Solving the Maximum Satisfiability Problem Using an Evolutionary Local Search Algorithm

Mohamed El Bachir Menai[1] and Mohamed Batouche[2]
[1]Artificial Intelligence Laboratory, University of Paris 8, France
[2] Computer Science Department, University Mentouri of Constantine, Algeria

Abstract: *The MAXimum propositional SATisfiability problem (MAXSAT) is a well known NP-hard optimization problem with many theoretical and practical applications in artificial intelligence and mathematical logic. Heuristic local search algorithms are widely recognized as the most effective approaches used to solve them. However, their performance depends both on their complexity and their tuning parameters which are controlled experimentally and remain a difficult task. Extremal Optimization (EO) is one of the simplest heuristic methods with only one free parameter, which has proved competitive with the more elaborate general-purpose method on graph partitioning and coloring. It is inspired by the dynamics of physical systems with emergent complexity and their ability to self-organize to reach an optimal adaptation state. In this paper, we propose an extremal optimization procedure for MAXSAT and consider its effectiveness by computational experiments on a benchmark of random instances. Comparative tests showed that this procedure improves significantly previous results obtained on the same benchmark with other modern local search methods like WSAT, simulated annealing and Tabu Search (TS).*

Keywords: *Constraint satisfaction, MAXSAT, heuristic local search, extremal optimization.*

Received February 29, 2004; accepted June 30, 2004

1. Introduction

An expression is satisfiable if it is true under some interpretation. The SATisfiability problem (SAT) involves finding a truth assignment that satisfies a CNF formula in propositional logic. There are two main approaches to SAT. One class of solutions uses a systematic approach, meaning that each possible assignment of truth values is tested until a solution is found. The other involves randomly testing assignments, and is called stochastic method. The first approach guarantees to find a solution if one exists, but in the worst case can be very inefficient. The second approach is more effective for practically solving large SAT instances but it cannot be used to prove that a given problem instance is unsatisfiable.

The class of all problems for which it is not known if a polynomial time algorithm exists, is called Non-deterministic Polynomial (*NP*) and the related problems are called *NP*-complete. These problems require algorithms of exponential time complexity in the worst case to obtain optimal solutions. SAT is the archetypical *NP* problem, that is all *NP*-complete problems can be polynomially reduced to it (The concept of polynomial-time reductibility) [8]. Indeed, many problems in various areas like artificial intelligence, computer aided design and databases, involve the solution of SAT instances or its variants.

An extension of SAT is the MAXimum SATisfiability problem (MAXSAT) which consists of satisfying the maximum number of clauses of the propositional formula. MAXSAT is of considerable interest from both the theoretical side and the practical side. It plays an important role in the characterization of different approximation classes like Polynomial Time Approximation Algorithm (PTAA) and Polynomial Time Approximation Scheme (PTAS) [34] and many optimization problems can be formulated in this form of satisfiability. MAXSAT is known to be *NP*-hard (an optimization problem that has a related *NP*-complete decision version problem) even when each clause contains exactly two literals MAX2SAT [9, 17]. Since finding an exact solution to this problem requires exponential time, approximation algorithms to find near optimal solutions in polynomial time, appear to be viable. Developing efficient algorithms and heuristics for MAXSAT, can then lead to general approaches for solving combinatorial optimization problems.

Extremal Optimization (EO) was recently introduced as a heuristic search method [5] for approximating solutions to hard optimization problems. It is inspired by the Bak-Sneppen model of biological evolution and natural selection [1] where the high degree of adaptation of most species emerge from the dynamics through a selection against the extremely bad ones [5]. The present work was encouraged by the high quality results obtained with EO on graph coloring [6] and graph partitioning problems [5, 7]. This method has been proved competitive and even

better than the more elaborate general purpose heuristics such as simulated annealing [19] or Tabu Search (TS) [13, 14] on these hard optimization problems.

In this paper, we extend the EO method to solve MAXSAT problems and compare it to the more frequently used local search methods for this problem such as simulated annealing [32], TS [21] and the well known procedure WSAT [29]. This paper is organized as follows. Section 2 presents the MAXSAT problem and related methods to solve it. Section 3 presents the Extremal Optimization method. Its adaptation to handle MAXSAT problems and implementation are discussed in section 4. Experimental results and conclusion follow, respectively, in sections 5 and 6.

