Abstract. The satisfiability problem or SAT for short and many of its variants have been widely and constantly studied these last two decades. Competitive general purpose algorithms based on meta-heuristics like genetic algorithms, taboo search and scatter search are known for this problem. This paper introduces the ant colonies approach for the maximum weighted satisfiability problem, namely MAX-W-SAT. This meta-heuristic inspired from the real cooperative behavior of ant colonies searching for food has been proposed and then improved recently to solve hard combinatorial optimization problems. We describe an ant colonies algorithm for MAX-W-SAT called AC-SAT and provide an overview of the results of the empirical tests performed on the hard Johnson benchmark. A comparative study of the algorithm with well known procedures for MAX-W-SAT is done and shows that AC-SAT outperforms the other evolutionary meta-heuristics especially the scatter search, which has been developed very recently.

Keywords. Evolutionary algorithms, optimization, ant colonies, maximum weighted satisfiability, cooperative agents

1 Introduction

Many NP-complete problems are better solved now than in the past. One undeniable reason of the improvement in solutions quality and running time is the development of an important number of new powerful meta-heuristics. If the problems sizes have been very limited in getting a reasonable execution time in the nearest past, larger ones are more manageable nowadays. They have increased in a tremendous way mostly because of the competition of new approaches and the recent physical machines performances.

The algorithm A* has marked the beginning of the introduction of heuristics in solving complex problems. Since then a plethora of meta-heuristics approaches often inspired from natural phenomena have been developed. They have contributed greatly in problem solving since they are simple for use in most situations. Simulated annealing, genetic algorithms, taboo search and scatter search are examples of these methods.

The ant colonies or AC for short is another meta-heuristic mimicking the real behavior of ants colonies when searching for food. The approach has gained a large reputation since it has solved with success many combinatorial optimization problems like the traveling salesman problem or TSP[8], the quadratic assignment problem or QAP[13], the vehicle routing problem or VRP[4] and the job shop scheduling problem or JSSP[7].

The maximum weighted satisfiability problem or MAX-W-SAT is an NP-Hard optimization problem. Many meta-heuristics have been tested for this problem and for SAT. We can mention among these works, genetic algorithms[12], taboo search[1,19], simulated annealing[16] and recently scatter search[11]. In this paper, the AC approach is proposed for this problem. An algorithm has been designed and tested on the real-life Johnson problems. Numerical results are compared with those of other well known approaches and in particular with scatter search.

2 The MAX-W-SAT Problem

The satisfiability problem or SAT in abbreviation is a problem of logic. Its importance and reputation come from the fact that logic is a theoretical tool used in problem solving and many other domains.

SAT is defined to answer the question whether there exists an interpretation of variables that satisfies a logical formula written in the setting of propositional or first order calculus. In this work, only propositional formulae are considered, which present yet a complex problem. Since every well formed formula can be converted in a conjunction of disjunctions of literals, an instance of SAT is often presented in this simple form. More precisely, an instance of SAT is a set of clauses, a clause being a disjunction of literals and a literal is defined to be a Boolean variable with or without a negation. If the number of variables is equal to n and the number of clauses equal to k then the pair (n,k) constitutes the problem parameters.

When a valuation of variables satisfies all the clauses of the instance, we say that the instance is satisfiable. Otherwise when no instanciation of variables satisfies all the clauses simultaneously, we say that the data is contradictory. In this situation, another preoccupation arises: find an assignment that satisfies the maximum number of clauses. This problem is known as the maximum satisfiability problem or MAX-SAT. When weights are associated to

clauses, the goal is to find the assignment that maximizes the sum of weights of the clauses that are simultaneously satisfied. The related problem is called maximum weighted satisfiability problem or MAX-W-SAT.

Example of an instance of SAT

$x_1 + -x_3 + x_4$
$x_3 + -x_4 + -x_5$
$-x_2 + x_3 + -x_5$

This is an instance of three clauses having each three literals built over five Boolean variables x_1, x_2, x_3, x_4 and x_5. The plus sign denotes the Boolean disjunction operator while the minus expresses the unary negation connector. This instance is also called 3-SAT because each clause has a length equal to 3, the length is defined to be the number of literals in the clause. In general, a r-SAT instance is a data where all the clauses have a length equal to or less than r.

