Asynchronous Team Algorithms for Boolean Satisfiability

Carlos Rodríguez
Politechnical School
National University of Asuncion
San Lorenzo, P.O. Box 2169
Paraguay
Tel: +595-(21)-585-599
crodriguez@cnc.una.py

Marcos Villagra
Science and Technology School-DEI
Catholic University of Asuncion
Asuncion, P.O. Box 1683,
Paraguay
Tel: +595-(21)-334-650
marcos_villagra@uca.edu.py

Benjamín Barán
Politechnical School
National University of Asuncion
San Lorenzo, P. O. Box 2169,
Paraguay
Tel: +595-(21)-585-599
bbaran@pol.una.py

ABSTRACT
The Boolean Satisfiability Problem or SAT is one of the most important problems in computer science. Nowadays, there are different types of algorithms to solve instances with thousands of variables, and much research is being carried out looking for more efficient algorithms to solve larger and harder instances. This work proposes the utilization of a Team Algorithm (TA) strategy combining different local search algorithms for SAT as WalkSAT, R-Novelty+, Adaptive Novelty+, RSAPS, IROTS and SAMD. TAs allow the combination of different algorithms that interact with each other searching for a good global solution. Experimental results show that the proposed TA is a general strategy capable of obtaining promising results for a variety of instances.

Keywords
SAT, MAX-SAT, Team Algorithms, Asynchronous Team, Local Search.

1. INTRODUCTION
Boolean Satisfiability or SAT is the problem of finding an assignment of values to Boolean variables for a propositional logic formula, in such a way that it is evaluated as true. SAT is a $\mathcal{NP}$-Complete problem and it was the first proved to belong to this class [2], and as such it is considered to be one of the most important problems in computer science.

There are many algorithms for SAT (usually known as solvers) for different varieties of the problem. These algorithms are divided in two strategies: the Davis-Putnam method (DP) and Local Search [3]. Algorithms based on DP are backtracking search algorithms with heuristics to quickly cut-off the search tree. On the other hand, local search techniques for SAT work with complete assignments and there is no guaranty of finding satisfying assignments if one exists or proving the unsatisfiability of the formula.

Since the appearing of SAT, there has been a lot of progress in the development of solvers. Nowadays, there are many methods that turned out to be very fast to solve hard instances of the problem, and there is a great demand for better algorithms that could solve even larger and harder problems. Sequential solvers dominate the field. This is mainly justified by the performance improvements in modern sequential computers [11]. However, there is an important challenge concerning parallel solvers, since there are SAT instances that are out of the reach of state-of-the-art sequential algorithms. In practice, most of the sequential algorithms can be mapped to parallel computers resulting in parallel algorithms for SAT. This is the approach used in most parallel implementations [11]. There are parallel implementations for different multiprocessing hardware architectures, as there are also distributed implementations. Most of these implementations are based on selecting some efficient sequential algorithm and parallelizing it according to the target architecture [11].

On the other hand, this work proposes the application of a *Team Algorithm* (TA) strategy [1] for SAT. This bio-inspired strategy is based on the accomplishment of animal societies working as a team, as ants, bees, and other insects which cooperatively achieve considerable objectives using simple mechanisms. TAs allow the combination of different algorithms that interact with each other in the process of searching for the global solution to a problem. This strategy can be naturally implemented in parallel, assigning different parts of a problem to processors of an asynchronous distributed system, like a computer network, in which case it is known as *Asynchronous Team* (A-Team) [1]. TAs have been applied to many important and difficult problems such as multiobjective optimization, topological optimization of reliable networks, hydroelectric optimization, to name a few.

The remainder of this work is organized as follows. Section 2 presents the SAT problem. TAs and A-Teams are introduced in section 3. In section 4, the main proposal of this work is presented. Experimental results proving the advantages of using TAs are given in section 5. Conclusions and future work are left for section 6.