2. MAXSAT Problem

Formally, SAT is defined as follows: Given a set of n variables $X = \{x_1, x_2, ..., x_n\}$ and a set of literals over X $L = \{x, \bar{x} / x \in X\}$. A clause C on X is a disjunction of literals. An assignment of Boolean variables is a substitution of these variables by a vector $v \in \{0, 1\}^n$. A clause C is satisfied by an assignment v if the value of the clause equals 1 ($C(v) = 1$), otherwise the clause is unsatisfied ($C(v) = 1$). Given m clauses C_1, C_2, …, C_m, the SAT problem asks to determine an assignment $v_0 \in \{0, 1\}^n$ that satisfies the Boolean formula in conjunctive normal form (CNF) $C_1 \wedge C_2 \wedge ... \wedge C_m$ or to derive its infeasibility.

A weighted formula is a pair $WF = \{CF, W\}$ where $CF = (C_i)_{i \le m}$ is a clause form and $W = (w_i)_{i \le m} \in \mathrm{N}^m$ is an integer vector; for each $i \le m$, w_i is the weight of the clause C_i. An assignment $v \in \{0, 1\}^n$ determines a weight value in the weighted formula WF as:

$$wc(CF, W, v) = \sum_{c_i(v)=1} w_i \qquad (1)$$

The MAXSAT problem asks to determine an assignment $v_0 \in \{0, 1\}^n$ that maximizes the sum of the weights of satisfied clauses:

$$wc(CF, W, v_0) = Max\{wc(CF, w, v) \mid v \in \{0, 1\}^n\} \qquad (2)$$

MAXkSAT is the subset of MAXSAT instances in which each clause has exactly k literals. If each weight is equal to one, we call the problem unweighted MAXSAT, otherwise we speak of weighted MAXSAT or simply MAXSAT. Note that SAT is a special case of the MAXSAT in which all clauses have unit weight and one wants to decide if there is any truth assignment of total weight m.

MAXSAT algorithms can be classified as complete and incomplete depending on whether they can find the optimal assignment. Complete algorithms can determine optimal assignments but they are computationally expensive and are not appropriate in solving large instances. These include Davis-Putnam's procedure [10], and various heuristics using different branching rules and reinforcement techniques like equivalence reasoning [20] or backbone detecting [11].

Incomplete methods are usually faster and can solve some large problems of thousands of variables that complete methods cannot handle. In this outline, stochastic local search has become an important general purpose method for solving satisfiability problems: It starts with a random initial solution and tries to improve it by moving to neighbouring solutions. It can be trapped in local poor minima or plateaus and it requires, therefore, a strategy to both escape from these local minima and to guide the search toward good solutions. Its performance lies in the choice of the initial solution and the transformation applied to the current solution. Many stochastic local search methods have been designed such as simulated annealing [32], TS [21], Greedy Randomized Adaptive Search Procedure (GRASP) [26], Discrete Lagrangian based search Method (DLM) [31] and Reactive search [3]. Although the list of competitive heuristics is not exhaustive, the most accepted one is Greedy SATisfiability (GSAT) defined initially for SAT [18, 27, 28, 30] and applied next to MAXSAT. It begins with a randomly generated initial assignment and at each iteration, it flips the variable that had the largest decrease in unsatisfied clauses. It accepts also the moves which either produce the same objective function value or increase it. This process is repeated until a maximum number of non-improving moves is reached. Different *noise* strategies to escape from local optima are added to GSAT like WSAT [29], Novelty and R-Novelty [22], and UnitWalk [15]. The WalkSAT (WSAT) procedure [29] is a particularly powerful variant of GSAT. It mainly consists of a random walk on the unsatisfied clauses. The experimental results obtained by Jiang *et al.* [16] with an adaptered version of WSAT on MAXSAT encodings of the Steiner tree problem showed that this approach is competitive with the best current specialized Steiner tree algorithm.

