A New Population-Based Method for Satisfiability Problems

Jin-Kao Hao and Raphaël Dorne [1]

Abstract. This paper presents the mask method (MASK), a new population-based, evolutionary search procedure for finding models for satisfiability problems (SAT). In this method, a partial truth assignment for a given boolean expression containing N variables is represented by a partially instantiated 0/1 string with N positions (called mask), with each position coding one variable. The mask method begins with a population of masks. An iteration process follows to evaluate each mask in the population using an evaluation mechanism, then to discard half of them, and finally to divide each remaining mask into two new ones by fixing a free position in the mask to 0 and 1 respectively. This process continues until all the positions in the mask have a fixed value. The method is compared with a class of genetic algorithms (GAs) on a set of SAT instances and proves to be much more efficient.

1 INTRODUCTION

The satisfiability problem or SAT [5] is of great importance in Artificial Intelligence both in theory and in practice. The statement of the problem is very simple. Given a well-formed boolean expression E, is there a truth assignment that satisfies it? In theory, SAT is one of the six basic core NP-complete problems. In practice, many applications such as VLSI test & verification, consistency maintenance, fault diagnosis, planning and so on can be formulated with SAT. SAT is originally stated as a decision problem. However, there are many interesting problems which are related. We can mention, among others, 1) model finding: find one satisfiable assignment, find all satisfiable assignments, find the "best" assignment under some criteria constraints; 2) model counting: find the number of solutions, the number of non-isomorphic solutions; 3) information deduction.

Although the SAT problem is NP-complete, many methods have been devised to satisfy practical needs. These methods can be roughly classified into two large categories: complete and incomplete methods. As examples of the first category, we can mention the logic-based approach (the Davis & Putnam algorithm, Binary Decision Diagrams), the constraint-network-based approach (CSP, local propagation), and 0/1 linear programming. The second category includes such methods as evolutionary algorithms, simulated annealing, and local search. Theoretically, complete methods are able to find all the solutions to any boolean problem, or to prove that no solution exists provided that no time constraint is present. However, the combinatorial nature of SAT makes these complete methods impractical when the size of the problem increases. On the other hand, incomplete methods look for efficient heuristics which help to delay this explosion as late as possible at the expense of completeness. Indeed, incomplete methods have recently proven that they are able to solve hard SAT instances in a wide class of problems ([3], [14], [10], [6], [8], [13] [11]).

We are especially interested in model finding. In this paper, we present the mask method, MASK for short, a population-based, incomplete, evolutionary search procedure. MASK has some characteristics similar to classic genetic algorithms (GAs) such as population and binary coding. The principle of the method is simple. It is based on three elements: a set of partially instantiated 0/1 strings (called masks) representing potential solutions to the given problem, an evaluation function and an evolution mechanism. The method begins with a population of masks. Each mask is then scored using the evaluation function. According to their scores, high-scored masks are allowed to evolve by instantiating one more non-fixed variable and low-scored masks are eliminated. This process continues until all variables in the masks are instantiated. Such completely instantiated masks will be expected to be solutions to the boolean expression.

The paper is organized as follows. In §2, the mask method is presented in detail and its associated interpretation is given. In §3, results of experiments and comparisons with GAs are presented. In the last section, different possibilities for improving the method and future work are discussed.

2 THE MASK METHOD (MASK)

2.1 Some definitions

A *mask* of length N is an N-position, ordered string composed of fixed values 0, 1 and non-fixed values represented by $*$. Any mask of length N with P fixed positions is denoted by a generic mask M($V_1 ... V_P, F_1 ... F_{N-P}$) where V_i=0 or 1($1 \leq i \leq P$) and $F_j = *$($1 \leq j \leq N-P$).

The length N of a mask is defined as the number of boolean variables of the boolean expression at hand, each position in the mask coding a boolean variable. Masks are particularly appropriate for representing a set of potential solutions to the given boolean expression: fixed positions are instantiated boolean variables and free positions are variables to be instantiated by 0 or 1. Moreover, masks are organized into populations to represent a bigger set of potential models. Masks are the basic element manipulated by the mask method.