2. THE SAT PROBLEM
SAT is defined by a set of Boolean variables $X=\{x_1,...,x_n\}$ and a Boolean formula or sentence $\phi:\{0, 1\}^n \rightarrow \{0, 1\}$ where n denotes the number of variables. The objective is to find a variable assignment $m=\langle x_1,...,x_n\rangle \in \{0, 1\}^n$ (i.e., a model) where $\phi(m)=1$. The formula is called *satisfiable* if such a model exists, and *unsatisfiable* otherwise. A literal is a variable (also called atom or symbol) or its negation. A clause is a disjunction of literals, i.e., literals connected by an $\vee$ operator. The formula ϕ is in conjunctive normal form (CNF) if $\phi(m)=c_1(m)\wedge...\wedge c_p(m)$, where each c_i is a clause and p is the number of clauses. The family of k-CNF sentences has exactly k literals per clause. SAT can be assumed having CNF formulae without loss of generality, and k-

BIONETICS'07, December 10-13, 2007, Budapest, Hungary.

SAT only contains sentences in *k*-CNF. While 2-SAT is solvable in polynomial time, *k*-SAT is $\mathcal{NP}$-complete for $k \geq 3$.

MAX-SAT is an optimization variant of SAT. Given a set of clauses, MAX-SAT is the problem of finding a model *m* that maximizes the number of satisfied clauses *f(m)*. In *weighted* MAX-SAT each clause c_i has a weight w_i assigned to it, while *unweighted* MAX-SAT uses $w_i = 1$ for all *i*. The following equation defines the MAX-SAT objective function:

$$f(m)=\sum_{j\in\{1,\ldots,p\}} w_j \cdot c_j(m) \quad (1)$$

where $c_i(m) = 1$ if the clause c_i is satisfied under the assignment *m*, and 0 otherwise. Thus, when searching for a solution to MAX-SAT, one is actually looking for models to SAT formulae. Therefore, solutions refer to models and vice versa. There are only some differences between particular cases of SAT and MAX-SAT, where MAX-*k*-SAT is $\mathcal{NP}$-hard for $k \geq 2$.

3. TEAM ALGORITHMS

It is well known that a given algorithm may be better than others in a subset of problems; however, the "No Free Lunch" theorems state that any two good algorithms are equivalent when their performance is averaged across all possible problems. In SAT, it is mainly due to the structures in instances what penalizes some kind of algorithms while favoring others [3]. As a consequence, it is common that, when searching for solutions to some subsets of the problem, the selection of algorithms is restricted to the most suited and convenient [3]. Even though, there are cases in which the utilization of some algorithms for solving a specific instance of the problem gives an unacceptable performance and poor execution times [3].

There are many SAT solvers available for public access [13]. Therefore, the question about which are the best alternatives and the possibility of combining them in some way to face hard problems emerges. There is a technique that combines different algorithms that interact with each other in the search process. This strategy is known as Team Algorithms (TA) [1]. Solving problems with TAs can be naturally implemented in parallel, assigning different parts of the problem to processors in an asynchronous distributed system, such as a computer network, in which case it is known as Asynchronous Team (A-Team) [1].

In a distributed system, an A-Team consists in a collection of processors to which sub-problems are assigned in such a way that each processor executes the most suited algorithm for the given sub-problem. An A-Team is always paying attention to the progress in the resolution of the sub-problems, in order to find a global solution to the problem [1]. It is important to notice that the main feature of A-Teams (with respect to TAs) is asynchronism, i.e., different processors communicate with each other avoiding any kind of central clock or blocking communication protocol. In this manner, it is assured that each processor of the network is used efficiently, avoiding idle times and computational resource wastes.

4. A-TEAM FOR SAT

This work proposes the application of the A-Team strategy with overlapping [1] to SAT. Since there are a great number of algorithms with very efficient heuristics, there is a valuable foundation to build an A-Team with these methods.