In the TS method [13, 14], a memory of forbidden moves called *tabu list* is used to avoid the search process revisiting the previously found solutions. At each iteration, a configuration once visited is made tabu for a given number of iterations. Different history-based heuristics have been proposed to intensify and diversify the search into previously unexplored regions of the search space with information collected from the previous phase. HSAT [12] introduces a tie-breaking rule into GSAT so that if more moves produce the same best results, the preferred move is the one that has been applied for the longest period. The results obtained on some SAT benchmark tasks present a better performance with respect to WSAT. TSAT is a Tabu search procedure for SAT problems [21] that has

also been shown to be competitive with WSAT on hard random 3-SAT instances.

simulated annealing is a method inspired by natural systems [19]. It emulates the behaviour of frustrated physical systems in thermal equilibrium: A state of minimum energy may be attained by cooling the system slowly according to a temperature schedule. The simulated annealing local search method moves through the space configurations according to the Metropolis algorithm [24] driving the system to equilibrium dynamics. Experimental comparison between GSAT and simulated annealing [32] on hard satisfiability problems indicated that simulated annealing satisfied at least as many clauses as GSAT, however, simulated annealing requires careful tuning of *temperature schedule* parameters. Selman *et al.* [27] affirm that they were unable to find a cooling schedule that outperformed GSAT.

3. Extremal Optimization Method

The concept of *Self-Organized Criticality* (SOC) was introduced by Bak *et al.* in 1987 [2] to describe some open, dissipative, spatially extended systems such as biological evolution, earthquakes and solar flares, that spontaneously achieve a critical state characterised by power-law distribution of event sizes [33]. The Bak-Sneppen model [1] is a SOC model which was proposed to describe the self-organization phenomenon in the biological evolution of species. In this model, species have an associated value between 0 and 1 called *fitness* and a selection process against the extremely bad ones is applied. At each iteration, species having the smallest fitness value is replaced by a random value which impacts obviously the fitness of interconnected species. After a sufficient number of steps, the system reaches a highly correlated state SOC in which all species have reached a fitness of optimal adaptation.

This coevolutionary process has been converted into an optimization algorithm called Extremal Optimization (EO) and introduced by Boettcher and Percus [5]. The basic structure of the algorithm EO proceeds as follows. Consider a system described by a set of N couples of variables x_i each with an associated fitness value (or individual cost) λ_i. [5]:

1. Choose at random an initial state of the system.
2. Rank each variable x_i of the system according to its fitness value λ_i.
3. Update the variable with the smallest fitness λ_i. according to some move class.
4. Repeat (2) a preset number of times.

It consists of updating extremely undesirable variables of a sub-optimal solution, replacing them by random values to ensure efficient exploring of many local minima. The rank ordering allows EO to maintain well-adapted pieces of a solution, while updating an unfit variable gives the system enough flexibility to explore various space configurations. In addition, EO gives no consideration to the moves outcome. A general modification of EO [5, 6], noted τ-EO, consists of ranking all variables from rank $n = 1$ for the worst fitness to rank $n = N$ for the best fitness λ. For a given value of τ, a power-law probability distribution over the rank order is considered:

$$P(n) \propto n^{-\tau} \quad (1 \leq n \leq N) \tag{3}$$

At each update, select a rank k according to $P(k)$ and change the variable x_k state. The worst variable (with rank 1) will be chosen most frequently, while the higher ones will sometimes be updated. In this way, a bias against worst variables is maintained and no rank gets completely excluded from the selection process. However, the search performance depends on the parameter τ. For $\tau = 0$, the algorithm is simply a random walk through the search space. While for too large values of τ the process approaches a deterministic local search where only a small number of variables with particularly bad fitness would be chosen at each iteration. The optimal value of τ is placed at "*...a point between having τ large enough to descent into local minima while having τ just small enough to not get trapped inside the basin of any local minimum*" [4]. Boettcher and Percus [4, 7] have established a relation between τ, the run time t_{max} and the number of the variables of the system N to estimate the optimal value of τ. Let $t = AN$, where A is a constant, then:

$$\tau \sim 1 + \frac{\ln(A/\ln(N))}{\ln(N)} \quad (N \to \infty,\ 1 << A << N) \tag{4}$$

At this optimal value, the best fitness variables are not completely excluded from the selection process and hence, more space configurations can be reached so that greatest performance can be obtained. It may appear that the strategy of EO is similar to an ineffective random search, but in fact, by persistent selection against the worst fitness, the algorithm returns frequently to near-optimal configurations.

A disadvantage of EO remains in the haziness of variable individual fitness definition. Also, in highly connected systems, EO is slowed down significantly by the fitness re-evaluating process at each iteration. However, these disadvantages could be ineffective in the case of problems that have a natural choice of fitness function with a low variable connectivity like satisfiability problems.