[1] EERIE-LERI, Parc Scientifique G. Besse, F-30000 Nîmes, France email: {hao, dorne}@eerie.fr

ECAI 94. *11th European Conference on Artificial Intelligence* Edited by A. Cohn
Published in 1994 by John Wiley & Sons, Ltd.

A *RGS* of length N is a *randomly generated string* of N-positions composed of fixed values 0 and 1.

An *evaluation function* is a function allowing the "fitness" of a potential solution represented by a 0/1 string to be evaluated with respect to a given boolean expression. There are many ways to define this function. In this paper, we use the function **Eval** defined below. Let E be a succinct boolean expression,[2] V(E)={$V_1 ... V_N$} the ordered set of all the variables in E, and S a 0/1 string of length N. **Eval**: E×S→[0,1] is then recursively defined as follows:

1. Eval(V_i,S) =0 if the ith position of S is 0
   =1 if the ith position of S is 1
2. Eval($\bar{V_i}$,S)=1-Eval(V_i,S)
3. Eval($E_1 \wedge \ldots \wedge E_m$,S)= Ave(Eval($E_1$,S) ... Eval($E_m$,S))
4. Eval($E_1 \vee \ldots \vee E_m$,S)= Max((E_1,S) ... Eval(E_m,S))

The following properties of this evaluation function can be easily verified.

1. 0≤Eval(E,S)≤1
2. Eval(E,S) = 1 iff S is a solution to E.
3. Eval(E,S) < 1 iff S is not a solution to E.

2.2 The method

Like most evolutionary approaches, the mask method is based essentially on three elements: a population of partially instantiated masks, an evaluation function and an evolution mechanism. Masks represent partial truth assignments (potential solutions) to the given boolean expression. The evaluation function allows the masks to be measured with respect to the given problem. Finally, the evolution mechanism eliminates some masks and makes each remaining mask evolve to the next generation by producing two new masks. Figure 1 gives the general outline of the method.

Initialize mask-population P (containing 2^K masks,
each having K positions instantiated to one
different integer from 0 to 2^K-1
while ∃ free-position in masks **do**
{
evaluate P;
select the better half of P (2^{K-1} masks);
divide each selected mask into two masks by
fixing a free-position to 0 and 1;
}

2.2.1 Initialization

Initial mask-population: A population of 2^K (K≥1) masks of length N is constructed as follows. 2^K strings of length N are first generated. The leftmost K positions are then set to the 2^K integers from 0 to 2^K-1. The other N-K positions are left undefined. This initialization thus creates a population of 2^K masks of the form M($V_1 ... V_K$,$F_1 ... F_{N-K}$) where V_i=0 or 1(1≤i≤K) and F_j=∗ (1≤j≤N-K). For example. {00∗∗∗∗∗∗, 01∗∗∗∗∗∗,10∗∗∗∗∗∗,11∗∗∗∗∗∗} is a population of 2^2(=4) masks of length 8 and represents $2^2 \times 2^6 = 2^8$ potential solutions to a boolean expression containing 8 distinct variables.

[2] A boolean expression is said to be succinct if it contains only *and*, *or*, and *not*, and if the negations are all pushed down to the boolean variables using DeMorgan's rules.

Generation of a RGS-set: A set of RGS of length N is constructed as follows. For each position of each RGS, the value 0 or 1 is randomly generated and set to the position. As will be explained in the next section, the RGS-set will only be used to score the masks and will not evolve.

2.2.2 Evolution

After initialization, an evaluation-selection-division cycle follows to make the mask-population evolve towards solutions.

a) Mask evaluation The aim of the mask evaluation is to measure the "fitness" of partially instantiated masks with respect to the given boolean expression. The difficulty concerning this evaluation is to have a good measuring rule. However, no exact rule is available and only approximate rules can be used. MASK uses **Eval** and RGS together as the approximate rule. The evaluation function **Eval(E,S)**, defined in §2.1, works only with fully instantiated 0-1 strings. In order to use this function to evaluate masks which contain free positions, we must complete these non-fixed positions in some way. Moreover, in order to guarantee homogeneity, the same completion mechanism should be applied to the whole mask-population.

