (f_1-1) : p^{w_1}_{t_1} \vee \uparrow (f_1+1) : p^{w_1}_{t_1} \vee$$
$$\downarrow (f_1-1) : p^{w_1}_{t_2} \vee \uparrow (f_1+1) : p^{w_1}_{t_2} \vee$$
$$\downarrow (f_2-1) : p^{w_2}_{t_1} \vee \uparrow (f_2+1) : p^{w_2}_{t_1} \vee$$
$$\downarrow (f_2-1) : p^{w_2}_{t_2} \vee \uparrow (f_2+1) : p^{w_2}_{t_2}$$
3. *No team plays more than twice in the same field over the course of the season.* For each team t, for each field f, and for each three different weeks w_1, w_2, w_3, we define the clause
$$\downarrow (f-1) : p^{w_1}_{t} \vee \uparrow (f+1) : p^{w_1}_{t} \vee$$
$$\downarrow (f-1) : p^{w_2}_{t} \vee \uparrow (f+1) : p^{w_2}_{t} \vee$$
$$\downarrow (f-1) : p^{w_3}_{t} \vee \uparrow (f+1) : p^{w_3}_{t}$$

The number of clauses of our pure regular encoding is in $O(n^6)$, like the number of clauses for the best known Boolean encoding of the problem (Béjar & Manyà 2000). It is possible to define both SAT and MV-SAT encodings with a number of clauses in $O(n^4)$, but their performance profile is much worse than encodings in $O(n^6)$.

```
procedure MV-WalkSAT

Input: a many-valued CNF formula $\Gamma$, MaxChanges, MaxTries and $\omega$
Output: a satisfying interpretation of $\Gamma$, if found
  for $i := 1$ to MaxTries
   $I$ := a randomly generated interpretation for $\Gamma$;
   for $j := 1$ to MaxChanges
    if $I$ satisfies $\Gamma$ then return $I$;
    $C$:= an unsatisfied clause of $\Gamma$;
    $S := \{ (p,k) \mid S' : p \in C, k \in S' \}$;
    $u := min ( \{broken((p,k),\Gamma) \mid (p,k) \in S\} )$;
    $p'$ := a variable from $\{p \mid broken((p,k),\Gamma) = u, (p,k) \in S\}$;
    $k'$ := a value from $\{k \mid broken((p',k),\Gamma) = u, (p',k) \in S\}$;
    if $u > 0$ then
     with probability $\omega$
      $p'$ := a variable from the set $\{p \mid (p,k) \in S\}$;
      $k'$ := a value from the set $\{k \mid (p',k) \in S\}$;
    end if
    $I := I$ with the truth assignment of $p'$ changed to $k'$;
   end for
  end for
  return "no satisfying interpretation found";
```

Figure 1: The MV-WalkSAT procedure

MV-SAT Solvers

To obtain competitive problem solving approaches we need both suitable encodings and fast solvers. In our research program we are designing and implementing complete and incomplete solvers for MV-SAT. Concerning systematic search, we are currently implementing Mv-Satz (Ansótegui *et al.* 2002), which is an extension of Satz (Li & Anbulagan 1997). Our preliminary results are quite promising, at least for the benchmarks proved so far. Mv-Satz is, to our best knowledge, the first complete many-valued solver that could compete with state-of-the-art Boolean solvers.

Concerning local search, there exists two efficient many-valued extensions of WalkSAT (Selman, Kautz, & Cohen 1994): NB-WalkSAT (Frisch & Peugniez 2001), dealing only with monosigned CNF formulas; and Regular-WalkSAT (Béjar 2000; Béjar *et al.* 2001; Béjar & Manyà 1999c), dealing only with regular CNF formulas.

In this paper we describe MV-WalkSAT — the pseudocode is shown in Figure 1—, which is a local search solver for many-valued CNF formulas that we have recently implemented, and that builds on Regular-WalkSAT. MV-WalkSAT incorporates monosigned literals in addition to regular literals.