EO complements approximation methods inspired by equilibrium statistical physics like simulated annealing [19]. But when EO fluctuates to take the system far from the equilibrium state [5, 6], simulated annealing drives the system to equilibrium dynamics. The experiment results presented by Boettcher and Percus [6, 7] compare EO with simulated annealing on

benchmarks of the Graph Partitioning problem (GP) and the Travelling Salesman Problem (TSP). For the GP problem, EO appears to be more successful over a large range of graphs while simulated annealing is useful for highly connected graphs. In the TSP Euclidian case, EO results trail those of simulated annealing. But, in the non-Euclidian TSP case, EO results outperform simulated annealing ones.

4. Extremal Optimization for MAXSAT Problem

Given a MAXSAT problem instance of n Boolean variables and m weighted clauses, the definition of the fitness λ_i of a variable x_i depends on its relation with the other variables [4]. In a previous study [23] we have defined λ_i as the sum of weights of clauses satisfied by the variable x_i over the total weights sum. The results obtained with such a definition were not very satisfying because a lowly connected variable will be too fit to be updated even if it satisfies all of its weights. For example, given an unweighted MAXSAT problem of 100 clauses (all the weights are equal to 1) and 20 variables, a variable that appears in 5 clauses and satisfies all of them, will have a fitness of 5/100, while another variable which appears in all the clauses and satisfies only 10 clauses, will have a fitness of 10/100 and be considered as better than the previous variable. Hence, the algorithm will continue to update a variable that has satisfied all of its clauses. This waste of search time, explains the low quality solutions obtained with such definition. In this new implementation, we consider a more effective definition of a variable fitness as the negation of the sum of weights of clauses unsatisfied by x_i:

$$\lambda_i = -\sum_{x_i \in C_j \text{ and } C_j(v)=0} w_j \qquad (5)$$

The cost function $C(S)$ is the individual cost contributions λ_i for each variable x_i. $C(S)$ has to be minimized in order to maximize the number of satisfying clauses. Then, the following equation holds:

$$C(S) = -\sum_{i=1}^{n} \lambda_i \qquad (6)$$

Procedure: EO-MAXSAT(*WeightedClauses*, *MaxSteps*, *Tho*)
1. *S* := a random truth assignment over the variables that appear in clauses.
2. *Sbest* := *S*.
3. *UnsatClauses* := clauses not satisfied by *S*.
4. *WeightUnsat* := sum of *UnsatClauses* weights.
5. *For* k := 1 *to MaxSteps do*
   a. if S satisfies WeightedClauses then return (S)
   b. Evaluate λ_i for each x_i in *WeightedClauses* according to equation (5)
   c. Rank x_i according to λ_i from the worst to the best
   d. Select a rank *j* according to $P(j) \propto j^{-Tho}$
   e. S' := S in which xj value is flipped
   f. *If C (S) < C (Sbest) then Sbest* := *S'*
   g. *S* := *S'*
   h. Update (*UnsatClauses*, *WeightUnsat*)

   Endfor
6. *Return* (*Sbest*, C(*Sbest*), *UnsatClauses*, *WeightUnsat*)

End {EO-MAXSAT}

Figure 1. General structure of the procedure EO for MAXSAT.

The pseudo code of the procedure EO-MAXSAT (EO for MAXSAT) is described in Figure 1.

1. Lines 1-4: The search begins at a random truth assignment *S* and the current best assignment, *Sbest*, is set to *S*. Further, the initial set of unsatisfied clauses and their total weights are calculated.
2. Line 5: A fixed number of tries *MaxSteps* is executed. Each step in the search corresponds to flipping the truth value assigned to a variable according to EO strategy.
3. Lines b-c: The variable individual fitnesses are evaluated and sorted using a hash table to accelerate the EO process.
4. Lines d-e: A variable is selected according to the power-law distribution (equation (3)) and its value is flipped.
5. Lines f-h: *Sbest* is related to the current assignment with the minimum cost (equation (6)). The current assignment, the set of unsatisfied clauses and their total weights are then maintained.
6. Line 6: The current best assignment is returned when no model can be found.

5. Experimental Results

We now present experimental results obtained with a prototype of the procedure EO-MAXSAT. We compare it to WSAT, simulated annealing, and TS. The code of WSAT was taken from the web site http://www.cs.cornell.edu/home/selman/sat/sat-package.tar. A version of TS has been developed, while for simulated annealing we refer to experimental results presented in [29]. The algorithms were coded in C language and run on a computer (Pentium III, 512 Mb RAM, 450 MHz clock).

