Going Non-Clausal

Zbigniew Stachniak
Department of Computer Science
York University, Canada
e-mail: zbigniew@cs.yorku.ca

1 Introduction

The use of clauses and clause-like normal forms for representing knowledge and reasoning is widespread and has many proponents. And rightly so: exploiting the unique syntactic structure of a clause can often result in a surprisingly simple formalization of a reasoning system, a proof procedure, or a satisfiability checker. The approach hinges upon a translation of the object language formulas into formulas in a specified normal form. In classical propositional logic, such a translation into clauses (a CNF-translation) increases the length of a formula (in some cases exponentially, when equivalence preserving CNF-translation is used); it may also enlarge the size of the search space (when a structure preserving CNF-translation is employed).

In this paper I present a non-clausal stochastic local search satisfiability procedure for classical propositional logic that avoids the translation of an input formula into clauses. This new procedure, called polWSAT, resembles clausal Walksat solver of Selman, Kautz, and Cohen ([4]). polWSAT and Walksat coincide in the domain of formulas in conjunctive normal form and, hence, their performance on clausal instances of SAT is similar. On free-form formulas, however, polWSAT can significantly outperform Walksat solver augmented with a structure preserving CNF-translation. The empirical results reported in this paper, that concern the relative efficiency of polWSAT and Walksat, provide some evidence in support of this claim.

In the paper I also discuss possible improvements of the polWSAT solver as well as its extensions to finitely-valued propositional logics.

2 polWSAT Solver

Similarly to Walksat, polWSAT performs a stochastic local search over the space of truth-assignments. Its design follows that of the Walksat procedure with the following exceptions:

- Input formulas are free form.
- Variable selection heuristic is based on the comparison between polarity values and truth-values of subformulas of an input formula.

The search for a variable whose truth-value is to be modified proceeds in two steps. First, a list *CandList* of variables in the problem instance is computed – the variable to be flipped will come from this list. Then, similarly to Walksat, the algorithm makes a 'random walk' with probability p (selecting a variable from *CandList* at random) or it makes a 'greedy move' with probability $1-p$ and selects a variable from *CandList* whose flip minimizes a certain parameter called the *clash value* of an input formula which is defined in Definition B.

DEFINITION A. *Let α be a formula, let h be a truth-value assignment, and let β be an occurrence of a subformula in α. The polarity of β is said to clash with the truth-value $h(\beta)$, if either the polarity of β is positive and $h(\beta) = 0$, or the polarity of β is negative and $h(\beta) = 1$.*

DEFINITION B. *Let α, β, and h be as in Definition A. Moreover, suppose that α is in negation normal form. The cumulative clash value for β, denoted by $clash(\beta, h)$, is defined recursively as follows:*

(i) if β is a literal occurrence (i.e. an occurrence of a propositional variable or its negation) whose polarity clashes with $h(\beta)$, then $clash(\beta, h) = 1$, else $clash(\beta, h) = 0$;

(ii) let $\beta = \beta_0 \vee \beta_1$; if the polarity of β clashes with $h(\beta)$, then $clash(\beta, h) = min(clash(\beta_0, h), clash(\beta_1, h))$, else $clash(\beta, h) = 0$;

(iii) let $\beta = \beta_0 \wedge \beta_1$; if the polarity of β clashes with $h(\beta)$, then $clash(\beta, h) = clash(\beta_0, h) + clash(\beta_1, h)$, else $clash(\beta, h) = 0$.

Intuitively, the clash value of an input formula α with respect to h estimates the amount of work required to modify h to a satisfying assignment (measured in the number of flips), if at all possible.

The *CandList* consists of variables of α such that the change of their truth-values would affect the clash value of α. Since an input formula is satisfiable by an assignment h if and only if its clash value with respect to h is 0, the heuristic objective is to minimize the clash parameter. Both the clash value of an input formula and the *CandList* can be computed efficiently.

polSAT and Walksat coincide in the domain of CNF formulas (i.e. formulas in conjunctive normal form): when an input formula α is a conjunction of clauses, then *CandList* becomes a list of variables that occur in a randomly selected clause of α that is false under the current assignment h. Furthermore, the *clash value* of α becomes the number of clauses false under h.