The A-Team for SAT is proposed as a Master/Slave model. The Master is in charge of coordinating the activities of the slaves, which effectively solve the problem. It is also responsible for launching the slave processes with a default algorithm, indicating the problem to solve. Each Slave executes the algorithm assigned to it, and reports the results asynchronously to the Master delivering the solution found so far. Solutions obtained from the Slaves are stored in the Master process, in a pool of global solutions with an arbitrary maximum size. The A-Team is executed until the number of unsatisfied clauses equals zero (i.e., a model was found), or until a maximum execution time is reached reporting the best approximation and the algorithm which found it.

The model used by the A-Team is the overlapping solution with independent runs, i.e., each Slave will solve the whole problem independently and report the results to the Master. Also, a technique to promote synergy among algorithms is implemented, in which the Master starts sharing solutions from the global pool among the algorithms of the A-Team after some fixed execution time. This scheme is used to refine the solutions of the algorithms.

The algorithms implemented for the Slave processes are all based in local search techniques. Most of these algorithms are available for public access, and they have been optimized in various stages with very efficient executions. This work proposes to take advantage of these implementations, and adapt them to the framework of the proposed A-Team.

A total of six local search algorithms were selected for the A-Team, which merges traditional algorithms of the state of the art. The implementations of these algorithms where selected from UBCSAT [13]. For this work, the following were selected:

- WalkSAT [10].
- R-Novelty+ [6].
- Adaptive Novelty+ [5].
- RSAPS [8].
- IROTS [12].
- SAMD [4].

Since all these algorithms are prepared to solve MAX-SAT, the resulting A-Team will be able (in principle) to work with any type of instance, obtaining this way a general strategy. This way, the proposed A-Team constitutes a framework for solving MAX-SAT problems using algorithms based on local search strategies. Due to the diversity of algorithms composing the A-Team, it is expected that the proposed parallel solver will be able to solve any kind of SAT instance, regarding the structure present in the problem. This is an important advantage, especially when there is a need for exploring new instances with unknown characteristics.

5. EXPERIMENTAL RESULTS

5.1 Environment

The environment in which the experiments were performed consists of a computer network with seven personal computers, each having 3.0 GHz Intel Pentium processors, 512 MB of RAM and 100 Mbps Network Interface Card (NIC). The computers were connected in a star topology using a switch of 100 Mbps forming a Local Area Network (LAN). The operating system running in each computer was Linux Fedora Core4 with a kernel version 2.6.11-1.

The algorithms were implemented with ANSI C programming language, using a GCC version 4.0.0 compiler. Libraries of

Parallel Virtual Machine (PVM) version 3.4 were used for communications.

To compare the performance of the different algorithms, a benchmark with a variety of instances from SATLIB [7] was chosen, considering mainly hard instances as detailed in Table 1.

Table 1. Instances used for experiments.

ID	Instance	#Var	#Cla	Sat
1	par32-1-c	1315	5254	S
2	ewddr2-10by5-1	21800	118607	S
3	4blocsb	540	34199	S
4	bf1355-075	2180	6778	S
5	ssa6288-047	10410	34238	N
6	g250.15	3750	233965	S
7	hanoi5	1931	14468	S
8	qg5-13	2197	125464	N
9	bmc-ibm-11	32109	150027	S
10	bmc-galileo-9	63624	326999	S
11	ais12	265	5666	S
12	logistics.d	4713	21991	S
13	ii32e5	522	11363	S
14	bw_large.d	6325	131973	S
15	g250.29	7250	454622	S
16	par16-5	1015	3358	S
17	3bitadd_31	8432	31310	S
18	f3200	3200	13600	S
19	f6400	6400	27136	S
20	jnh310	100	900	N

The experiment consisted in executing the algorithms mentioned above, as well as the A-Team combining all of them. For each instance, the A-Team was executed ten times for a period of 300 seconds for each run. It was also established that the Master process would start distributing solutions from the global pool at 120 seconds since the beginning of the run.