The test suite is dedicated to randomly generated MAX3SAT and MAX4SAT instances mostly unsatisfied [25]. They are simply generated and do not hold any hidden structure inherent to a specific problem. The generator of such instances is available at the URL given above. For random MAX3SAT the instances considered are (100, 200), (100, 500), (100, 700), (300, 600), (300, 800), (300, 1500), (300, 2000) and (500, 5000) where each couple (n, m) means n

variables and m clauses. For random MAX4SAT, the instances considered are: (100, 700), (300, 1500), (300, 2500) and (300, 3000). For each couple (n, m) 50 instances were randomly generated. In the following, we refer to MAX3SAT and MAX4SAT instances with U3-n: m and U4-n: m respectively.

The first experiments involve the investigation of the optimal value of τ. The algorithm was run 10 times for each instance and the parameter *MaxSteps* was fixed to 100n. Figure 2 presents the average number of unsatisfied clauses for each instance while varying τ between 1.1 and 2. Results show that optimal performance had been observed for τ ranging from 1.4 to 1.6 for all instances. For U3 instances, τ is equal to 1.6 for n = 100 (Figure 2-a) and equal to 1.5 for n = 300 or n = 500 (Figures 2-b and 2-c). While for U4 instances, τ is equal to 1.5 for n = 100 and between 1.4 and 1.5 for n = 300 (Figures 2-d and 2-e). These numerical values are relatively close to those predicted by equation (4). (τ = 1.6683 for n = 100, τ = 1.5021 for n = 300 and τ = 1.4470 for n = 500). We note that equation (4) cannot be applied for long run times (A > n).

Subsequently, we examined the effectiveness of the procedure EO-MAXSAT on the benchmark instances where τ was set to the default value of 1.6 for n = 100 and to 1.5 for n = 300 or n = 500. Table 1 shows the average, minimum and maximum number of unsatisfied clauses for each instance obtained over 10 runs and the maximum CPU time allowed for each run. Satisfying assignments were found for 5 out of 6 known satisfiable problems (U3-100:200, U3-300:600, U3-300:800, U4-300:1500 and U4-300:2500). The good robustness of the algorithm could be suggested by the empirical values obtained for the standard deviation.

Table 1. Average results over 10 runs achieved by EO-MAXSAT on MAX3SAT (U3-n:m) and MAX4SAT (U4-n:m) instances.

Problem Identifier	Avg	Min	Max	Std. Dev.	CPU secs
U3-100:200	0	0	0	0	20.0
U3-100:500	3.10	3.00	4.60	0.89	20.0
U3-100:700	12.80	12.60	13.72	0.59	20.0
U3-300:600	0	0	0	0	60.0
U3-300:800	0	0	0	0	60.0
U3-300:1500	8.00	7.90	9.40	0.83	60.0
U3-300:2000	30.40	30.20	32.60	1.33	60.0
U3-500:5000	163.20	162.50	173,00	5,87	100.0
U4-100:700	0.022	0.020	0.0235	0.0015	20.0
U4-300:1500	0	0	0	0	60.0
U4-300:2500	0	0	0	0	60.0
U4-300:3000	4.20	4.09	4.58	0.25	60.0

Table 2 represents the average number of unsatisfied clauses for U3 and U4 instances. The column in the table denoted with simulated annealing is derived from [29]. It represents reported results obtained with simulated annealing on the same instances. The Additional columns refer to our experiments on EO-MAXSAT, TS and WSAT where the maximum allowed run-times are the same as reported in Table 1. The tests show that EO-MAXSAT improves significantly upon the results obtained with simulated annealing and TS on all the benchmark instances. EO-MAXSAT has been able to satisfy 5 out of 6 satisfiable problems, while WSAT has found models for all these problems. However, EO-MAXSAT has outperformed WSAT on 5 out of 6 remaining unsatisfiable problems.

Table 2. Average number of unsatisfied clauses for the benchmark instances. The data for simulated annealing are derived from [29].

