Randomness in Heuristics: an Experimental Investigation for the Maximum Satisfiability Problem

H. Drias
USTHB, Institut d'informatique
BP 32 El-Alia
16111 Alger, Algeria
email: drias@wissal.dz

ABSTRACT. In this paper, a genetic algorithm and a tabu search are investigated for the maximum satisfiability problem. When the evolutionary algorithm is hybridized with the randomized procedure G-bit [14], better performance is achieved and it even outperforms the well known probabilistic procedure GSAT [25]. On the other hand, when the random noise strategy is introduced in the tabu search, the latter competes with GSAT with walk [27] independently of the length of the tabu list. The basic result we can argue from this study is that the robustness of a method seems to be bound to the degree of 'randomness' involved in it, but at the expense of the running time. According to the experiments, GSAT and the genetic algorithm are more powerful than tabu search in its simplest form because they incorporate more 'randomness'. GSAT with random walk is even more interesting than simple GSAT for the same reason. Also heuristic methods and local search become more efficient when a random strategy such as a noise is introduced to deviate the search from its usual rules.

1 Introduction

The satisfiability of a logical formula is a challenging problem studied by a great number of researchers the two last decades. Its wide application to the domain of AI in automatic reasoning and problem solving for instance and other domains like VLSI and graph theory motivates the huge interest shown for this problem. Given a set of boolean variables, we define a literal as a variable with or without a negation, a clause as a disjunction of literals and an instance of satisfiability as a collection (understood as a conjunction) of clauses. An r-SAT instance is a set of clauses having the same number of literals (called length) equal to r. The problem of satisfiability or SAT for short is the pioneer of the NP-completeness theory in the sense that it is the first known NP-complete problem.

Local search has been used to solve the maximum satisfiability or MAX-SAT, an NP-hard optimization problem of finding a valuation of variables that satisfies the maximum number of clauses. Such local search has been successful at finding satisfying assignments for even hard SAT instances. But the major problem with this approach is that it is usually unable to search beyond the local optima [16, 19, 24]. Some of these methods have been enhanced by adding mechanisms for escaping from local optima in order to scan the most important part of the whole search space for attaining global optima [25, 26, 27]. GSAT, the most interesting procedure for local search for MAX-SAT, generates a random assignment and repeats for a number of steps flipping the variable that produces the largest number of satisfied clauses. The process is iterated until an optimal solution is found or a maximum number of iterations is reached.

Genetic algorithms or GA have been applied to complex optimization problems with remarkable success in some cases. Their behavior mimics the process of natural evolution. A population initially made of candidate solutions representing individuals improves towards another population of individuals with higher quality along a process repeating a finite number of time, sequentially reproduction between individuals, mutation of chromosomes and selection of better individuals. The goal is to create an individual which is very fit, in our case an assignment that satisfies the maximum number of clauses.

Tabu search or TS is another metaheuristic used for solving difficult problems. Unlike the GA approach, the TS considers one candidate solution at a time. It is a history sensitive heuristic, it performs a local search enriched with a scheme used to pursue the search beyond local optima. The information gathered along the process at some point makes the history at this stage and is exploited to reach territories not yet explored. Tabu search has been investigated for MAX-SAT [2] and for SAT with a special interest to random hard r-SAT instances [20].

In this paper, we describe four algorithms for MAX-SAT; GA-SAT based on the GA approach, GA-SAT+ a hybrid genetic algorithm, RNSTS-SAT a variant of TS called TS with random noise and RNSTS-SAT+, a variant of the last algorithm with an intensification search strategy. We then present the extensive experimental tests performed for these algorithms and for the procedure GSAT with the random walk strategy called RWS-GSAT [27]. At the end, we discuss the results obtained and conclude with a comparison between all these algorithms.

2 A genetic algorithm for MAX-SAT

The modelization of the problem MAX-SAT by means of a genetic algorithm necessitates the definition of the following components; the encoding of solutions, the initial population, the evaluation function measuring the quality of individuals and the genetic operators.

Unlike for many combinatorial optimization problems where the encoding sets a real problem, for MAX-SAT this operation is trivial since a solution is a binary valuation of variables. An individual is represented by a binary chain identifying such valuation and the measurement of solutious quality is evaluated as the number of clauses satisfied by this assignment.

We propose to generate at random the initial population for a first batch of experimental tests. A second way to create the initial population is to spontaneously generate individuals using a local search method and inject them among random individuals.