A great number of research works have been devoted to SAT and its important variants all around the world these last two decades. Results on both the theoretical and practical settings are known but are not sufficient to answer the question whether the class of non polynomial problems coincides with the class of polynomial ones. We know that SAT is NP-complete in general, and that 2-SAT and Horn-SAT (restricted to Horn clauses) are polynomial. However r-SAT for r>= 3 and MAX-SAT are NP-complete. The optimization version of MAX-SAT is NP-hard. On the other hand, exact algorithms like DPP[3,5], many heuristics like John1, John2[18], GSAT[22] and numerous meta-heuristics like GRASP[21], simulated annealing[16], genetic algorithms[12] and taboo search[1,19] have been written for SAT and its variants.

3 The Ant Colonies Optimization

The ant colonies optimization meta-heuristic or ACO has revealed its efficiency for many combinatorial optimization problems. It is based on a simple mechanism of communication between artificial agents imitating the behavior of real ants.

The ants are capable of cooperating in searching in a collective and asynchronous manner an optimal path linking the food source to their nest. They start exploring the food source in a random way. Then, they take their itinerary through a search space according to probabilistic decision rules.

The ants build the solutions in a progressive manner. When an ant finds food, that is when a solution is built, it evaluates partially the obtained solution and deposits on its way back to the nest a hormonal substance called pheromone in order to guide and help the other ants to get straightly to the food source.

A concise translation of the ant colonies behavior can be described by the following general algorithm:

Begin
  Generate a population of ants
  initialize the pheromone
  repeat for each ant until stop condition
  begin
    build the solution using the pheromone
    update the pheromone table
  end
end

4 Solving MAX-W-SAT with ACO

The implementation of the meta-heuristic for our problem requires the design of the following components: the artificial world where the ants live, the solutions or ants representation, the fitness function that evaluates solutions, the probabilistic rules that direct the ants movements and the strategy of updating the pheromone.

4.1 The Artificial World or Search Graph

According to the nature of MAX-W-SAT problem, the search space is a hypercubic graph, G = (V,E) such that:

- $V = \{O,1\}^n$ is a finite set of vertices, each vertex corresponds to a truth assignment of variables, the total number of these vertices is equal to 2^n
- $E = \{(x,y)$ such that x and $y \in V$ and have only one different bit$\}$, the total number of the edges is equal to $n2^{n-1}$

In this graph, some vertices represent the food sources for the ants, they correspond to optimal solutions. Other vertices represent the ants nests and correspond to initial solutions. The artificial ants are supposed to cross a part of the graph in order to find the shortest path existing between initial and optimal solutions. Since a hypercubic graph is a Hamiltonian graph, it allows the ants to reach any vertex from any other one in less than n moves.

The ants store pheromone in the vertices they visit. An auxiliary structure associated with a vertex is necessary to memorize the pheromone quantity deposited by ants.

4.2 The Solutions or Ants Representation

An optimal solution is a vertex denoting an assignment that minimizes the fitness function. Generally in most evolutionary approaches, artificial agents represent problem instantiations that are to be improved towards the optimal solutions. In MAX-W-SAT, a solution is clearly a truth assignment of the instance variables.

For the AC approach, the ants are artificial agents that cooperate in order to emerge the optimal solution. Each ant possesses an internal memory for storing the current solution.

4.3 The Fitness Function

The objective or fitness function measures the solution quality. In SAT, this function counts the number of clauses satisfied by the solution or the weights sum of the clauses satisfied by the solution in MAX-W-SAT. Note that this function makes the only difference in the development of the AC algorithm for both problems.

4.4 The Initial Population

The ants population is initially drawn at random. This population is modified by the ant process that generates another population of agents having better characteristics in terms of solutions quality.