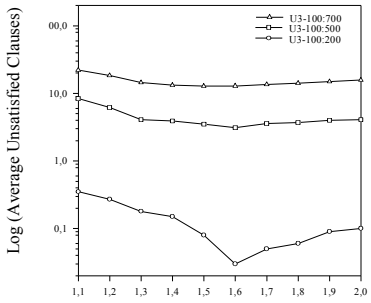
Problem Identifier	EO-MAXSAT	Simulated Annealing	TS	WSAT (p = 0.5)
U3-100:200	0	0.2	0.24	0
U3-100:500	3.1	8.2	4.7	2.76
U3-100:700	12.8	18.1	14.4	13.4
U3-300:600	0	2.7	2.3	0
U3-300:800	0	6.1	4.3	0
U3-300:1500	8.0	30.0	13.2	8.12
U3-300:2000	30.4	58.0	38.4	32.28
U3-500:5000	163.2	226.4	174.2	167.0
U4-100:700	0.022	0.3	0	0
U4-300:1500	0	1.0	0	0
U4-300:2500	0	11.9	3.4	0
U4-300:3000	4.2	14.3	5.9	4.71

Next, we compare the variation of the average number of unsatisfied clauses as a function of the number of iterations for EO-MAXSAT and WSAT. Figure 3 shows results obtained on two hard instances U3-300:2000 and U3-500:5000. It is easy to observe that EO-MAXSAT converges faster to a local optimum than WSAT. For example, on U3-300: 2000, assignments containing in average 44 violated clauses, were obtained at the 400th iteration for EO-MAXSAT but only after the 1000th iteration for WSAT.

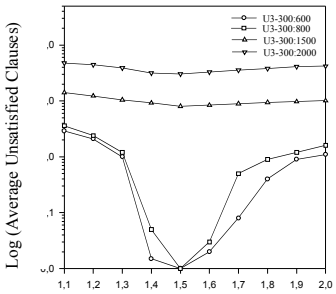
On the instance U3-500:5000, EO-MAXSAT was able to reduce the number of violated clauses to 175 after 1000 iterations, while WSAT reached this local optimum after 10000 iterations.

As a result, the average performance of EO-MAXSAT exceeds that of simulated annealing, TS and WSAT on the tested benchmarks. Its high quality solution can be explained as follows. On the one hand, the large fluctuations of EO allow the search to escape quickly from local minima and to explore other assignments in the search space, and on the other hand, the extremal selection process obliges the search to frequently visit near-optimal assignments. Comparable to WSAT, simulated annealing and TS, the procedure EO-MAXSAT is incomplete, which means that there is no guarantee that it can find a model to the problem if one exists. However, when a solution is found, we are sure that it is correct, which means that the procedure is sound. EO-MAXSAT is constructed around a local

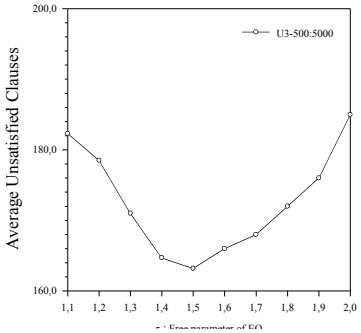
search scheme like WSAT family algorithms. The main difference between EO and WSAT, simulated annealing and TS, resides in the fact that EO makes moves using a fitness that is based not on anticipated outcome but purely on the current state of each variable. Also, EO needs only one tuning parameter, while simulated annealing requires careful tuning *temperature schedule* [19] parameters.



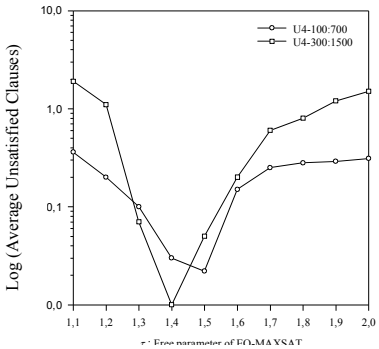
(a)



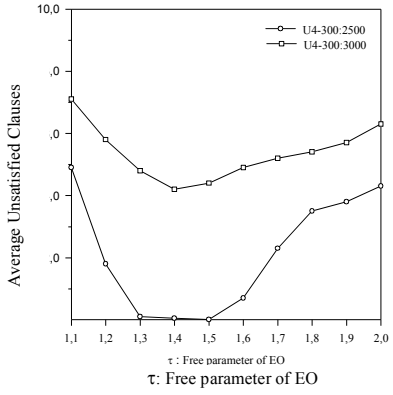
(b)



(c)



(d)



(e)

Figure 2. The effect of the parameter τ on the average number of unsatisfied clauses.