To compare the performance of the A-Team with the six different algorithms (each one on a different Slave process), parallel versions of WalkSAT [10], RNovelty+ [6] and RSAPS [8] were implemented. For these implementations, the same framework of the A-Team was used, running the same algorithm on each slave process but avoiding the solution sharing scheme. This allowed the comparison of the effect of combining different algorithms in the process of solving a given problem with respect to the different algorithms individually parallelized.

The configuration parameters of the algorithms which were used for the runs are the ones predefined in UBCSAT [13]. A restart parameter of 10^6 *flips* was established as a general parameter for all algorithms. When a model is found or when this number of flips is reached, the best solutions found by the Slaves are sent to the Master which keeps a global solutions pool. The maximum size for this pool is fixed to Z=50 solutions.

5.2 Comparative Results

Table 2 presents the results of several runs. The following metrics were used for comparisons purposes:

- *Mean of unsatisfied clauses (mean):* This is the mean of unsatisfied clauses obtained as results of the runs.
- *Success Rate (SR):* This is the proportion of runs in which a model was found ($0 \leq SR \leq 1$).
- *Mean Execution Time (sec.):* This is the mean execution time measured in seconds, for those runs in which a model was found. For this metric, ε is used to denote an execution time lower than 0.005 seconds.

Among the presented metrics, *sec* and *SR* are used only in those cases in which a model was found at least once. In all other cases, the minimum number of unsatisfied clauses is reported instead.

The results presented in Table 2 show that the proposed A-Team has obtained a very competitive performance with respect to the other algorithms in terms of the *mean* metric. In most of the instances the A-Team has achieved the lowest *mean* and, particularly, it can be noticed that in instances 9 (bmc-ibm-11), 14 (bw_large.d), 15 (g250.29) and 16 (par16-5) the *mean* obtained by the A-Team has defeated all other algorithms. Moreover, it is important to notice that in those instances in which the A-Team was defeated in terms of *mean* (1 (par32-1-c), 5 (ssa6288-047), 8 (qg5-13) and 19 (f6400)), the difference is relatively very small considering the number of clauses. Table 3 presents a ranking of the strategies considering the *mean* metric across all instances. Here, it can be seen that the A-Team has achieved the best performance in average.

6. CONCLUSIONS AND FUTURE WORK

This work presented a MAX-SAT solver based on the Asynchronous Team Algorithms (A-Team) strategy, a bio-inspired method based on the accomplishment of animal societies working as a team, as ants, bees, and other insects which cooperatively achieve considerable objectives. The algorithms used are all based on local search methods, which were selected among representative state-of-the-art algorithms and traditional methods for MAX-SAT as WalkSAT, R-Novelty+, Adaptive Novelty+, RSAPS, IROTS and SAMD. The A-Team was implemented using the Master/Slave model, promoting this way a synergy effect among algorithms by means of global results sharing during the execution of the slaves.

Experimental results showed that the proposed A-Team has the ability of solving efficiently a great variety of instances. Also, in the majority of the studied instances, improved solutions were obtained with respect to the parallel versions of the local search methods. As it can be noticed in the experimental results, the proposed A-Team can be used as a general strategy to face a great variety of instances obtaining very competitive results. Therefore, the A-Team turns out to be a very useful strategy when there is a need for exploring new instances with unknown characteristics.

Directions for future work include the implementation of a strategy for dynamically replacing algorithms, which may include a scheme to rank those algorithms. This way, it would be possible to replace different algorithms with lower performance with those algorithms previously selected for their outstanding performance. Also, an A-Team with a larger number of local search algorithms mixing them with metaheuristics (e.g., GASAT [9], Ant Colony Optimization [14]) could be implemented.

Table 2. Experimental results.