4.5 The Pheromone Structure

The pheromone information is structured as a table of two dimensions, one for the variables and the other for the possible values that can be taken by these variables. Each time the pheromone is updated, it is stored in this table.

4.6 The Improvement Method

Our ants colonies approach is hybridized with an improvement method in order to increase the algorithm performance. The improvement technique is applied to the initial solutions and even to the computed ones to enhance their quality. It is in most situations context-dependent. For our problem, many heuristics exist and can be used for this purpose. We can evoke the following ones: John1, John2[18], local search and G-SAT[22]. The last mentioned one is time consumer, this is the first reason why local search has been chosen. The second argument for this choice is for its simplicity. Our improvement method consists in choosing the best solution in the neighborhood of the current solution, a neighbor is obtained by just flipping a bit. This operation is repeated until no improvement in the evaluation function is observed.

4.7 The Overall Procedure

The process starts generating an artificial ants population and initializing the pheromone. Each ant has to construct a solution using the pheromone information initialized by the process or communicated by all the other ants afterwards. Once a solution is built, it is improved by means of a local search and the pheromone is updated. This phenomena simulates the process of knowledge acquisition and the forgetting of the past.

The whole process is iterated until the optimal solution is found or after a certain number of iterations dictated by the physical machine limits. The ACO procedure or AC-SAT can be summarized as follows:

Procedure AC-SAT;
Begin
Initialize the pheromone table using a heuristic;
Generate *pop-size* ants at random;
Best = any ant representing a solution;
for (*Iter* = 1 **to** *max-Iteration*) **do**
Begin
For each ant *s* **do**

```
            Begin
                s' = build-sol(s);
                (*Build the solution using the pheromone*)
                s'' = Improve the solution s';
                if f(s'') < f(best) then best = s'';
            End;
        Update the pheromone;
    End;
End;
```

Max-iter and *pop-size* are empirical parameters.

Let consider the following instance to better understand the main instructions of the algorithm:

$x_1 + x_3$	3
$x_1 + x_4$	4
$x_2 + x_4$	2
$x_2 + x_4$	1
$-x_1$	2
$-x_2$	3

The pheromone table called *phero* is a matrix of two dimensions. The lines are indexed by the problem variables and the column specifies the positive or negative form of the literal. An entry in the table reports the pheromone quantity brought by the literal. The table is initialized as follows:

$$Phero[x_i,k] = \frac{\sum \text{weight}(c_j)}{\sum \text{weight}(c_k)}$$

where c_k is any clause and c_j a clause containing x_i if k =1and $-x_i$ if k =0. The initialization of the pheromone table for the example is shown in Table 1.

Table 1. Initialization of the pheromone table for the example

	0	1
x_1	2/15	7/15
x_2	3/15	3/15
x_3	0	3/15
x_4	0	7/15

4.8 The Probabilistic Decision Rules

The solution is built by an ant according to the pheromone information. In fact, a certain number of bits of the current solution will be changed according to the following formula:

$$P(x_i=k) = \frac{Phero[x_i,k]}{Phero[x_i,0] + Phero[x_i,1]}$$

That computes the probability that a literal x_i takes the Boolean value k. A random number is generated and compared to this probability to deduce the value of the bit. The procedure that constructs a solution is as follows:

```
Procedure build-sol (s : assignment) : assignment;
Begin
    Let s = (x1, x2, …,xn);
    For j := 1 to max-changes do
        begin
            draw at random a variable xi;
            (* apply probabilistic decision rule *)
            compute P(xi=0);
            generate a random number r such that 0<=r<=1;
            If r <= P(xi=0) then xi =0 else xi =1;
    Return s;
End;
```

The total number of bit changes is denoted by *max-changes* and is set by empirical tests.

4.9 The Pheromone Updating

Our strategy is a delayed pheromone updating, that is it is undertaken once the ants finish their search cycle. The evanescence process is first simulated by decreasing the pheromone values according to the following formula:

$$Phero[x_i,j] = (1-\alpha)\ phero[x_i,j] \quad \text{for } i=1..n \text{ and } j \in \{0,1\}$$

Where $0<\alpha<1$ is the fourth empirical parameter.

