2nd Hellenic Conf. on AI, SETN-2002, 11-12 April 2002, Thessaloniki, Greece,
Proceedings, Companion Volume, pp. 129-140

Heuristic Backbone Sampling for Maximum Satisfiability

Orestis Telelis and Panagiotis Stamatopoulos

Department of Informatics and Telecommunications
University of Athens
Panepistimiopolis, 157 84 Athens, Greece
{telelis,takis}@di.uoa.gr

Abstract. The weighted maximum satisfiability (MAXSAT) problem is of great interest to the Artificial Intelligence community, as a model for several constraint satisfaction problems (emerging e.g. from planning) which require that an optimum subset of their constraints be satisfied. Recent research on satisfiability (SAT) problems has reached interesting conclusions regarding their hardness. In this paper, we introduce an algorithm designed in a way inspired by these results. Based on the newly introduced concept of the backbone of a formula in conjunctive normal form, we try to sample the most likely values of boolean variables through an iterative process. Experiments conducted on appropriate unsatisfiable SAT instances show that the algorithm converges to a near optimum subset of satisfied disjunctive clauses. Evidence of remarkable success on weighted MAXSAT instances is also presented and discussed.

1 Introduction

The boolean satisfiability problem (SAT) has attracted the interest of the AI community, as an aspect of fundamental importance for automated reasoning systems. SAT was also the first combinatorial problem shown to be NP-complete [1].

The problem involves determining a satisfying assignment on boolean variables that participate in the formation of a boolean formula in *conjunctive normal form (CNF)*. A CNF formula is a conjunction of clauses. Each clause consists of the disjunction of literals, where a literal is a boolean variable or its logical negation.

The optimization version of SAT (MAXSAT) is the problem of finding an assignment over the variables of a CNF formula that maximizes the total number of satisfied clauses. In this paper, we also consider the more generic *weighted* MAXSAT problem, where each clause is associated with a positive weight. The objective function of weighted MAXSAT maps an assignment of the boolean variables to the sum of the weights of the satisfied clauses. The aim is to maximize the aforementioned sum. Setting all weights to 1 yields the unweighted form of MAXSAT. For the rest of our discussion, MAXSAT will refer to the generic weighted form, unless otherwise stated.

MAXSAT constitutes a special case of the generic *valued Constraint Satisfaction Problem (valued-CSP)* [11] and, as such, it may be used to model several overconstrained problems, which require that an optimal subset of their constraints be satisfied. As an example, we mention the *Steiner Tree* problem faced in [4]. Furthermore, several planning problems can be transformed into boolean CNF formulae, which require either complete satisfaction of all clauses, or satisfaction of an appropriate subset, according to some objective optimality criterion.

We introduce *HBS* (Heuristic Backbone Sampling), an iterative heuristic for MAXSAT problems. The main contribution of the proposed methodology concerns a stochastic initialization scheme, which provides a simple hill climbing heuristic with potentially interesting startup states. Our work was inspired by recent studies on the hardness of SAT problems, which have revealed the property of the *backbone* [7] for CNF formulae. We conjecture that it may be possible to measure the likelihood of a variable being assigned to a particular value in several good assignments, given a set of good assignments. We then exploit this measure in producing stochastically a new assignment, which is the startup state of a hill climbing procedure.

Several theoretical and experimental MAXSAT studies have appeared. Impressive approximation algorithms have been introduced, that acquire assignments of quality 75% of the optimum [15] and beyond [3]. In [4], a modification of the WalkSat [12] algorithm is presented, which deals with MAXSAT problems. In [10], a constructive procedure provides startup assignments for a hill climbing heuristic (GRASP). Their combination finds near optimal solutions on many MAXSAT instances. Another algorithm which performs remarkably better on the same instances appears in [14].

The paper is organized as follows: In section 2, we briefly survey some issues concerning the backbone structure property. The proposed algorithm is discussed in section 3. Experimental results and conclusions follow in sections 4 and 5.

2 Phase Transition and the Backbone

Recent research in SAT problems has provided several statistical and theoretical results, concerning the hardness of satisfying CNF formulae. Early experimental results [6] have shown that randomly generated 3-SAT instances (each clause contains exactly 3 literals) of M clauses and N variables with the property $\alpha = M/N \simeq 4.26 = \alpha_c$ are hard to solve. Furthermore, instances with $a < \alpha_c$ or $a > \alpha_c$ seem to be relatively easy to solve: their search space is either dense in satisfying assignments, or empty, respectively, thus making it easy to prove their satisfiability or unsatisfiability. This easy-hard-easy effect is characterized as the phase transition of SAT.