MV-WalkSAT tries to find a satisfying interpretation for a many-valued CNF formula Γ performing a greedily biased walk through the space of possible interpretations. It starts with a randomly generated interpretation I. If I does not satisfy Γ, it proceeds as follows: (i) it randomly chooses an unsatisfied clause C, (ii) it chooses a variable-value pair (p', k') from the set S of pairs (p,k) such that C is satisfied by the current interpretation I if the truth value that I assigns to p is changed to k, and (iii) it creates a new interpretation I' that is identical to I except that $I'(p') = k'$. Such changes are repeated until either a satisfying interpretation is found or a pre-set maximum number of changes (Max-

instance	WalkSAT			MV-WalkSAT			R-Novelty+			MV-R-Novelty+		
	ω	flips	time	ω	flips	time	ω	flips	time	ω	flips	time
DSJC125.5	13	$4.41 \cdot 10^6$	69.10	14	$8.91 \cdot 10^5$	32.58	11	$11.2 \cdot 10^5$	13.22	47	$2.5 \cdot 10^5$	9.33
DSJC250.5	15	$2.16 \cdot 10^6$	104.30	16	$5.16 \cdot 10^5$	63.74	12	$4.75 \cdot 10^5$	21.36	47	$1.6 \cdot 10^5$	18.02

Table 1: Experimental results for the graph coloring problem (DIMACS instances: DSJC125.5.col (17 colors) and DSJC250.5.col (29 colors)). The parameter h of R-Novelty+ and MV-WalkSAT was set to 15. Mean time in seconds.

	order 33			order 36		
algorithm	ω	flips	time	ω	flips	time
WalkSAT	30	$9.41 \cdot 10^6$	21.49	22	$89.61 \cdot 10^6$	255.52
MV-WalkSAT	20	$1.63 \cdot 10^6$	3.09	18	$10.59 \cdot 10^6$	18.78

Table 2: Experimental results for QWHs of order 33 and 36 in the peak of hardness. Mean time in seconds.

Changes) is reached. This process is repeated as needed, up to a maximum of MaxTries times. In step (ii), MV-WalkSAT calculates, for each pair (p, k) in S, the number of broken clauses; i.e. the number of clauses that are satisfied by I but that would become unsatisfied if the assignment of p is changed to k. If the minimum number of broken clauses found (u) is greater than zero then it randomly chooses, with probability ω, a pair (p', k') from S, or it randomly chooses, with probability $1 - \omega$, a pair (p', k') from those pairs for which the number of broken clauses is u. If $u = 0$, then it randomly chooses a pair from those pairs for which $u = 0$. Apart from this basic heuristic, we have also incorporated more advanced heuristics to escape from local minima such as G, Novelty, R-Novelty, and R-Novelty+ (Selman, Kautz, & Cohen 1994; McAllester, Selman, & Kautz 1997; Hoos 1998).

As in Regular-WalkSAT, MV-WalkSAT uses specialized incremental data structures for handling changes in the value of flip counters. Updating the value of flip counters is the key point of the algorithm, because every time the algorithm performs a flip, lots of flips counters can be affected. When a regular literal $S : p$, that has $|S|$ flips that satisfy p, appears in a clause affected by a flip, the algorithm *keeps track* of the updating needed to be performed in the $|S|$ flips with a single operation. Because S is either of the form $\downarrow i$: or $\uparrow j$:, all these flips can be described with a starting truth value and a direction. So, the algorithm uses a special counter that indicates an updating that should be performed to all the $|S|$ flips. The same counter is used to accumulate further updates to the same set of flips. Later on, when the flip selection function needs to consult the value of the flip counters of p, their actual value is updated using the information accumulated in the special counters. MV-WalkSAT, in addition to these these data structures incorporates specific mechanisms for handling monosigned literals, getting a final worst-case time complexity of $O(|C|)$ when processing a clause with $|C|$, regular or monosigned, literals.

Actually, due to the improvements performed in the current implementation, MV-WalkSAT is slightly faster on regular CNF formulas than the last version of Regular-WalkSAT (Béjar *et al.* 2001). On the other hand, MV-WalkSAT performs less flips than NB-WalkSAT when solving a monosigned instance (at least for the problems we have tried) because it uses a slightly different way of picking the next variable to flip: MV-WalkSAT first chooses a variable from among those in the clause and then chooses one of the values that appear with that variable, whereas NB-WalkSAT simply chooses from among all the literals in the clause. The cost of performing a flip is approximately equal in NB-WalkSAT and MV-WalkSAT. NB-WalkSAT incorporates the basic heuristic, but not G, Novelty, R-Novelty, and R-Novelty+. Since MV-WalkSAT incorporates more heuristics to escape from local minima, and gives slightly better results on both regular and monosigned CNF formulas, we have solved pure regular and pure monosigned encodings with MV-WalkSAT in our empirical evaluation.