The pheromone is then reinforced by the quality of the solution found such that each ant contribution is inversely proportional to the solution evaluation. This pheromone is weighted by the ratio of the difference in quality between the current and the worst solution on the difference in quality between the best and the worst solution.

$$Phero[x_i,j] = (1-\alpha)\ phero[x_i,j] + \alpha \frac{1}{1+f(s)} \times \frac{f(worst)-f(s)}{f(worst)-f(best)}$$

α allows the control of the pheromone quantity added. When it is close to 1, the ants become unstable because of the lack of the pheromone. However when it is close to 0, the solutions do not progress toward the optimal ones.

5 Numerical Results

The procedure AC-SAT has been implemented in C on a Pentium personal computer and numerical tests have been performed on benchmark instances available on the web site given below. These hard problems translating real-life problems have been converted from the Johnson class of the second DIMACS Challenge implementations. The weights of the clauses have been drawn from 1 to 1000 and assigned at random to clauses, the number of clauses being ranging from 800 to 950. The Johnson class namely 'jnh', has been used in many works for testing algorithms performance. It includes three subclasses characterized by the variables number, which is equal to 100 for all instances. The clauses number is

- 800 for the subclasse1: Jnh201 ..Jnh220
- 850 for the subclass 2 : Jnh01..jnh19
- 900 for the subclass 3 : Jnh301..Jnh310

Each subclass contains instances that are satisfied or not. On each instance, 10 executions have been undertaken. The solutions quality as well as the running time have been considered as performance criteria.

5.1 Parameters Setting

Preliminary tests have been carried out in order to fix the key parameters of the AC-SAT algorithm. Figure 1 shows an example of the results of the tests done on some Johnson problems for setting the parameter *max-iter*. Identical tests

have been performed on all the other instances and for all the other parameters. Table 2 summarizes the parameters values obtained after theses extensive experiments.

5.2 Performance Comparison

Comparison of AC-SAT with SS-SAT, GRASP and the optimal solution is done. GRASP is a parallel greedy algorithm proposed for solving MAX-W-SAT[21]. It has been considered in this comparison because of its high efficiency. SS-SAT is another evolutionary algorithm based on scatter search approach and recently developed [11].

Table 3 contains for each Johnson problem, the optimal solution and the best solution found respectively by AC-SAT, SS-SAT and GRASP. The bold numbers represent the best values found. Their number is clearly more important for AC-SAT. this fact translates that AC-SAT outperforms SS-SAT and also GRASP for most instances of problems. Besides, the most important observation is that for most instances, solutions provided by AC-SAT are equal to the optimal solutions and for the other cases, the difference between both solutions is almost null.