The statistical and theoretical study of phase transitions is intended to reveal the increase in complexity for the various distributions of SAT problems. In [8], through the application of methods from statistical mechanics and extended ex-

perimentations, the phase transition of K-SAT is investigated for the estimation of the complexity increase rate with problem size.

An important structural property of unweighted CNF formulae, namely the *backbone*, has been revealed through the study of phase transitions [7, 9]. The backbone stands for the set of variables which appear constrained to the same value in all optimal variable assignments. As shown experimentally in [9], the backbone size is an important parameter for the cost of local search procedures. A large backbone keeps most of the variables of the formula frozen to some value in every optimal assignment, thus implying that all optimal assignments will lie in a restricted area of the search space. Small backbones, on the contrary, tend to preserve a wider distribution of optimal assignments. Since in a large backbone, participating in the optimum assignment, many variables have a restricted value, there are many erroneous decisions (at least as many as the restricted variables) to be taken during search. Backbones of considerable size seem to emerge in CNF formulae lying on the phase transition and beyond ($a \geq a_c$). Occurrence of backbones in optimization problems is also discussed in [13].

Optimal and near optimal assignments are expected to include at least a subset of the formula's backbone constrained to appropriate values. The core of *HBS* involves maintaining a set of the best assignments found so far. This set is used to determine the likelihood of a variable being assigned to 1. We expect that, at least for the variables of the backbone, this likelihood measure will eventually converge to some very small (near 0) or very large (near 1) value. A recent systematic search algorithm, which exploits the backbone is described in [2]. In this work a constructive search procedure is described, enhanced with analytic techniques for exploiting the backbone concept, towards achieving satisfying assignments or deciding the unsatisfiability of 3-SAT CNF formulae. *HBS* is, to our knowledge, the first local search strategy, designed to capture the backbone of (unweighted) MAXSAT problems, in a statistical manner, for guiding the search towards optimal assignments.

3 The *HBS* Algorithm

In this section, we describe the *HBS* algorithm. *HBS* is an iterative algorithm. In each iteration, a stochastic procedure produces a new assignment, which is further optimized by a hill climbing heuristic. The stochastic procedure is examined first and a short description of the hill climbing heuristic follows. In the following paragraphs, we consider a CNF formula built upon n boolean variables, $x_i, i = 1 \ldots n$. If a is an assignment, then we denote the value of variable x_i in a with $a(x_i)$. The objective function value corresponding to a is denoted with $Z(a)$.

3.1 The Stochastic Initialization Procedure

The stochastic initialization procedure is memory-based. A set $\mathcal{S}$ of restricted size contains the best assignments found during previous iterations of *HBS*. $\mathcal{S}$ is an input to the procedure.

A new startup assignment is produced by assigning x_i the boolean value 1 with probability:

$$p_i = \Big(\sum_{a \in \mathcal{S}} f_a a(x_i)\Big) / \Big(\sum_{a \in \mathcal{S}} f_a\Big)$$

If we set $f_a = 1$, then p_i is equal to the frequency of *positive* appearances of the variable x_i in the set $\mathcal{S}$. Thus, p_i intuitively dictates the *most likely* value assignment of x_i with respect to the assignments contained in $\mathcal{S}$. An alternative way of obtaining a more representative p_i value is setting $f_a = Z(a)$. In this way, we also assign a measure of importance to x_i's value in the various assignments of $\mathcal{S}$. We adopted the latter approach during our experimentations.

HBS $(t, I, |\mathcal{S}|)$

1. Initialize $\mathcal{S}$ with random $(P[x_i = 1] = 0.5)$ assignments

2. repeat I times

   a. Calculate $p_i, i = 1 \ldots n$

   b. Pick the best assignment among t randomly
      created assignments, using the p_i probabilities.

   c. Do hill climbing until local optimum, and store in $\mathcal{T}$
      all the evaluated assignments during this iteration.

   d. Insert in $\mathcal{S}$ the best assignment in $\mathcal{T}$
      not already in $\mathcal{S}$ and better than the worst of $\mathcal{S}$.
      Delete the worst in $\mathcal{S}$.