Suppose that M($V_1 \ldots V_P, F_1 \ldots F_{N-P}$) is the mask to be evaluated with V_i=0 or 1(1≤i≤P, P being the number of fixed positions until now) and F_j=∗ (1≤j≤N-P, N-P being the number of free positions), and also that L is the size of the RGS-set, we give below three possible evaluation mechanisms using Eval and RGS-set. The first mechanism is to replace $F_1 \ldots F_{N-P}$ of the mask M($V_1 \ldots V_P, F_1 \ldots F_{N-P}$) with the corresponding values of each RGS, leading to L instantiated masks, then to score each transformed masks using bf Eval, and finally to take the average of all the scores as the score of the mask.[3] In the second evaluation mechanism, instead of using the average, we take the maximum from among all the scores obtained from the L transformed masks. The third approach is to evaluate only those RGS the P leftmost positions of which are equal to $V_1 \ldots V_P$ of the mask and then to take the average or the maximum of these scores. The first two mechanisms apply whatever the size of the RGS-set while the third does not because there may be no such RGS in the RGS-set. In the experiments reported in §3, the first evaluation mechanism was used.

Note that the RGS-set is *only used to measure the quality of each mask and will not evolve.* The size of this RGS-set is a user-controlled parameter which can be changed to influence the precision of mask measurement. In extreme cases, the RGS-set may contain a single random 0/1 string.

b) Mask selection Once all the 2^K masks of the current population have been evaluated and scored, those with low scores are eliminated leaving the 2^{K-1} masks which have better scores in the mask-population.

c) Mask division This step allows the current mask-population to effectively evolve into a new mask-population with the help of the division operation defined as follows.

For each M($V_1 \ldots V_P, F_1 \ldots F_{N-P}$) of the 2^{K-1} remaining masks, the following operations are carried out:

1. The next free position F_1 in the mask M is chosen.

[3] If one of the transformed masks happens to be evaluated at the value 1, then it is a solution and the method stops.

2. M($V_1 \ldots V_P, F_1 \ldots F_{N-P}$) is divided into two new masks M($V_1 \ldots V_P, \mathbf{0}, F_2 \ldots F_{N-P}$) & M($V_1 \ldots V_P, \mathbf{1}, F_2 \ldots F_{N-P}$).
3. These two new masks are added to the mask-population.

This evolution process (steps **a-c**) continues until all positions in the masks are instantiated. Since there are N-K free positions in the masks after initialization and each division fixes one free position, it is clear that this process repeats exactly N-K times before all positions receive a fixed value. Two cases may occur at this stage: either one of the fully instantiated masks is a solution; or none of the masks is a solution. In the first case the process stops. In the second case, we know that no solution has been found during this execution, thus a new execution is started with a new random RGS-set. As is explained above, the set of RGS and **Eval** together constitute the approximate measuring rule for mask evaluation. If no solution is found during one execution, this implies that the measuring rule used in this execution is not sufficiently precise. As **Eval** is fixed, the only way to change the rule is to vary the RGS-set.

2.3 Interpretation

In the following sections, we will always use N as the length of a mask, and K (K≤N) as the number of initially fixed positions in a mask. We remind the reader that N is defined as the number of boolean variables of the boolean expression (BE) at hand and each position in the mask represents a boolean variable. As a result, a mask with NF free positions represents a set of 2^{NF} potential models of BE. In fact, replacing the NF free positions by 0 or 1 gives a possible truth assignment to BE. Therefore, a population of 2^K masks each having NF free positions represents a collection of 2^{NF+K} models.