The selection phase favors the fittest individuals for participating at the reproduction process. The probability for an individual to be chosen is proportional to its evaluation function so that the individuals with the highest value are likely to be selected.

The crossover of two individuals consists in exchanging parts of their chromosomes creating in this way an offspring. The crossover operator when applied must respect the constraints of the problem. As an example of a constraint is to avoid revisiting a city in the traveling salesman problem. In MAX-SAT, no such constraint exists, therefore this specific characteristic help us to make a choice between the one-point, the two-points or the uniform crossover. The type of crossover considered is a combination of the three and a special uniform crossover with a mask determined by the experiments.

The mutation viewed usually as a background operator consists in modifying one or many chromosomes chosen at random with a probability of mutation called mutation rate. For MAX-SAT, this operator is merely the flipping of a bit drawn randomly. In order to palliate the problem of premature convergence due to the similarity of individuals in the population, we apply an adaptive mutation; we decrease the mutation rate during the first iterations of the algorithm where the individuals are likely to be different, so as to speed the process and we increase the mutation rate at the last iterations so as to promote diversification of individuals and escape from local optima.

The final step of the GA is the replacement of the bad individuals of the population by the fittest ones of the offsprings generating this way a new population. The whole process is iterated until finding an optimal solution or reaching a maximum number of iterations imposed by the computational limits.

2.1 A Hybrid genetic algorithm for MAX-SAT

We propose to hybridize the simple GA with a heuristic method called G-bit in order to improve the solutions. The latter consists in modifying the bits of the solutions that enhance their quality. G-bit is applied to good solutions, each of them is scanned bit by bit and the current bit is complemented with another one drawn at random. The solution obtained is saved if it is better than the old one otherwise it is rejected. The procedure G-bit is written as:

```
Procedure G-bit()
For each solution do
begin
  For $i = 1$ to indiv-size do
  begin
    flip the $i$th variable;
    choose at random a variable and flip it;
    evaluate the new individual;
    if the new individual is better
      then substitute this individual to the old one
      else reject the new individual;
  end
end
```

indiv-size expresses the number of chromosomes of an individual. The hybrid genetic algorithm called GA-SAT is as:

Algorithm GA-SAT+

input: an instance of satisfiability

output: an assignment of variables and the maximum number of satisfied clauses

```
Begin
  generate at random the initial population;
  while (the maximum number of generations is not reached
         and the optimal solution is not found) do
  begin
    repeat
      select two individuals;
      generate at random a number $Rc$ from [0,100];
      if $Rc <$ crossover rate then apply the crossover;
      generate at random $Rm$ from [0,100];
      while $Rm <$ mutation rate do
      begin
          choose at random a chromosome from the individual
```

```
            obtained by the crossover and flip it;
            generate at random $Rm$ from [0,100];
        end;
        evaluate the new individual;
    endrepeat;
    replace the bad individuals of the population by the fittest new ones;
    G-bit();
  end;
end;
```

When the call to the procedure G-bit is suppressed, we obtain the simple genetic algorithm for MAX-SAT called GA-SAT.

3 Tabu search for MAX-SAT

Three Tabu search algorithms have been designed for MAX-SAT; a simple TS introducing the basic components of the short term memory, a TS with a random noise strategy and a TS with an intensification search strategy.

3.1 A simple Tabu Search

The simple TS algorithm uses the basic elements of the short term memory. An elementary move consists in flipping one of the variables of the solution. The neighborhood of a solution is constituted by all the solutions obtained by applying an elementary move on this solution. A variable has the tabu state if it has been modified during the current move and it keeps it during a certain number of iterations called tabu tenure. A move is considered tabu if the variable that is flipped in this move is tabu. The tabu state of a move is removed if the occurring in the instance of the variable that is flipped in this move is greater than a given threshold (the aspiration criterion). The search stops when the quality of the solution is not improved during a maximum number of iterations or when we reach the optimum. With these parameters we state our simple algorithm as:

Algorithm TS-SAT

input: an instance of SAT

output: an assignment of variables and the maximum number of satisfied clauses

```
begin
Let $x$ be an initial solution;
$x* = x$, $f* = f(x)$;    ($f$ computes the number of satisfied clauses by $x$)
While the stop criterion is not satisfied do
    begin
        compute $N$ the neighborhood of $x$ by ignoring the tabu moves
```

```
            not removed by the aspiration criterion;
        $x$ = best solution of $N$, $f = f(x)$;
        if $f > f*$ then $f* = f$, $x* = x$ endif;
    end;
output $(x*, f*)$;
end
```

3.2 A Tabu Search with a random noise strategy

From the experimental point of view, GSAT with walk or RWS-GSAT for short, outperforms the simple TS and shows its robustness for the quality of solutions. For this reason, the 'random noise' strategy was introduced in TS-SAT to yield the algorithm RNSTS-SAT. The noise disturbs the search and turns it away from its usual rules that choose the best non tabu neighbor and generates at random a neighbor. This strategy presents the advantages of applying some diversification and speeding the search. The noise injected is a probability p with which a solution is built according to the classical rules. A solution is chosen at random with a probability equal to $(1-p)$.