3 Empirical Results

To estimate the relative efficiency of polWSAT and Walksat, I tested both algorithms on randomly generated formulas from the class $H\text{-}CNF(m,n)$ of all formulas of the form:

$$(c_{11} \vee c_{12}) \wedge (c_{21} \vee c_{22}),$$

where c_{11}, c_{12}, c_{21}, and c_{22} are instances of 3CNF. Each formula $c_{i,j}$ contains the same m variables $p_0, \ldots, p_{m-1}$. Moreover, each $c_{i,j}$ consists of exactly n clauses. I shall call such formulas – *hyper-CNF formulas*. For example, if

$$c11 = (p_0 \vee \neg p_1 \vee p_3) \wedge (p_0 \vee \neg p_2 \vee \neg p_3),$$

$$c12 = (p_1 \vee \neg p_2 \vee p_3) \wedge (p_0 \vee \neg p_1 \vee p_3),$$

$$c21 = (\neg p_0 \vee \neg p_1 \vee \neg p_3) \wedge (p_0 \vee \neg p_1 \vee \neg p_2),$$

$$c22 = (p_1 \vee p_2 \vee p_3) \wedge (p_0 \vee \neg p_1 \vee \neg p_3),$$

then the resulting formula is a hyper-CNF formula from the class $H\text{-}CNF(4,2)$. The classes $H\text{-}CNF(m,n)$ have a number of features that make them suitable candidates for the testing of non-clausal SAT solvers. Random generation of hyper-CNF formulas reduces to the task of the random generation of 3CNF instances. Moreover, the ratio of satisfiable-to-unsatisfiable instances of *hyper-CNF formulas* can be set by adjusting the parameters m and n. For a given value of m, the number of satisfiable hyper-CNFs in $H\text{-}CNF(m,n)$ increases (resp. decreases) when the value of n decreases (resp. increases). For example, this ratio is near 50% for $H\text{-}CNF(10,31)$ and $H\text{-}CNF(50,118)$.

In the empirical studies reported in this section, I used B. Selman's WalkSAT Version 35 program available from the Satisfiability Library. [1] WalkSAT was executed on CNF-formulas obtained from hyper-CNFs using various structure preserving CNF-translations discussed below.

CNF-Translation of Plaisted and Greenbaum. The difference in the performance between polWSAT and WalkSAT augmented with the structure preserving translation of Plaisted and Greenbaum (cf. [3]) was significant. WalkSAT (with default settings: cutoff=100,000 and max_tries=10) was unable to find satisfying assignments for many formulas from $H\text{-}CNF(40, 97)$ that were demonstrated satisfiable by polWSAT. WalkSAT failed to find satisfying assignments for most of the formulas from $H\text{-}CNF(60, 140)$ that were demonstrated satisfiable by polWSAT. During the tests, every hyper-CNF that was demonstrated satisfiable by WalkSAT was also found satisfiable by polWSAT.

3CNF-Translation + (trans). The performance of WalkSAT on hyper-CNFs improved when the structure preserving translation of Plaisted and Greenbaum was used to convert hyper-CNFs into 3CNFs and, in addition, when the resulting set of clauses was simplified using the rule:

(trans) replace clause $\neg A \vee B$ and every clause $\neg B \vee \alpha$ with $\neg A \vee \alpha$, where A and B are new variables introduced during the structure preserving CNF-translation.

Table 1 contains the results of the comparison between polWSAT and WalkSAT augmented with the above mentioned CNF-translation. The tests were performed on randomly generated $H\text{-}CNF(m, n)$ formulas, where $10 \leq m \leq 100$. 200 instances were generated for every class $H\text{-}CNF(m, n)$ used in the study.

[1] WalkSAT Version 35 available from SATLIB, http://www.satlib.org/solvers.html.

$H\text{-}CNF(m,n)$	polSAT			WalkSAT		
	avg nr flips when satisfied	success rate	outperf. by WalkSAT	avg nr flips when satisfied	success rate	outperf. by polWSAT
$H\text{-}CNF(10,31)$	43	100%	0%	2543	100%	0%
$H\text{-}CNF(40,97)$	1199	100%	0%	48817	52%	11.7%
$H\text{-}CNF(60,140)$	1855	80%	0%	69507	11%	40.6%
$H\text{-}CNF(100,225)$	1301	48%	0%	98179	3%	96.8%

TABLE 1: polWSAT and WalkSAT compared. WalkSAT augmented with CNF-translation of Plaisted and Greenbaum into 3CNF and the rule (trans). Parameters for polWSAT: cutoff (max. number of flips) =10,000, tries (number of tries)=10, probability of random walk=0.5. Parameters for WlakSAT: default (cutoff=100,000, tries=10). Note significantly lower cutoff parameter for polWSAT.