Fig. 1. The complete *HBS* algorithm

3.2 Hill Climbing

The neighbourhood explored by the heuristic is the standard *flip* neighbourhood for SAT problems. A transition from one assignment to a neighbouring one is performed by flipping a selected variable (i.e. setting it to its complementary value). Let $C^+(x_i)$ and $C^-(x_i)$ be the sets of clauses that become satisfied and unsatisfied respectively by flipping the variable x_i. The gain obtained by flipping x_i is then defined as:

$$g_i = \sum_{c_j \in C^+(x_i)} w_j - \sum_{c_j \in C^-(x_i)} w_j$$

The *steepest ascent* version of hill climbing performs in each iteration a calculation of the $\boldsymbol{g}$ vector and flips the variable x_i with $i = \arg\max_j(g_j > 0)$. If $(\forall j)(g_j \leq 0)$, then a local optimum has been reached and the search stops. The complete *HBS* algorithm appears in Fig. 1.

3.3 Implementation Details

We clarify here some implementation issues, not directly discussed in previous paragraphs, but imposed by the description in Fig. 1.

The stochastic initialization scheme is repeated t times, as shown in the figure. The t value is a parameter to the algorithm, which tunes the probability of finding randomly a startup assignment of high quality. During our experimentations, we determined the value t in combination with a *clipping* policy for the p_i values: all p_i values outside the range $[0.1, 0.9]$ were appropriately clipped to the margins of this range. We then experimented with values of $t \geq 10$, so that each boolean variable could obtain randomly one of the two values with probability at least 10%. The bias of producing assignments extremely similar to the ones that appear in the set $\mathcal{S}$ is thus reduced.

$\mathcal{T}$ stores candidate solutions for updating $\mathcal{S}$. Updating $\mathcal{S}$ in the end of each iteration means inserting the best assignment $a \in \mathcal{T}$ for which $Z(a) \neq Z(s)$, $\forall s \in \mathcal{S}$ holds. Furthermore, the size of $\mathcal{S}$ is maintained constant during the execution of the algorithm: each time a new assignment enters the set, an assignment of worst quality is erased. It appears plausible that the convergence speed of the algorithm to high quality assignments is depended on $|\mathcal{S}|$. Finally, we should note that assignments produced during stochastic initialization which could contribute to the enrichment of $\mathcal{S}$ are also stored in $\mathcal{T}$.

4 Experimental Results

The behaviour of the algorithm was investigated through experiments carried on weighted and unweighted CNF formulae. In particular, we experimented on the `uuf125-538-100` dataset[1] of unweighted formulae, which contains 100 unsatisfiable 3-SAT instances of 125 variables and 538 clauses. All instances are "phase transition"-hard. Improved results are also discussed on the weighted MAXSAT instances of [10]. This dataset (`jnh`) contains 44 CNF formulae with clauses of varying sizes and weights uniformly distributed in the range 1–1000. These problems consist of 100 variables and 800–900 clauses. All experiments were performed on a Sun Ultra SPARC 5 workstation with 269 MHz CPU and 128 MB RAM. The run time of *HBS* for $I = 500$ iterations did not exceed 3 seconds in

[1] Available from `http://www.satlib.org`

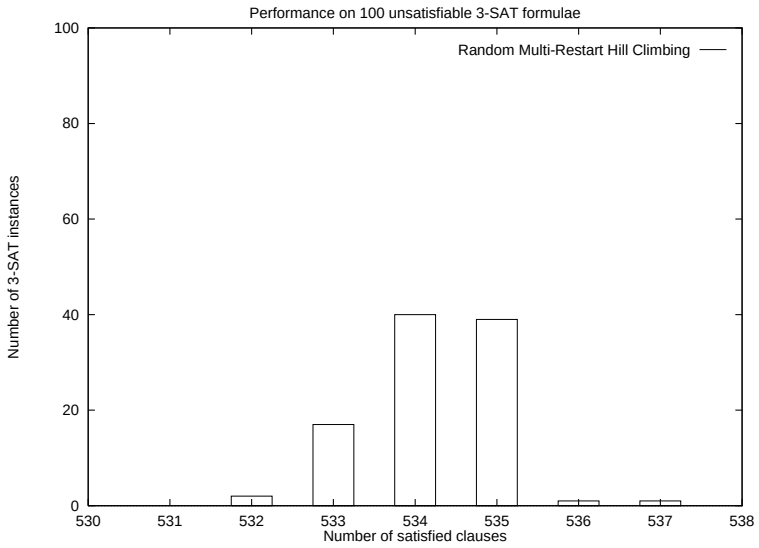

Fig. 2. Performance of MRHC on `uuf125-538-100`

our experiments. Efficiency of local search implementations has received considerable attention. For *HBS*, storing and processing of assignments constitutes an overhead to the hill climbing search part. However, it still is a polynomial time process, which we have efficiently integrated within our implementation.