Algorithm RNSTS-SAT
input: an instance of SAT
output: an assignment of variables and the maximum number of satisfied clauses

```
begin
Let $x$ be an initial solution;
$x* = x$, $f* = f(x)$;    ($f$ computes the number of satisfied clauses by $x$)
while the stop criterion is not satisfied do
begin
   compute $N$ the neighborhood of $x$ by ignoring the tabu moves
     not removed by the aspiration criterion;
   if (RandomNumber() $< p$)
     then $x$ = best solution of $N$
     else $x$ = a random neighbor of $x$ endif;
   $f = f(x)$;
   if $f > f*$ then $f* = f$, $x* = x$ endif;
end;
output $(x*, f*)$;
end
```

RandomNumber() generates a random number between 0 and 1. The value attributed to p is critical. When p increases, the search is speeded up but it loses parts of its performance. On the contrary, when p decreases, the search is slowed down and a gain in efficiency is recorded.

3.3 A tabu search with an intensification search strategy

RNSTS-SAT presents yet very good results under experimentations, nevertheless an improvement has been brought to it in order to increase more its efficiency. This is done by adding an intensification strategy for searching more promising territories in terms of solutions quality. RNSTS-SAT with intensification called RNSTS-SAT+ starts by building at random or by means of a heuristic, m good solutions considered as candidates to be enhanced. RNSTS-SAT is then applied for each candidate. If the latter is improved, it is inserted in a list called the candidates list otherwise it is thrown away. This process is iterated until the list becomes empty. RNSTS-SAT with intensification is written as:

```
Algorithm RNSTS-SAT+
input: an instance of SAT
output: an assignment of variables and the maximum number of satisfied
clauses

begin
1- build a candidate list with $m$ solutions by applying $m$ times
   an algorithm for selecting the best solutions from a randomly
   generated set of solutions;
2- if the list is empty then stop else choose a solution from the list
3- Execute RNSTS-SAT for this solution
4- if the solution is improved then replace it by the new one in the list
   else remove it from the list
5- goto 2
```

4 Experimental results

In order to give a sense to the results obtained by our algorithms, we have implemented some existing procedures like GSAT and RWS-GSAT for a comparison goal. All these algorithms have been developed with the C++ language under Linux for Pentium. Two kinds of experimental tests have been undertaken. The goal of the first ones is the setting of the different parameters of the GA algorithm like the mutation rate, the crossover rate, the type of crossover, the number of iterations, the population size and the interaction between these parameters and those of the TS like the stop criterion and the random noise. The second kind of experiments concerns the maximum satisfiability of the MAX-2SAT, MAX-3SAT and MAX-4SAT instances. For each (n, k) representing respectively the number of variables and the number of clauses, 50 instances have been generated at random. On each instance all these algorithms have been executed 10 times in order to compute the average of the maximum number of satisfied clauses. The comparative tables below show the results obtained by RWS-GSAT, GA-

SAT, GA-SAT+, RNSTS-SAT and RNSTS-SAT+. These columns contain the rate of satisfied clauses. The occuring of the symbol '-' means that the running time is extremely large. Note that GA-SAT competes with RWS-GSAT with a more reasonable running time. For MAX-2SAT, GA-SAT is less efficient and GA-SAT+ is more interesting but slower. On the other hand RWS-GSAT outperforms the simple TS with a non optimal tabu list. This is the reason why RNSTS-SAT was tested. In fact the latter yields almost the same performance as RWS-GSAT. And in general observe the superiority of the genetic algorithm in comparison with the others from the performance and the running time point of view.