In the table, the second column contains average number of flips performed by polWSAT (for 10 tries) when a satisfying assignment was found. The column labeled 'success rate' gives the percentage of polWSAT tries that succeeded when the satisfiability of a given input formula was established either by polWSAT or WalkSAT. The fourth column gives the percentage of hyper-NCFs which were found satisfiable by WalkSAT but not by polWSAT. The meanings of the remaining columns (for WalkSAT) are analogous.

As evident from Table 1, polWSAT is still a superior solver on the class of hyper-CNFs. WalkSAT's performance (the average number of flips performed by a program over 10 tries as well as the success rate) diminishes for $H\text{-}CNF(m,n)$ instances with larger values of m and n: the average success rate for finding solutions falls from 100% for $H\text{-}CNF(10,31)$, to 52% for $H\text{-}CNF(40,97)$, down to 3% for $H\text{-}CNF(100,225)$. At the same time, the percentage of instances of hyper-CNFs that were found satisfiable by polWSAT but not by WalkSAT increases from 0% for $H\text{-}CNF(10,31)$, to 11% for $H\text{-}CNF(40,97)$, up to 96.8% for $H\text{-}CNF(100,225)$.

CNF-Translation of Plaisted and Greenbaum + (trans). Table 2 contains the results of the comparison between polWSAT and WalkSAT augmented with the structure preserving CNF-translation of Plaisted and Greenbaum and the rule (trans). This CNF-translation minimizes the number of additional variables that are introduced during the CNF-translation of hyper-CNF formulas. As seen from Table 2, even with such an efficient encoding of hyper-CNF formulas, WalkSAT does not perform as well as polSAT (in terms of the number of flips and for similar values of cutoff, tries, and probability of random walk).

$H\text{-}CNF(m,n)$	polSAT			WalkSAT		
	avg nr flips when satisfied	success rate	outperf. by WalkSAT	avg nr flips when satisfied	success rate	outperf. by polWSAT
$H\text{-}CNF(10,31)$	43	100%	0%	177	100%	0%
$H\text{-}CNF(20,53)$	226	100%	0%	590	100%	0%
$H\text{-}CNF(40,97)$	1199	100%	0%	4715	100%	0%
$H\text{-}CNF(60,140)$	1855	80%	0%	16405	94%	0%
$H\text{-}CNF(100,225)$	1301	48%	0%	47416	49%	8%

TABLE 2: polWSAT and WalkSAT compared. WalkSAT augmented with CNF-translation of Plaisted and Greenbaum and the rule (trans). Parameters for polWSAT and WalkSAT as in Table 1.

4 Conclusions

This paper presents a new stochastic local search satisfiability algorithm, polWSAT, for classical propositional logic. polWSAT is a non-clausal solver: it exploits the structure of a free-form formula for the purpose of guiding local search for a satisfying assignment. The reported experimental results indicate that for some classes of problems non-clausal SAT solvers may dominate their clausal counterparts and, hence, they may render the search for an efficient CNF encoding of SAT instances redundant.

The performance of polWSAT solver can be improved in a number of ways. For instance, one can incorporate more of the stochastic magic into the basic polWSAT algorithm (as in the Novelty solver, [1]). Or, polWSAT may try to flip the truth-value of more than just one variable at a time. Indeed, if the search heuristic returns a variable p of an input formula α, then the substitution of the logical constant that defines $1 - h(p)$ for p in α (h is the current assignment) may render some variables positive or negative in α. Clearly, the assignment h can be further modified by assigning 1 to variables of positive polarity and 0 to variables of negative polarity.

The polWSAT solver can also be adopted to finitely-valued domains. Indeed, the notion of polarity is readily available for such logics and so is the notion of a clash between the polarity value and the truth-value of a subformula. Multiple-valued logics have been extensively studied in areas ranging from AI and logic programming to circuit design and the analysis of correctness of software and hardware designs. (The annual *IEEE International Symposium on Multiple-Valued Logic* is the premier forum for the state-of-the-art research in multiple-valued systems.) For a number of finitely-valued logics known from the literature, the satisfiability problem is NP-complete (e.g., for the n-valued and infinitely-valued sentential calculi of Łukasiewicz, cf. [2]). One might expect that the stochastic local search satisfiability algorithms can prove as rewarding in the multiple-valued setting as in the case of classical logic.

References

[1] D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. *Proc. AAAI-97* (1997), 321–326.

[2] D. Mundici. Satisfiability in Many-Valued Sentential Logics is NP-Complete. *Theoretical Computer Science* 52 (1987), 145–153.

[3] D. Plaised and S. Greenbaum. A Structure-preserving Clause Form Translation. *J. Symbolic Computation* 2 (1986), 293–304.

[4] B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. *Proc. AAAI-94*, 1994, 337–343.