The mask method may be better viewed as a heuristic-guided search procedure in a complete binary search tree composed of 2^N branches. It is easy to see that a mask of length N with NF free positions covers a subspace of 2^{NF} branches of a total search space of size 2^N. Therefore, a population of 2^K masks covers 2^K subtrees representing 2^{NF+K} branches. If we define the initial masks which have N-K free variables as the first level nodes in the partial search tree traveled by the method, we can show that this partial search tree has the following properties:

1. A node (mask) at the level i (1≤ i ≤N-K) has N-K-i+1 free variables.
2. At any level i (1≤i≤N-K), there are exactly 2^K nodes.
3. The first level represented by the initial 2^K masks covers the total search space 2^N.
4. Any higher level covers a search space of 2^{N-i+1} branches.
5. The lowest level, i.e. N-K+1 level, covers a search space of 2^K branches.

The exploration of the search tree is carried out in a top-down, level-by-level manner. As the search progresses, the search space decreases. Beginning with the 2^K initial masks covering the total search space 2^N, the method subsequently explores 2^K branches in the search tree. At each level i≥1 covering $2^K \times 2^{N-K+i-1} = 2^{N-i+1}$ branches, only half of these 2^K partially instantiated branches are allowed to go down to the next level. When the search reaches the leaves of the search tree, two cases are possible: either the instantiation of the N variables of a branch verifies the given boolean expression and thus represents a solution, or the branch is not a solution. In the first case, the search stops. In the second case, the search is restarted from the first level to try other branches.

By retaining a set of partially instantiated masks during the search, MASK has many different alternatives at each iterative step. Consequently, MASK demonstrates good diversity during its search. Another point is that larger mask populations do in fact cover bigger subspaces, and thus the search is more likely to find a solution during one execution.

2.4 Complexity

The initialization step takes time Θ(P∗N) where P=max{L1, L2}, L1 being equal to |Mask-population| and L2 to |RGS-set| and N the length of masks. The complexity of the evolution step is determined by the mask evaluation, mask selection and mask division. Among these steps, the evaluation is the most computationally intensive one. The complexity of the mask evaluation depends on the given boolean expression and takes time Θ(L1∗L2∗X), X being the complexity of one evaluation. For 3-SAT, this complexity becomes Θ(L1∗L2∗NC), NC being the number of clauses. Both the mask selection and division take time Θ(L1∗logL1).

3 RESULTS OF EXPERIMENTS

The method has been implemented in C. The implementation integrates a user interface with which one can easily control various parameters such as the size of the mask-population, the size of the RGS-set, the number of independent runs and so on. To test MASK, we have used several structured SAT instances and a family of Hamiltonian Circuit (HC) problems defined in [3][12] and Hamilton's World Tour (WT) problem [1]. The desirable property of these problems is that we can control their size and difficulty. Most of them, especially HC and WT, are believed to be hard SAT instances.

MASK was compared in detail with genetic algorithms, and in particular, with the GA described in [3][12] that will be referred to as DSGA. Due to the limited length of the paper, all the results cannot be presented here. We only underline that great care was taken to make sure that the same SAT instances and the same criteria were used in the comparison. Results indicate that MASK outperforms DSGA in all the test examples used by [3][12]. In this section, we give some examples. More details about the context of the comparison and the results can be found in [7].

Like [3][12], we use the evaluation number (EN) (defined as the number of evocations of the **Eval** evaluation function) as the first comparison criterion. For MASK, a second criterion used is the number of failures (FN) before a solution is found. This number roughly corresponds to the number of re-runs in GAs. In our experiments, unless indicated explicitly, we used a population of 2^5=32 masks for the search and a set of 3 RGSs and **Eval** for the mask measurement. The same **Eval** was used by DSGA. The size of the chromosome-population used in DSGA was fixed at 100. Results of MASK represent data averaged over 20 independent runs while results of DSGA from [3][12] represent data averaged over 10 runs.