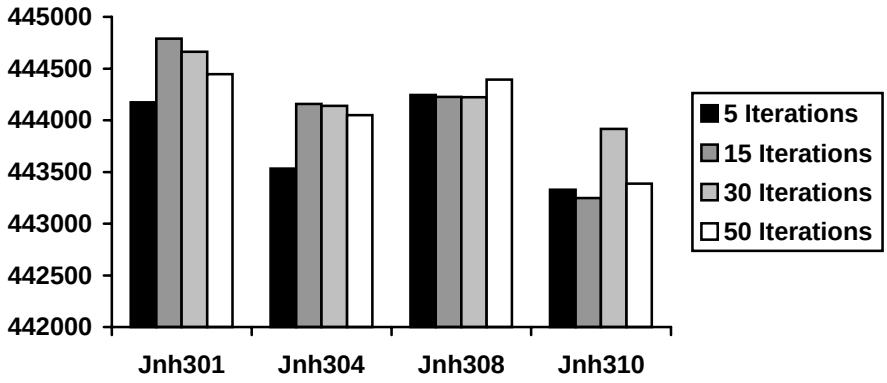

figure1. The results of tests for setting the parameter *max-iter*

Table 2. Empirical parameters values for AC-SAT

Pop-size	6
α	0.1
maxchanges	50
Max-iter	30

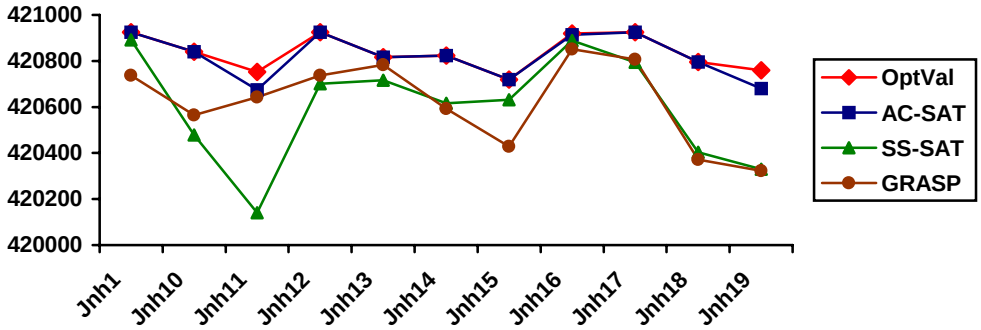

Figure 2. A schematic comparison between ac-sat, ss-sat, grasp and the optimal solution for the subclass jnh01-Jnh19

http://www.research.att.com/~mgcr/data/index.html

The execution time of AC-SAT is also very competitive and more interesting than that of SS-SAT. A schematic view of these results is shown through the curves of Figure 2, Figure 3 and Figure 4. The same remark can be done on the performance of AC-SAT.

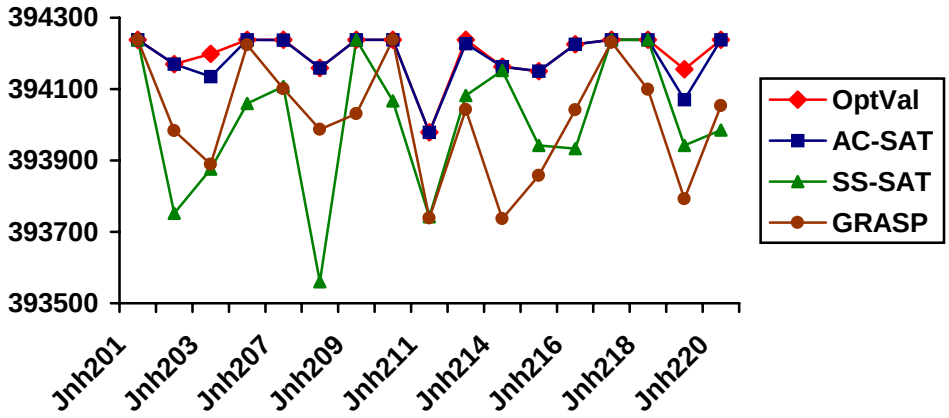

Figure 3. A schematic comparison between ac-sat, ss-sat, grasp and the optimal solution for the subclass jnh201-jnh220

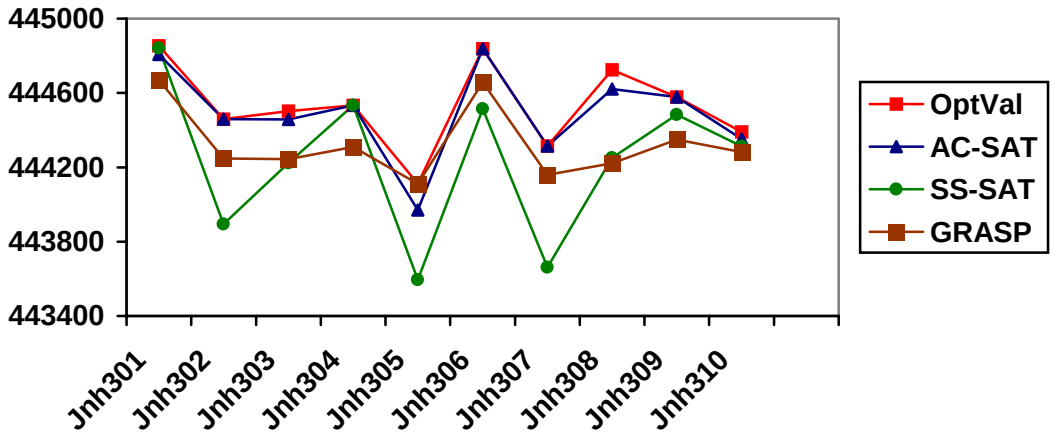

Figure 4. A schematic comparison between ac-sat, ss-sat, grasp and the optimal solution