4.1 On Unweighted CNF Formulae

Performance on unweighted CNF formulae is compared towards the performance of random multi-restart hill climbing (*MRHC*), where each restart is initiated with a random assignment (each variable is set to true with probability 0.5). Figures 2 and 3 summarize the results. Each of the compared heuristics was ran independently 20 times on every SAT instance. The proposed algorithm ran under the configuration $HBS(20, 300, 10)$, whereas for *MRHC*, the number of restarts was set to 300 per run.

The barcharts of Fig. 2 and Fig. 3 depict the amount of instances for which a certain number of satisfied clauses was reached. We consider the highest achieved number of satisfied clauses among twenty runs of the algorithms on each instance. We might carefully observe a shift of the performance on the core of the dataset, which corresponds to a small increase of satisfied clauses at least by two. Improvements in this range, however, are hardly achievable by random *MRHC*, since they bring most formulae to their optimum satisfiability state.

In order to fine tune the algorithm's clipping range of the p_i values, we experimented with some clipping ranges over several instances of the `uuf-125-538-100` dataset. A typical picture of the algorithm's performance is shown in Fig. 4. *HBS* quickly moves to a locally optimum area, whereas the selected clipping range af-

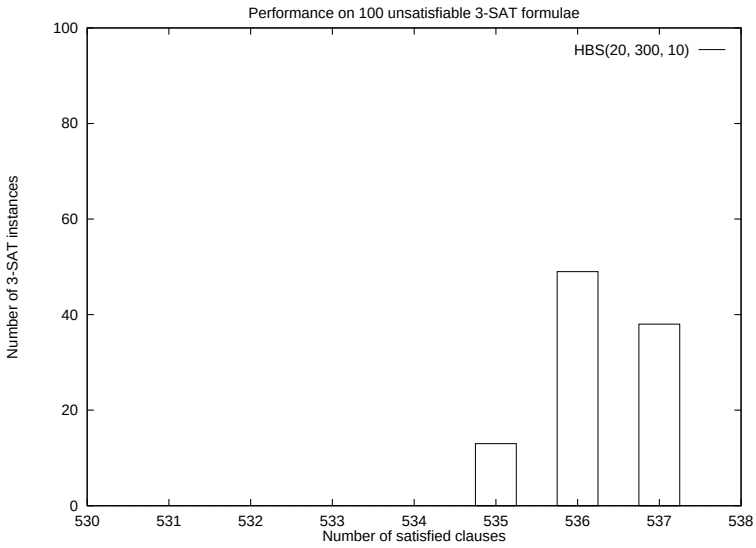

Fig. 3. Performance of *HBS* on `uuf125-538-100`

fects its performance during the subsequent iterations. In our experiments the selected $[0.1, 0.9]$ range proved to be the most appropriate. In fact, shrinking the range corresponds to approaching the *MRHC* bias of producing a new startup assignment. On the contrary, leaving the p_i values unclipped biases the algorithm towards production of startup assignments highly dependent on the ones contained in $\mathcal{S}$.

4.2 On Weighted CNF Formulae

HBS was applied on the dataset[2] of [10] with remarkable success. Twenty runs of $HBS(10, 500, 12)$ were conducted on each instance. The best solution achieved for each instance exceeded the solution quality reached by GRASP in [10]. Due to lack of space we only mention in Table 1(a) ten instances with the greatest improvement over GRASP's results. It is important to note that optimum solution was reached for 17 instances, whereas GRASP managed to solve optimally only 3 instances.

Table 1(b) depicts the best improvement achieved by *HBS* over WalkSat (WSAT) on ten instances. We experimented with the weighted MAXSAT version of WSAT, as it appears in [4]. The default parameters of WSAT were used (that is, 0.5 noise, 10000 flips), as suggested in the authors' implementation. The best improvements were calculated over 20 runs of the algorithms on each problem instance. As shown in the table, solution qualities reached by *HBS* exceeded the results obtained by WSAT. In particular, the least obtained best improvement