Table 1 and 2 present the results of the comparison between DSGA and MASK for the following two SAT instances:

$(X_1 \wedge \ldots \wedge X_n) \vee (\neg X_1 \wedge \ldots \wedge \neg X_n)$ (TP)
which has two solutions (all 0s and all 1s) and
$(X_1 \wedge \ldots \wedge X_n) \vee (X_1 \wedge \neg X_1 \wedge \ldots \wedge \neg X_n)$ (FP)
in which one solution (all 0s) is now almost correct and the only correct solution is that of all 1s. The results indicate that MASK scales up better than DSGA for these problems. Note that MASK always finds a solution to TP for each run and needs at most 2 runs to find a solution for FP.

Vars	En(DSGA)	En(Mask)	FN(Mask)
20	1 200	1 152	0
60	5 000	4 992	0
80	11 000	7 200	0
90	13 000	8 064	0
200	?	18 432	0
400	?	28 320	0
600	?	57 120	0
800	?	76 416	0
1000	?	95 616	0

Table 1. Comparison of DSGA and MASK for TP

Vars	En(DSGA)	En(Mask)	FN(Mask)
20	2 000	1853	0.3
60	18 000	8 390	0.6
80	20 000	11 268	0.6
90	35 000	12 979	0.5
200	?	31 766	0.7
400	?	48 980	0.2
600	?	152 288	1.6
800	?	133 500	0.75
1000	?	143 400	0.5

Table 2. Comparison of DSGA and MASK for FP

Table 3 shows the results for the so-called 6-peak problem which has one global optimum and 5 local optima.
3-peak: $(X_1 \wedge ... \wedge X_n) \vee (X_1 \wedge \neg X_1 \wedge ... \wedge \neg X_n) \vee$
$(X_1 \wedge \neg X_1 \wedge \neg X_2 \wedge ... \wedge \neg X_{n/2} \wedge X_{n/2+1} \wedge ... \wedge X_n)$
4-peak: 3-peak $\vee$
$(X_1 \wedge \neg X_1 \wedge X_2 \wedge ... \wedge X_{n/2} \wedge \neg X_{n/2+1} \wedge ... \wedge \neg X_n)$
5-peak: 4-peak $\vee$
$(X_1 \wedge \neg X_1 \wedge X_2 \wedge \neg X_3 \wedge X_4 \wedge \neg X_5 ... \wedge \neg X_{n-1} \wedge X_n)$
6-peak: 5-peak $\vee$
$(X_1 \wedge \neg X_1 \wedge \neg X_2 \wedge X_3 \wedge \neg X_4 \wedge X_5 ... \wedge X_{n-1} \wedge \neg X_n)$
The result of DSGA for this problem is not reported in [3]. However, from another paper [4], we do have some information about the performance of DSGA for this problem. In fact, a large number of generations was needed to find a solution. For example, about 150 generations are needed to solve the easier 5-peak problem, with a population of 1000 individuals. This corresponds to about $150 \times 1000 = 150000$ evaluations. In contrast, MASK needs on average only two runs to find the solution for these problems with much lower EN.

Vars	En(DSGA)	En(Mask)	FN(Mask)
30	150 000	3 496	0.7
100	?	16 521	0.8
200	?	37 151	1
400	?	75 748	1.2
600	?	85 608	0.6
800	?	104 050	1
1000	?	191 060	1

Table 3. Results of MASK for 6-peak problem

Table 4 presents the results of the comparison for the Hamiltonian Circuit (HC) problem in a particular class of directed graphs constructed as follows. The N nodes of a graph are labeled using consecutive integers from 1 to N. For any 1<i<j<N, there is a directed edge from i to j. The first node has directed edges to all other nodes except the Nth node. The Nth node has a directed edge back to the first node. Such a graph of N nodes has N×(N-1)/2 edges. Note that although only one Hamiltonian tour exists in such a graph, there are many almost complete tours (local optima) scattered throughout the search space. In order to solve this problem, we first transform it into a SAT problem in the following way. Each edge is represented by a boolean variable. Each node is specified by its input and output constraint on boolean variables. A Hamiltonian circuit is thus the conjunction of the input/output constraints associated with all of the nodes. The same transformation is used in DSGA.