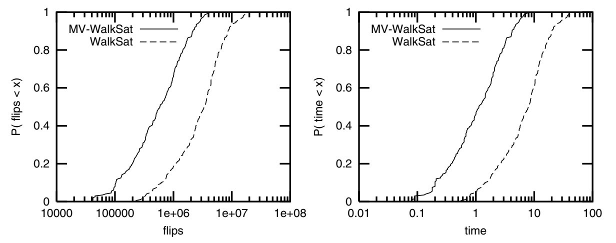

Figure 2: RLDs (left) and RTDs (right) on the median instance for order 33

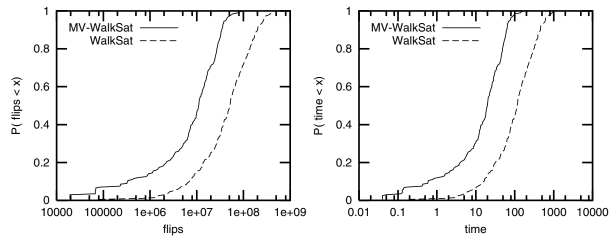

Figure 3: RLDs (left) and RTDs (right) on the hardest instance for order 33

Experimental Results

In this section we present in detail the experimental investigation we have conducted in order to compare the perfor-

size	WalkSAT			MV-WalkSAT			R-Novelty			MV-R-Novelty		
	ω	flips	time	ω	flips	time	ω	flips	time	ω	flips	time
14	5	$2.11 \cdot 10^6$	9.03	5	$4.1 \cdot 10^5$	1.9	10	$7.5 \cdot 10^5$	3.61	11	$0.8 \cdot 10^5$	0.62
16	4	$13.4 \cdot 10^6$	59.03	4	$4.29 \cdot 10^6$	21.8	8	$5 \cdot 10^6$	27.62	8	$4.7 \cdot 10^5$	4.51
18	3	$63.73 \cdot 10^6$	358.54	3	$26.33 \cdot 10^6$	145.6	6	$24.7 \cdot 10^6$	160.05	7	$2.07 \cdot 10^6$	24.18

Table 3: Experimental results for the all interval series problem. Mean time in seconds.

n	WalkSAT/G			MV-WalkSAT/G		
	ω	flips	time	ω	flips	time
16	14	$30.2 \cdot 10^6$	4098	4	$2.3 \cdot 10^6$	1092
18	12	$261.2 \cdot 10^6$	45258	2	$27.0 \cdot 10^6$	14987

Table 4: Experimental results for round robin problem. Mean time in seconds.

mance between SAT and MV-SAT local search algorithms when solving the combinatorial problems described above. In our experiments we used approximately optimal noise parameters (ω), and a very high cutoff value in order to get a solution in every try. We performed 200 tries for every instance. For all the benchmarks we report on results using the basic heuristic of WalkSAT and MV-WalkSAT, and also report on results using the best performing heuristic we have found for every benchmark among G, Novelty, R-Novelty, and R-Novelty+. All the experiments were performed on 1 GHz Pentiums III with 512 Mb of memory.

For graph coloring instances, we considered two challenging instances taken from the DIMACS repository that are beyond the reach of complete SAT solvers. We found that the best performing encoding is the pure regular encoding described above. Table 1 shows results for WalkSAT and MV-WalkSAT with the basic heuristic and R-Novelty+. The table shows the mean solution times (since we are dealing with local search methods, the means are well-behaved, i.e., no large outliers). The results show that MV-WalkSAT slightly outperforms WalkSAT, both in terms of time and number of flips. The difference in terms of time is less significant. This is because the cost of performing flips is somewhat larger for MV-WalkSAT.