5 Discussion and Conclusion

Note first that all of the experimented algorithms use the same elementary move, which is the flipping of one variable of a solution (TS and GSAT share more common features). The most 'probabilistic' way to undertake this operator is used in GA-SAT, GSAT and even more in their respective variants GA-SAT+ and RWS-GSAT. This may be the reason why these algorithms are more efficient than tabu search in its simplest form with a non optimal tabu list. It may be explained by the fact that the diversification in the search space is very high. When the random noise is introduced in TS, we supply more 'randomness' to the procedure thus more diversification in the search and the result is better. In the same sense, when G-bit is incorporated in GA-SAT, the solutions space is better searched.

This observation leads us to conclude that randomness in heuristics methods play an important role as for the technique performance. The setting of optimal parameters for meta-heuristics by experiments indeed increases their efficiency but in default of optimality, randomness may be introduced. This result follows the same direction as the recommendation expressed by Cook [5] for exploring stochastic methods in solving complex problems.

MAX-2SAT

n	k	GA-SAT+	RWS-GSAT	GA-SAT	RNSTS-SAT	RNSTS-SAT+
10	30	96	95.50	95.20	95.67	95.67
	50	92.50	91.40	91	91.40	91.40
	80	88.59	88.00	87.33	88	88
30	50	98	98.50	96	98.77	99.20
	100	94	93.60	93	94	94.20
	150	91.33	91.07	90.66	92	92
50	100	97	96.66	96	95.60	97.40
	200	93.50	90.50	92.50	90.83	91.67
	250	91.59	91.20	90.40	90.00	91.73
100	150	98.66	96.87	98	95.20	97.20
	300	94.33	92.33	94	90.56	93.56
	450	92.22	89.80	91.55	87.80	90.20
200	400	96.52	95.75	96	93.25	96.5
	600	94.33	92.50	93.83	91.33	93.83
	800	92.40	90.25	92.40	89.55	92.21
500	800	98.75	94.40	98.37	92.38	96.63
	1200	95.83	95.20	94.66	93.80	96.40
	1500	94.33	92.40	93.80	90.87	92.93
1000	1500	-	-	98	-	-
	2500	-	-	94.80	-	-
	3500	-	-	92.59	-	-
1500	2500	-	-	97.59	-	-
	3500	-	-	95.31	-	-
	4000	-	-	94.27	-	-
2000	2500	-	-	98.23	-	-
	3500	-	-	96.85	-	-

MAX-3SAT

n	k	RWS-GSAT	GA-SAT	RNSTS-SAT	RNSTS-SAT+
10	30	100	100	100	100
	50	99.98	99.77	100	100
	80	98.46	99.33	98.5	98.5
30	50	100	100	100	100
	100	99.90	99	99.85	100
	150	98.69	99.33	99.05	99.31
50	100	100	99	100	100
	200	99.35	99	99.25	99.45
	250	98.72	98.8	98.56	99.20
100	150	98	100	99.60	100
	300	99.67	99.66	98.73	99.93
	450	98.22	98.22	98.24	99.56
200	400	99.75	99.66	98.75	99.93
	600	99.67	99.83	98.23	99.83
	800	98.50	99.22	97.38	99.25
500	800	99.80	99.87	99.38	100
	1200	99.50	99.58	98.50	99.90
	1500	98.80	99.46	97.33	99.13
1000	1500	-	99.86	-	-
	2500	-	99.59	-	-
	3500	-	99.11	-	-
1500	2500	-	99.60	-	-
	3500	-	99.48	-	-
	4000	-	99.34	-	-
2000	2500	-	99.87	-	-
	3500	-	99.77	-	-

MAX-4SAT

n	k	RWS-GSAT	GA-SAT	RNSTS-SAT	RNSTS-SAT+
10	30	100	100	100	100
	50	100	100	100	100
	80	100	100	100	100
30	50	100	100	100	100
	100	100	100	100	100
	150	100	99.33	100	100
50	100	100	100	100	100
	200	100	99.50	100	100
	250	100	99.60	99.97	100
100	150	100	100	100	100
	300	100	99.33	100	100
	450	100	99.55	99.90	100
200	400	100	99.66	100	100
	600	100	99.83	99.90	100
	800	100	99.33	99.42	99.87
500	800	100	99.87	100	100
	1200	100	99.91	99.85	100
	1500	100	99.93	99.67	100
1000	1500	-	99.93	-	-
	2500	-	99.92	-	-
	3500	-	99.80	-	-
1500	2500	-	99.96	-	-
	3500	-	99.82	-	-
	4000	-	99.80	-	-
2000	2500	-	99.96	-	-
	3500	-	99.85	-	-

References

[1] P. Alimonti, New local search approximation techniques for maximum generalized satisfiability problems, *Information Processing Letters* **57(3)** (1996), 151–158.