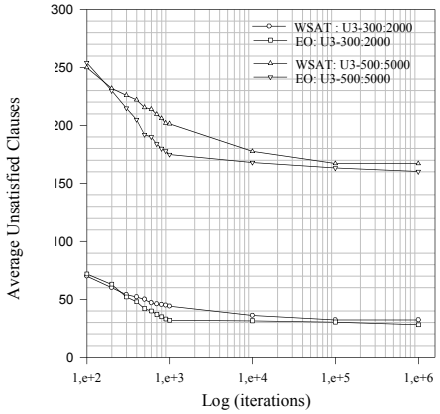


Figure 3. Average number of unsatisfied clauses as a function of the number of iterations for EO-MAXSAT and WSAT.

6. Conclusion

Extremal optimization is a surprisingly simple and powerful method to find high quality solutions to hard optimization problems. It belongs to the family of stochastic local search algorithms such as WSAT, simulated annealing and TS. Its straightforward implementation and test provide motivation to adapt it for solving various classes of these problems. In this paper, we have shown how to deal with this method to solve the MAXSAT problem, one of the most important problems in the encoding-resolution paradigm. The derived procedure EO-MAXSAT is sound but incomplete. It has been tested on random MAX3SAT and MAX4SAT problems. The experimental results demonstrate its superiority with respect to simulated annealing, TS and even to WSAT on the considered instances. The high performance achieved is due to the flexibility maintained by the EO process to explore more space configurations and to retain well adapted pieces of a solution. Its fast convergence to a near optimal solution may be useful in practice for especially large instances. Actually we are performing additional tests on other SAT and MAXSAT instances from the DIMACS archive at the

URL http://dimacs.rutgers.edu /challenges/ and comparisons with other local search methods like DLM.

Acknowledgments

We are particularly indebted to Stefan Boettcher at Emory University for his valuable comments on our study. We thank Prof. Selman B. at Cornell University who made available his MAXSAT random generator software. The comments of the anonymous referees are gratefully acknowledged.

References

[1] Bak P. and Sneppen K., "Punctuated Equilibrium and Criticality in a Simple Model of Evolution," *Physical Review Letters*, vol. 71, pp. 4083-4086, 1993.

[2] Bak P., Tang C., and Wiesenfeld K., "Self-Organized Criticality: An Explanation of 1/f-Noise," *Physical Review Letters*, vol. 86 no. 23, pp. 5211-5214, 1987.

[3] Battiti R. and Protasi M., "Reactive Search: A History-Sensitive Heuristic for MAX-SAT," *ACM Journal of Experimental Algorithmics*, vol. 2, 1997.

[4] Boettcher S. and Grigni M., "Jamming Model for the Extremal Optimization Heuristic," *Journal of Physics A: Mathematical and General*, vol. 35, pp. 1109-1123, 2002.

[5] Boettcher S. and Percus A. G., "Nature's Way of Optimizing," *Elsevier Science, Artificial Intelligence*, vol. 119, pp. 275-286, 2000.

[6] Boettcher S. and Percus A. G., "Optimization with Extremal Dynamics," *Physical Review Letters*, vol. 86, no. 23, pp. 5211-5214, 2001.

[7] Boettcher S. and Percus A. G., "Extremal Optimization for Graph Partitioning," *Physical Review E*, vol. 64, pp. 1-13, 2001.

[8] Cook S. A., "The Complexity of Theorem Proving Procedures," *in Proceedings of the 3rd Annual ACM Symposium of the Theory of Computation*, pp. 151-158, 1971.

[9] Crescenzi P. and Kann V., "How to Find the Best Approximation Results: A Follow-up to Garey and Johnson," *ACM SIGACT News*, vol. 29 no. 4, pp. 90-97, 1998.

[10] Davis M. and Putnam M., "A Computing Procedure for Quantification Theory," *Journal of the ACM,* vol. 7. pp. 201-215, 1960.

[11] Dubois O. and Dequen G., "A Backbone-Search Heuristic for Efficient Solving of Hard 3-SAT Formulae," *in Proceedings of the IJCAI'01*, pp. 248-253, San Francisco, CA, 2001.

[12] Gent I. and Walsh T., "Unsatisfied Variables in Local Search," in Hybrid Problems, Hybrid Solutions, Hallam J. (Eds), IOS Press, Amsterdam, pp. 73-85, 1995.

[13] Glover F., "Tabu Search: Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190-206, 1989.

[14] Glover F., "Tabu Search: Part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. +32, 1989.