The first column of Table 4 indicates the number of nodes in a graph, and the second column the number of boolean variables for coding the graph. In [3][12], tests were performed within the range $4 \leq N \leq 10$. When the number of nodes goes above 10, the problem becomes intractable for DSGA. On the contrary, for up to 20 nodes (about 200 variables), the effort needed for MASK to find the solution increases in an almost linear manner. Moreover, for up to 14 nodes, MASK needs at most two runs to find the solution. As can be seen in the table, even for a greater number of nodes, the number of re-runs required remains reasonably small.

Nodes	Vars	EN(DSGA)	EN(Mask)	FN(Mask)
7	21	20 000	3 500	0.2
8	28	70 000	4 100	0.2
9	36	300 000	5 600	0.2
10	45	800 000	12 600	1
14	91	?	28 300	1
18	153	?	62 000	10
22	231	?	126 760	2.5
26	325	?	906 880	16
28	378	?	1 046 300	16.5
30	435	?	2 076 500	30

Table 4. Comparison between DSGA and MASK for HC

Table 5 presents the results of MASK for Hamilton's World Tour (WT) problem which concerns a graph of 20 nodes representing a dodecahedron [1]. This problem should be difficult since there are many local optima in it. No result is available for DSGA for this problem. The results of MASK show also that bigger mask-populations have indeed a positive influence on the search performance for this problem.

Num. Mask	Num. RGS	EN	FN
8	10	420 944	898
16	10	332 112	35.5
32	10	174 080	8.8
64	10	122 240	2.6

Table 5. Results of MASK for WT problem

In addition to the above comparisons, we improved DSGA by tuning various factors such as selection, mutation, and population-size. Even if these improved versions performed better than DSGA for TP & FP, their behavior for HC was similar: when the number of nodes goes beyond 10, EN increases very rapidly. This phenomenon was also observed in [3] when they used an improved version of the **Eval** evaluation function. Moreover, for WT, none of our GAs was able to find the solution. Note also that compared to the total search space of these problems, the size of the mask-population (2^5=32) used in our experiments represents a very small fraction of this space. However, the method manages to find solutions rapidly.

4 DISCUSSION & CONCLUSIONS

The results of experimentation presented above are based on an elementary implementation of the method. Many improvements are possible concerning both the method itself and the implementation. In this section, we identify some of those possibilities and discuss future work.

Parallelism: At least two aspects of the mask method can be naturally parallelized: the mask evaluation and the mask division, since each of these two operations is independently applied to individual masks in a mask-population.

Variable ordering: As presented in this paper, variable instantiations in both the initialization and the division operations are carried out from left to right. If we suppose that all variables in a given expression are equally important,[4] then this particular instantiation order will not be harmful. However, this hypothesis does not necessarily hold for some problems where the order of variables is indeed important. Therefore, it will be desirable to put the variables in a decreasing order according to their importance. In this way, important variables will be instantiated first. For random boolean expressions, we can suppose that all variables are equally important. Consequently, MASK should work well for these problems. We are working in this direction to confirm (or disprove) this claim.

Multi-evaluation functions: The precision of the evaluation mechanism is essential for all evolutionary approaches including the mask method. As was explained in §2.2, two factors determine the evaluation of a mask: the RGS and the evaluation function. In this paper, we have used a special and fixed function, i.e. **Eval**, and a RGS-set to measure masks. It is clear that it will be better to have a set of evaluation functions, each specializing in a class of problems. Equipped with this multiple evaluation function mechanism, the method should be able to select the most appropriate function for measuring masks. In this way, the search will be better guided.

Combination with other methods: The mask method can be used as a pre-processing procedure for other search procedures working with initial configurations (0/1 strings) such as local search and simulated annealing. Another possible combination is to integrate local search inside population-based approaches such as MASK.

In this paper, we have presented MASK, an evolutionary search procedure for finding models for SAT. The method has been compared with a class of GAs using a set of structured SAT instances and better performance has been shown. Even if more tests on a broader class of problems are desirable, we think the method is promising and worthy of more investigation. In particular, it will be interesting and important to understand better its behavior and to identify the classes of problems which may be solved with the method.