6 Conclusion

In this paper, an ant colonies algorithm called AC-SAT has been designed and tested for the maximum satisfiability problem. Through the empirical results, AC-SAT outperforms the scatter search algorithm recently developed and the best version of GRASP. It also finds the optimal solutions for the majority of Johnson problems. For the others, the solution obtained is very close to the optimal one.

The robustness of the AC method relies on the design of the solution construction procedure based on probabilistic decision rules, the pheromone updating strategy and its hybridization with an improvement technique. For the time being, we are studying and experimenting the parallelization of the AC approach for MAX-W-SAT in order to achieve

increase in performance in terms of solutions quality and running time. The results will be communicated in the near future.

Table 3. Experimental results comparing AC-SAT, SS-SAT, GRASP and the optimal solutions

data	Opt sol	**Ac-sat**	ac-sat time (s)	ss-sat	ss-sat time (s)	Best of grasp
Jnh01	**420925**	**420925**	107.15	420892	203.99	420737
Jnh10	**420840**	**420840**	179.25	420479	901.47	420565
Jnh11	420753	**420674**	200.15	420141	439.82	420642
Jnh12	**420925**	**420925**	124.65	420701	250.65	420737
Jnh13	**420816**	**420816**	192.94	420716	930.12	420783
Jnh14	**420824**	**420824**	237.23	420616	298.22	420592
Jnh15	**420719**	**420719**	240.26	420632	624.63	420429
Jnh16	420919	**420914**	297.18	420889	716.51	420851
Jnh17	**420925**	**420925**	222.71	420794	401.95	420807
Jnh18	**420795**	**420795**	210.71	420404	129.64	420372
Jnh19	420759	**420680**	204.75	420330	403.10	420323
Jnh201	**394238**	**394238**	162.56	394238	161.72	394238
Jnh202	**394170**	**394170**	184.26	393752	269.99	393983
Jnh203	394199	**394135**	191.78	393876	294.10	393889
Jnh205	**294238**	**394238**	283.75	394060	203.37	394224
Jnh207	394238	**394237**	122.36	394107	313.58	394101
Jnh208	**394159**	**394159**	186.71	393560	137.19	393987
Jnh209	**394238**	**394238**	186.20	394238	395.38	394031
Jnh210	**394238**	**394238**	219.12	394067	394.46	394238
Jnh211	**393979**	**393979**	187.56	393742	408.04	393739
Jnh212	394238	**394227**	188.78	394082	758.00	394043
Jnh214	**394163**	**394163**	193.67	394152	1159.15	393737
Jnh215	**394150**	**394150**	184.37	393942	202.51	393858
Jnh216	**394226**	**394226**	189.13	393933	391.10	394042
Jnh217	**394238**	**394238**	144.6	394238	799.40	394232
Jnh218	**394238**	**394238**	217.12	394238	755.92	394099
Jnh219	394156	**394070**	130.45	393942	236.38	393792
Jnh220	**394238**	**394238**	166.23	393985	430.44	394053
Jnh301	444854	444807	204.4	**444842**	1267.63	444670
Jnh302	**444459**	**444459**	260.2	443895	698.77	444248
Jnh303	444503	**444457**	222.32	444223	437.91	444244
Jnh304	**444533**	**444533**	310.11	444533	1175.98	444310
Jnh305	444112	443970	330.28	443594	463.33	**444112**
Jnh306	**444838**	**444838**	321.46	444515	604.53	444658
Jnh307	**444314**	**444314**	231.45	443662	128.35	444159
Jnh308	444724	**444621**	116.15	444250	270.98	444222
Jnh309	**444578**	**444578**	211.02	444483	662.58	444349
Jnh310	444391	**444353**	247.29	444313	132.63	444282