For the QWH domain, we have compared the performance of WalkSAT and MV-WalkSAT when solving instances at the hardness peak of the phase transition (Kautz *et al.* 2001). Table 2 shows results for quasigroups of order 33 and 36; we considered 100 instances of each order. We found that the basic heuristic of MV-WalkSAT, together with the monosigned encoding, is the best approach for the QWH domain. For SAT, we used the best performing heuristic (basic) and encoding (so-called 3-D in (Kautz *et al.* 2001)). We observe that the MV-WalkSAT clearly outperforms WalkSAT. Also, note that, in this domain, SAT based approaches outperform standard CSP approaches (Achlioptas *et al.* 2000; Kautz *et al.* 2001).

Although the average complexity of solving instances from a problem domain distribution gives us a valuable information about the difficulty of the problem, the complexity of solving individual instances obtained with the same parameters can vary drastically from instance to instance. So, a more detailed analysis requires a study of the complexity of solving individual instances. To do so, we have constructed empirical Run-time distributions (RTDs) and Run-length (number of flips needed) distributions (RLDs) for WalkSAT and MV-WalkSAT when solving the same instance. The methodology followed has been the one used in (Hoos 1998). We have focused our attention on the median instance and the hardest instance of a given test-set. Here we present results for the test-set of quasigroups of order 33. Figure 2 shows the RLDs and RTDs for MV-WalkSAT and WalkSAT on the median instance. These empirical RLDs, in the cumulative form shown, give the probability that the algorithm finds a solution for the instance in less than the number of flips of the $x-$axis (similarly in the RTDs). We observe that MV-WalkSAT strictly dominates WalkSAT; *i.e.*, the probability of finding a solution with MV-WalkSAT in less than x flips is always greater than the probability of finding a solution with WalkSAT. MV-WalkSAT dominates the WalkSAT in the run time. Figure 3 shows the same results but for the hardest instance of the same test-set. We observe a similar relative difference between the run time performance of the two algorithms.

Table 3 shows results for the AIS problem using an hybrid encoding with monosigned and regular clauses, and Table 4 for the round robin problem using a pure regular encoding; these are the best performing encodings we have found. We see a dramatic reduction on both time and flips when using the MV-SAT encodings. The basic heuristic of both WalkSAT and MV-WalkSAT was not able to solve the round robin instances after 24 hours; we solved the instances with heuristic G. This fact, together with the good results obtained for AIS with R-Novelty, highlight the importance of not limiting our investigation to the basic heuristic as has been done so far for NB-WalkSAT. As a byproduct, we have also provided experimental evidence that NB-WalkSAT can be used to solve combinatorial problems other than the graph coloring instances of (Frisch & Peugniez 2001).

Conclusions

In this paper we have shown that MV-SAT is a very competitive problem solving approach that allows us to model combinatorial problems in a concise and natural way like in CSP, and at the same time to develop fast solvers operating on uniform encodings like in SAT. Our results provide experimental evidence that MV-SAT outperforms SAT on a range of hard combinatorial problems. The improvement in performance is due to both suitable encodings and fast solvers that incorporate advanced heuristics to escape from local optima. As a constraint programming language, MV-SAT combines the advantages of monosigned and regular encodings by allowing us to deal with pure monosigned and pure regular, as well as with hybrid encodings. Our next step to improve MV-SAT is to develop CSP-like constraint propagation techniques, to consider continuous domains, and to incorporate global constraints such as "all-diff".

Acknowledgements

This research was partially funded by the DARPA contracts F30602-00-2-0530 and F30602-00-2-0596, by project CICYT TIC2001-1577-C03-03, and by the Intelligent Information Systems Institute, Cornell University, funded by AFRL/AFOSR (F49620-01-1). The first author is supported by a doctoral fellowship of Intelligent Software Components (iSOCO).

References

Achlioptas, D.; Gomes, C. P.; Kautz, H.; and Selman, B. 2000. Generating satisfiable problem instances. In *Proc. of AAAI-2000*.

Ansótegui, C.; Béjar, R.; Cabiscol, A.; Li, C.-M.; and Manyà, F. 2002. Resolution methods for many-valued CNF formulas. In *Fifth International Symposium on the Theory and Applications of Satisfiability Testing, SAT-2002*.

Baaz, M., and Fermüller, C. G. 1995. Resolution-based theorem proving for many-valued logics. *Journal of Symbolic Computation* 19:353–391.

Beckert, B.; Hähnle, R.; and Manyà, F. 1999. Transformations between signed and classical clause logic. In *Proceedings, 29th International Symposium on Multiple-Valued Logics (ISMVL)*, 248–255.