[15] Hirsch E. A. and Kojevnikov A., "UnitWalk: A New SAT Solver That Uses Local Search Guided by Unit Clause Elimination," in *Proceedings of the 5th International Symposium on the Theory and Applications of Satisfiability Testing (SAT 2002)*, Cincinnatti, Ohio, USA, pp. 35-42, 2002.

[16] Jiang Y., Kautz H. A., and Selman B., "Solving Problems With Hard and Soft Constraints Using a Stochastic Algorithm for MAX-SAT," in *Proceedings of the 1st International Joint Workshop on Artificial Intelligence and Operations Research*, 1995.

[17] Johnson D., "Approximation Algorithms for Combinatorial Problems," *Journal of Computer and System Sciences*, vol. 9, pp. 256-278, 1974.

[18] Kautz H. A. and Selman B., "Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search," in *Proceedings of the AAAI'96*, vol. 2, pp. 1194-1201, MIT Press, 1996.

[19] Kirkpatrick S., Gelatt C. D., and Vecchi P. M., "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[20] Li C.M., "Integrating Equivalence Reasoning into Davis-Putnam Procedure," *in Proceedings of the AAAI'00*, pp. 291-296, 2000.

[21] Mazure B., Sais L., and Gregoire E., "Tabu Search for SAT," *in Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference*, pp. 281-285, 1997.

[22] Mc Allester D., Selman B., and Kautz H., "Evidence for Invariants in Local Search," *in Proceedings of the IJCAI-97*, 1997.

[23] Menaï M. B. and Batouche M., "EO for MAXSAT," *in Proceedings of the International Conference on Artificial Intelligence (IC-AI'02)*, Las Vegas, USA, vol. 3, pp. 954-958, 2002.

[24] Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., and Teller E., "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, vol. 21, pp 1087-1092, 1953.

[25] Mitchell D., Selman B., and Levesque H. J., "Hard and Easy Distributions of SAT Problems," *in Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI, San Jose, CA, pp. 459-465, 1992.

[26] Resende M. G. C., Pitsoulis L. S., and Pardalos P. M., "Approximate Solution of Weighted MAX-SAT Problems Using GRASP," *DIMACS Series on Discrete Mathematics and Theoretical*

Computer Science, American Mathematical Society, vol. 35, pp. 393-405, 1997.

[27] Selman B. and Kautz H. A., "An Empirical Study of Greedy Local Search for Satisfiability Testing," *in Proceedings of the 11th National Conference on Artificial Intelligence*, pp. 46-51, 1993.

[28] Selman B. and Kautz H. A., "Domain Independent Extensions to GSAT: Solving Large Structured Satisfiability Problems," *in Proceedings of the IJCAI-93*, pp. 290-295, 1993.

[29] Selman B., Kautz H. A., and Cohen B., "Noise Strategies for Improving Local Search," in *Proceedings of the 12th National Conference on Artificial Intelligence*, pp. 337-343, 1994.

[30] Selman B., Levesque H., and Mitchell D., "A New Method for Solving Hard Satisfiability Problems," *in Proceedings of the 10th National Conference on Artificial Intelligence*, pp. 440-446, 1992.

[31] Shang Y. and Wah B. W., "A Discrete Lagrangian-Based Global-Search Method for Solving Satisfiability Problems," *Journal of Global Optimization*, vol. 12, pp. 61-99, 1998.

[32] Spears W. M., "Simulated Annealing for Hard Satisfiability Problems," in Johnson D. S. and Trick M. A. (Eds), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* American Mathematical Society, vol. 26, pp. 553-558, 1996.

[33] Vieira M. S., "Are Avalanches in Sandpiles a Chaotic Phenomenon?," Research Report, 2004.

[34] Yannakakis M., "On the Approximation of Maximum Satisfiability," *Journal of Algorithms*, vol. 17, pp. 475-502, 1994.

Mohamed El Bachir Menai is currently a researcher at the AI Laboratory of the University of Paris 8, France. From 1988 to 2003, he was a researcher and an assistant professor at the Computer Science Department of the University of Tebessa. He received his BSc in computing systems from the University of Annaba in 1994. His main interests are in the areas of problem complexity, heuristic search, evolutionary computation, and quantum computing.

Mohamed Batouche received his MSc and PhD degrees in computer science from the Institut National Polytechnique de Lorraine (INPL), France, in 1989 and 1993, respectively. Currently, he is a full-time professor at the University Mentouri of Constantine, Algeria. His research areas include artificial intelligence and pattern recognition.