References

1. R. Battiti, M. Protasi, Reactive seach, a history-based heuristics for MAX-SAT', *Journal of experimental algorithmics*, 1997

2. V.D. Cung, T. Mautor, P. Michelon and A. Tavares, A Scatter Search based Approach for the Quadratic Assignment Problem, research Report N° 96/037 Versailles university, (1996)
3. B. Borchers and J. Furman, 'A two phase exact algorithm for MAX-SAT and weighted MAX-SAT problems' (1997)

4. B. Bullnheimer, R.F. Hartl and C. Strauss, 'Applying the ant system to the vehicle routing problem', in 2nd Metaheuristics Int Conf, MIC'97, Sophia-Antipolis, France, (July 1997)
5. M. Davis and H. Putnam, 'A Computing Procedure for Quantification Theory', *Journal of the ACM*, 7, 201-215, (1960)
6. M. Dorigo, G.D. Caro and L.M. Gambardella, 'Ant Algorithms for discrete optimization', Tech Rep iridia/98-10, Université libre de Bruxelles, 1998
7. A. Colorni, M. Dorigo, V. Maniezzo and M. Trubian, 'Ant system for Job-Shop Scheduling', Jorbel, 34(1), 39-53, (1994)
8. M. Dorigo and L.M. Gambardella, 'Ant Algorithms for the Traveling Salesman Problem', Biosystems, 43, 73-81, (1997)
9. M. Dorigo and G.D. 'Caro, Ant Colony Optimization : A New meta heuristic', IEEE, (1999)
10. ----, 'Randomness in Heuristics : An Experimental Investigation for the Maximum Satisfiability Problem'**,** *Journal of Combinatorial Mathematics and Computational Complexity***,** vol 38, 209-223, (August 2001)
11. --- , 'Scatter search with random walk strategy for solving hard MAX-W-SAT problems'**,** in proc of IEA-AIE'2001, Lectures Notes in Computer Science, LNAI 2070, Springer , Budapest, 35-44, (June 2001)
12. J. Frank, 'A Study of genetic algorithms to find Approximate Solutions to Hard 3CNF Problems', in proc of Golden West International Conference on Artificial Intelligence, (1994)
13. L.M. Gambardella, E. Taillard and M. Dorigo, 'Ant Algorithms for the QAP', Technical Report 97-4, IDSIA, Lugano, Switzerland, (1997)
14. M.R. Garey and D.S. Johnson, *Computers and intractability, a guide to the theory of NP-completeness*, W.H. Freeman and Co, San Francisco, 1979
15. F. Glover, M. Laguna and R. Marti, 'scatter search', internet document, (2000)
16. P. Hansen and B. Jaumard, 'Algorithms for the Maximum Satisfiability' , computing 44, 279-303, (1990)
17. M. Laguna, Scatter Search, internet document, (1999)
18. K.J. Lieberherr and E. Specker, 'Complexity of partial satisfaction', *Journal of the ACM*, vol 28, 2, 411-421, (1981)
19. B. Mazure, L. Sais and E. Grégoire, 'Taboo Search for SAT', in proc of AAAI, (1997).
20. P. Mills and E. Tsang, 'Solving the MAX-SAT problem using Guided Local Search', technical Report CSM-327, (1999)
21. M.G. Resende, L.S. Pitsoulis and P.M. Pardalos, 'Approximate Solution of Weighted MAX-SAT Problems using GRASP', in Dimacs series on discrete mathematics and theoretical computer Science vol 35, 393-405, (1997)
22. B. Selman, H.A. Kautz and B. Cohen, 'Local Search Strategies for Satisfiability Testing, in cliques, coloring and satisfiability', second DIMACS implementation challenge, vol 26, D.S. Johnson and M.A. Tricks eds., 521-531, (1996)