Beckert, B.; Hähnle, R.; and Manyà, F. 2000. The 2-SAT problem of regular signed CNF formulas. In *Proceedings, 30th International Symposium on Multiple-Valued Logics (ISMVL)*, 331–336.

Béjar, R., and Manyà, F. 1999a. A comparison of systematic and local search algorithms for regular CNF formulas. In *Proc. of ECSQARU'99*, 22–31. Springer LNAI 1638.

Béjar, R., and Manyà, F. 1999b. Phase transitions in the regular random 3-SAT problem. In *Proc. of ISMIS'99*, 292–300. Springer LNAI 1609.

Béjar, R., and Manyà, F. 1999c. Solving combinatorial problems with regular local search algorithms. In *Proc. of LPAR'99*, 33–43. Springer LNAI 1705.

Béjar, R., and Manyà, F. 2000. Solving the round robin problem using propositional logic. In *Proc. of AAAI-2000*.

Béjar, R.; Cabiscol, A.; Fernández, C.; Manyà, F.; and Gomes, C. P. 2001. Capturing structure with satisfiability. In *Proc. of CP-2001*, 137–152. Springer LNCS 2239.

Béjar, R.; Hähnle, R.; and Manyà, F. 2001. A modular reduction of regular logic to classical logic. In *Proceedings, 31st International Symposium on Multiple-Valued Logics (ISMVL)*.

Béjar, R. 2000. *Systematic and Local Search Algorithms for Regular-SAT*. Ph.D. Dissertation, Universitat Autònoma de Barcelona.

Cheeseman, P.; Kanefsky, B.; and Taylor, W. M. 1991. Where the really hard problems are. In *Proc. of IJCAI-91*.

Frisch, A. M., and Peugniez, T. J. 2001. Solving non-boolean satisfiability problems with stochastic local search. In *Proc. of IJCAI-2001*.

Gomes, C. P., and Selman, B. 1997. Problem structure in the presence of perturbations. In *Proc. of AAAI-97*.

Gomes, C. P; Selman, B.; McAloon, K.; and Tretkoff, C. 1998. Randomization in backtrack search: Exploiting heavy-tailed profiles for solving hard scheduling problems. In *Proc. of AIPS-98*.

Hoos, H. H. 1998. *Stochastic Local Search – Methods, Models, Applications*. Ph.D. Dissertation, Department of Computer Science, Darmstadt University of Technology.

Kautz, H. A., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proc. of AAAI-96*.

Kautz, H. A.; Ruan, Y.; Achlioptas, D.; Gomes, C. P.; Selman, B.; and Stickel, M. 2001. Balance and filtering in structured satisfiable problems. In *Proc. of IJCAI-2001*.

Li, C. M., and Anbulagan. 1997. Look-ahead versus look-back for satisfiability problems. In *Proc. of CP'97*.

Marques-Silva, J. P., and Guerra, L. 1999. Algorithms for satisfiability in combinational circuits based on backtrack search and recursive learning. In *Proc. of XII Symposium on Integrated Circuits and Systems Design (SBCCI)*.

McAllester, D.; Selman, B.; and Kautz, H. 1997. Evidence for invariants in local search. In *Proc. of AAAI-97*.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and easy distributions of SAT problems. In *Proc. of AAAI-92*.

Moskewicz, M.; Madigan, C.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an efficient sat solver. In *39th Design Automation Conference*.

Murray, N. V., and Rosenthal, E. 1993. Signed formulas: A liftable meta logic for multiple-valued logics. In *Proc. of ISMIS'93, Trondheim, Norway*, 275–284. Springer LNAI 689.

Régin, J.-C. 1994. A filtering algorithm for constraints of difference in CSPs. In *Proc. of AAAI'94*.

Selman, B., and Kautz, H. A. 1993. Domain-independent extensions of GSAT: Solving large structured satisfiability problems. In *Proc. of IJCAI-93*.

Selman, B.; Kautz, H. A.; and Cohen, B. 1994. Noise strategies for improving local search. In *Proc. of AAAI-94*.

Stergiou, K., and Walsh, T. 1999. The difference all-difference makes. In *Proc. of IJCAI'99*.