[2] R. Battiti and M. Protasi, Solving MAX-SAT with non-oblivious functions and history based heuristics, Dimacs workshop on satisfiability problem: theory and applications, Rutgers University, March 1996.

[3] R. Battiti and M. Protasi, Approximate algorithms and heuristics for MAX-SAT, *Handbook of combinatorial optimization*, vol. 1, D.Z. Du and P.M. Pardalos Eds, Kluwer Academic Publishers, (1998), 77–148.

[4] B. Borchers and J. Furman, A two phase exact algorithm for MAX-SAT and weighted MAX-SAT problems, (1997).

[5] S.A. Cook, An overview of computational complexity, communications of the ACM, vol. 26, 6 (1983), 401–408.

[6] P. Crescenzi and V. Kann, A compendium of NP-optimization problems, (1997).

[7] L. Davis, *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York, (1991).

[8] H. Drias, A Monte Carlo algorithm for the satisfiability problem, in *Proc of IEA-AIE 98*, LNAI 1415, Springer Verlag, (1998), 159–169.

[9] J. Frank, A Study of genetic algorithms to find Approximate Solutions to Hard 3CNF Problems, in *Golden West International Conference on Artificial Intelligence*, (1994).

[10] G. Gallo, C. Gentile, D. Pretolani and G. Rago, Max Horn SAT and the minimum cut problem in directed hypergraphs, TR 15/95 Dipartimento di informatica, University of Pisa, (1995).

[11] M.R. Garey and D.S. Johnson, *Computers and Intractability*, (Freeman & Co., 1979).

[12] F. Glover, J.P. Kelly and M. Laguna, Genetic algorithms and Tabu search: Hybrids for optimization, *Computers op Res*, vol. 22, 1 (1995), 111–134.

[13] M.X. Goemans and D.P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, *Journal of the ACM* 42 (1995), 1115–1145.

[14] D.E. Goldberg, *Algorithmes génétiques: exporation, expoitation et apprentissage automatique*, Adison-Wesley, France (1994).

[15] J. Gu, Global Optimization for satisfiability (SAT) Problem, *IEEE Transactions on knowledge and data engineering*, vol. 6, 3 (1994), 361–381.

[16] P. Hansen and B. Jaumard, Algorithms for the maximum satisfiability problem, *Computing* 44 (1990), 279–303.

[17] S. Joy, J. Mitchell and B. Borchers, A branch and cut algorithm for MAX-SAT and weighted MAX-SAT, Dimacs series in Discrete Mathematics and Theoretical Computer Science, (1996).

[18] V. Kann and A. Panconesi, Hardness of approximation, *Annotated bibliographies in combinatorial optimization*, John Wiley & Sons, Ltd (1997).

[19] K.J. Lieberherr and E. Specker, Complexity of partial satisfaction, *Journal of the ACM*, vol. 28, 2 (1981), 411–421.

[20] B. Mazure, L. Sais and E. Gregoire, Tabu search for SAT, American Association for Artificial intelligence.

[21] D. Mitchell, B. Selman and H.J. Levesque, Hard and easy distributions of SAT problems, in *Proc of the tenth conference on Artificial Intelligence (AAAI-92)*, San Jose, CA, (July 1992), 440–446.

[22] P. Nobili and A. Sassano, Strengthening Lagrangian bounds for the MAX-SAT problem, TR Universita di Roma, "La Sapienza".

[23] C.H. Papadimitriou and M. Yannakakis, Optimization, approximation and complexity classes, *ACM* (1998), 229–234.

[24] S. Poljak and D. Turzik, A polynomial algorithm for constructing a large bipartite subgraph with an application to a satisfiability problem, *Can. Journal Math*, vol. 34, 3 (1982), 519–524.

[25] B. Selman, H. Levesque and D. Mitchell, A New method for solving hard satisfiability problems, in *Proc of the tenth national conference on Artificial Intelligence (AAAI-92)*, San Jose, CA, (1992), 440–446.

[26] B. Selman et al, Local search strategies for satisfiability testing, Dimacs challenge on cliques, coloring and satisfiability, (Oct 1993).

[27] B. Selman, H.A. Kautz and B. Cohen, Noise strategies for improving local search, in *Proc of the national conference on Artificial Intelligence (AAAI-94)*, Seattle, WA, (July 1994).

[28] M. Yannakakis, On the approximation of maximum satisfiability, *Journal of Algorithms* 17 (1994), 475–502.