We are currently trying to compare our method with other incomplete methods such as GSAT [10][11] as well as efficient implementations of the Davis-Putnam procedure such as Tableau [2]. In order to do this, we must select a sufficiently large set of *objective* test examples and good comparison criteria to make comparisons really meaningful. As the first choice, we will use such hard random instances as proposed in [9] and those used in the second DIMACS challenge. Indeed, research on random formulae has recently drawn much attention and important progress has been carried out in this field. Besides, SAT encodings of other problems such as circuit synthesis and diagnosis will also be considered. Note that since MASK works with more general boolean formulae, it also works with these clausal formulae.

For clausal formulae, we are also investigating another evolutionary method which uses an efficient internal representation of SAT instances and intelligent recombination operators based on local search. The goal is to combine the better diversity offered by evolutionary approaches and the efficiency of the local search to avoid local optima as much as possible.

[4] The notion of importance can be defined according to many criteria. For example, a variable appearing several times in a given expression can be considered to be more important than a variable appearing only once. In terms of constraints, a variable having more constraints on it will be an important one.

ACKNOWLEDGEMENTS

We would like to thank the referees for their useful and pertinent comments which helped improve this paper. The work reported in this paper is partially supported by the BAHIA projet (PRC-IA: Programme de Recherches Coordonnées en Intelligence Artificielle).

REFERENCES

[1] Corman T.H. Leiserson C. and Rivest R.L. 'Introduction to Algorithms', MIT Press, Cambridge, MA, 1992.

[2] Crawford J.M & Auton L.D. 'Experimental Results on the Cross-Over Point in Satisfiability Problems'. Proc. of AAAI-93, Washington DC, 1993, pp21-27.

[3] De Jong K.A. & Spears W.M. 'Using Genetic Algorithms to Solve NP-Complete Problems'. Intl Conf. on Genetic Algorithms, Fairfax, Virginia, June 1989, pp124-132.

[4] De Jong K.A. & Spears W.M. 'An Analysis of the Interacting Roles of Population Size and Crossover in GAs'. Intl. Workshop on PPSN, Dortmund, Germany, Oct. 1990, pp38-47.

[5] Garey M.R. & Johnson D.S. 'Computers and Intractability: a guide to the theory of NP-Completeness'. Freeman, San Francisco, CA, 1979.

[6] Gu J. 'Efficient Local Search for Very Large-Scale Satisfiability Problems'. SIGART Bulletin, Vol.3, No.1, Jan. 1992.

[7] Hao J.K. & Dorne R. 'An Empirical Comparison of Two Evolutionary Methods for Satisfiability Problems'. EERIE TR No. 93-11-01, Nîmes, France, Nov. 1993.

[8] Koutsoupias E. & Papadimitriou C.H. 'On the Greedy Algorithm for Satisfiability'. Information Processing Letters, Vol.43 1992, pp53-55.

[9] Mitchell D., Selman B. and Levesque H.J. 'Hard and Easy Distributions of SAT Problems'. Proc. of AAAI-92, San Jose, CA, 1992, pp459-465.

[10] Selman B., Levesque H.J., and Mitchell M. 'A New Method for Solving Hard Satisfiability Problems'. Proc. of AAAI-92, San Jose, CA, 1992, pp.440-446.

[11] Selman B., Kautz H.A and Bram C. 'Noise Strategies for Improving Local Search'. to appear at AAAI-94, Seattle, WA, July 1994.

[12] Spears W.M. 'Using Neural Networks and Genetic Algorithms as Heuristics for NP-Complete Decision Problems'. NCARAI TR-AIC-91-019, Washington DC, 1991.

[13] Spears W.M. 'Simulated Annealing for Hard Satisfiability Problems'. NCARAI TR-AIC-93-015, Washington, DC 1993.

[14] Young R.A. & Reel A, 'A Hybrid Genetic Algorithm for a Logic Problem', Proc. of the 9th ECAI, Stockholm, Sweden, Aug. 1990, pp.744-746.