[2] Available from `http://www.research.att.com/~mgcr/data/index.html`

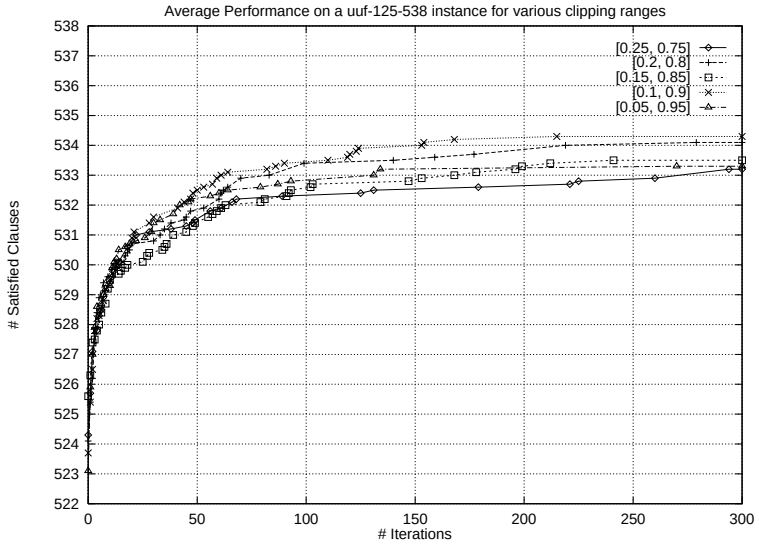

Fig. 4. Average performance over ten runs for each clipping range

over the 44 instances of the `jnh` dataset was 31. However, we should note that WSAT performed slightly better on 13 out of 44 instances, containing a minimum number of clauses.

4.3 Discussion

In [5], it is shown that randomly initialized hill climbing succeeds almost always in finding a satisfying assignment for satisfiable 3-SAT instances. Furthermore, it is shown that this success is also due to the small probability that the randomly generated initial assignment is bad (in the sense of sharing a small part in common with the satisfying assignment).

However, search spaces of MAXSAT problems are known to be seething with large amounts of local optima [14]. The existence of large backbones makes it even more difficult to construct randomly an initial assignment that captures a great part of an optimal assignment (in fact, the probability of success is exponentially small). Our experiments on the `uuf125-538-100` dataset have shown that *HBS* gradually manages, through sampling, to discover a great part of the backbone. This is confirmed by the progressive convergence of the collection $\mathcal{P} = \{p_i | i = 1 \ldots n\}$ to a state of informative certainty: many of the elements of $\mathcal{P}$ approach 0 or 1. As a result of that, the stochastically produced assignments tend to capture an even larger part of the backbone from one iteration to the next.

Reaching a state of maximum certainty for the collection $\mathcal{P}$ corresponds to maximizing the normalized sum of squares of deviations from 0.5: $D = (4/n)\sum_{i=1}^{n}(p_i - 0.5)^2$. Figure 5 depicts the maximization of D, which occurs dur-

Table 1. Performance on weighted formulae

(a)

Deviations from Optimum		
Problem	*HBS*	GRASP
jnh305	-142	-609
jnh219	0	-436
jnh8	-147	-578
jnh18	-20	-423
jnh214	-66	-462
jnh19	-79	-436
jnh308	-156	-502
jnh304	0	-319
jnh14	0	-314
jnh15	-52	-359

(b)

$HBS(10, 500, 12)$ vs. WSAT(0.5)	
Problem	Best Improvement
jnh302	564
jnh305	450
jnh303	295
jnh307	239
jnh211	194
jnh308	194
jnh216	142
jnh310	141
jnh15	129
jnh8	121

ing three executions of $HBS(20, 300, 10)$ on a particular instance of the dataset. As shown in the figure, there is a rough correspondence of the reached D level and the achieved solution quality. In particular, the optimum solution (OPT) was found during the run which reached the highest D value. The other two runs found successively worse solutions, OPT-1 and OPT-2 respectively, whereas the reached D values were lower. Although this is not always the case, it is generally desirable that the algorithm reaches soon a state of high D value (capturing the backbone) and keeps searching in this state for a long time.

The diagram of Fig. 5 confirms experimentally the convergence of the sample set $\mathcal{S}$ to a collection of backbone assignments. Control of the convergence speed and level (as indicated by D) remain as challenging matters of study.

5 Conclusions and Future Work

In this paper, we examined experimentally the effectiveness of sampling heuristically the backbone structure for hard CNF formulae, in order to provide a hill climbing heuristic with effective startup states. The algorithm *HBS* was introduced. Experimentations with *HBS* revealed remarkably improved behaviour on both weighted and unweighted MAXSAT instances.

The introduced stochastic initialization scheme seems to be a computing artifact which bears theoretical investigation in combination with the backbone theory. An interesting challenge concerns estimating the expected quality of a stochastically produced assignment with respect to qualities contained in $\mathcal{S}$. Theoretical identification of conditions ensuring that $\mathcal{S}$ will eventually converge to a collection of backbone assignments also constitutes a matter of future work.

Dynamic tuning of *HBS*'s parameters is an interesting aspect in its own right. Specifically, the size of the assignments sample $\mathcal{S}$ appeared to be of great importance for the method's performance during our experiments. For problem

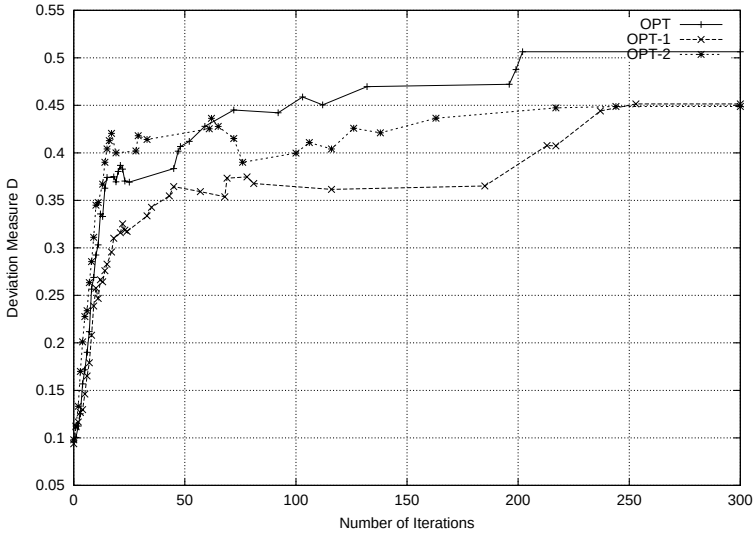

Fig. 5. Maximization of deviation measure D

instances of size similar to the ones' presented here, $10 \leq |\mathcal{S}| \leq 15$ seems to be a proper range.

The encouraging experimental results obtained on weighted CNFs confirmed *HBS*'s generic heuristic value. It gradually discovers valuable partial assignments through sampling, thus guiding the search to promising regions of the search space. Therefore, we consider experimenting with the algorithm on several NP-hard optimization problems, such as maximum graph bisection, coloring, and cut.

References

1. Cook S. A. The Complexity of Theorem-Proving Procedures. In *Proceedings of the 3rd IEEE Symposium on the Foundations of Computer Science*, pages 151–158, 1971.
2. Dubois O., Dequen G. A Backbone-Search Heuristic for Efficient Solving of Hard 3-SAT Formulae. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI '01)*, 2001.
3. Goemans M., Williamson D. P. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the ACM*, 42:1115–1145, 1995.
4. Jiang Y., Kautz H., 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.
5. Koutsoupias E., Papadimitriou C. H. On the Greedy Algorithm for Satisfiability. *Information Processing Letters*, 43(1):53–55, 1992.

6. Mitchell D., Selman B., Levesque H. Hard and Easy Distribution of SAT Problems. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI '92)*, pages 440–446, 1992.
7. Monasson R., Zecchina R., Kirkpatrick S., Selman B., Troyansky L. (2+p)-SAT: Relation of Typical-Case Complexity to the Nature of the Phase Transition. *Random Structures And Algorithms*, 15:414–435, 1999.
8. Monasson R., Zecchina R., Kirkpatrick S., Selman B., Troyansky L. Determining Computational Complexity from Characteristic 'Phase Transitions'. *Nature*, 400:133–137, 1999.
9. Parkes A. J. Clustering at the Phase Transition. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 340–345, 1997.
10. Resende M. G. C., Pitsoulis L. S., Pardalos P. M. Approximate Solution of Weighted MAX-SAT Problems Using GRASP. In *Satisfiability Problem: Theory and Applications*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 393–405, American Mathematical Society, 1997.
11. Schiex T., Fargier H., Verfaillie G. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI '95)*, pages 631–639, 1995.
12. Selman B., Kautz H. A., Cohen B. Noise Strategies for Improving Local Search. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI '94)*, pages 337–343, Seattle, 1994.
13. Slaney J., Walsh T. Backbones in Optimization and Approximation. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI '01)*, 2001.
14. Wah B. W., Shang Y. Discrete Langrangian-Based Search for Solving MAX-SAT Problems. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI '97)*, pages 378–383, 1997.
15. Yannakakis M. On the Approximation of Maximum Satisfiability. *Journal of Algorithms*, 17:475–502, 1994.