ID	A-Team			WalkSAT			RNovelty+			RSAPS		
	mean	**SR**	**sec.**	***mean***	**SR**	**sec.**	***mean***	**SR**	**sec.**	***mean***	**SR**	**sec.**
1	6.2	(5 clauses)		101.8	(93 clauses)		5.7	(5 clauses)		**5.6**	(5 clauses)	
2	**0**	1	ε	0.1	0.9	166.5	15.8	(14 clauses)		**0**	1	ε
3	**0**	1	ε	**0**	1	ε	0	1	27.34	**0**	1	ε
4	**1**	(1 clause)		**1**	(1 clause)		**1**	(1 clause)		**1**	(1 clause)	
5	44,6	(37 clauses)		102.6	(92 clauses)		**38.7**	(31 clauses)		47.6	(35 clauses)	
6	**0**	1	ε	**0**	1	ε	**0**	1	ε	**0**	1	ε
7	**1**	(1 clause)		2.8	(2 clauses)		**1**	(1 clause)		**1**	(1 clause)	
8	17.7	(14 clauses)		262.2	(257 clauses)		**15.2**	(14 clauses)		66.3	(51 clauses)	
9	**357**	(322 clauses)		1028.4	(1014 clauses)		545.7	(538 clauses)		2676.2	(1524 clauses)	
10	37.7	(28 clauses)		**30.1**	(24 clauses)		1269.8	(1235 clauses)		5229	(4465 clauses)	
11	**0**	1	ε	0	1	3.51	0	1	22.98	**0**	1	ε
12	**0**	1	ε	**0**	1	ε	0	1	17.21	**0**	1	ε
13	**0**	1	ε	3	1	0.01	**0**	1	ε	**0**	1	ε
14	**0**	1	59.06	15.4	(10 clauses)		2.9	(1 clause)		0.2	0.8	137.95
15	**0**	1	11.45	31.6	(30 clauses)		16.3	(14 clauses)		9.9	(8 clauses)	
16	**0.9**	0.1	148.4	16.2	(10 clauses)		0.9	0.1	261.23	0.9	0.1	182.3
17	**0**	1	ε	**0**	1	ε	4.7	(2 clauses)		9.2	(7 clauses)	
18	0	1	25.64	**0**	1	3.08	0	1	9.70	14.2	(9 clauses)	
19	4.4	(3 clauses)		3.5	(3 clauses)		**3.2**	(2 clauses)		45.3	(39 clauses)	
20	**3**	(3 clauses)		**3**	(3 clauses)		**3**	(3 clauses)		**3**	(3 clauses)	

Table 3. Ranking of strategies by *mean* metric.

Ranking	Estrategy	Average of *mean* metric
1	A-Team	23.68
2	WalkSAT	80.14
3	R-Novelty+	96.20
4	RSAPS	405.47

7. REFERENCES

[1] B. Barán, E. Kaszkurewicz, and A. Bhaya. Parallel Asynchronous Team Algorithms - Convergence and Performance Analysis. *IEEE Transactions on Parallel and Distributed Systems,* 7:677-688, 1996.

[2] S. Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of The 3th Anual ACM Symposium of the Theory of Computing,* pages 151-158. ACM, 1971.

[3] J. Gu, P. Purdom, J. Franco, and B. Wah. Algorithms for the Satisfiability (SAT) Problem: A Survey. In *DIMACS Series on Discrete Mathematics and Theoretical Computer Science,* 1997.

[4] P. Hansen, and B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing,* 44(4):279-303, 1990.

[5] H. Hoos, H. An adaptive noise mechanism for WalkSAT. In *Proceedings of the 18th National Conference in Artificial Intelligence (AAAI-02),* pages 655-660. 2002.

[6] H. Hoos. On the run-time behaviour of stochastic local search algorithms for SAT. In *Proceedings of the 16th National Conference on Artificial Intelligence(AAAI-99)*, pages 661-666. 1999.

[7] H. Hoos, and T. Stützle. SATLIB: An Online Resource for Research on SAT. In *Proceedings of SAT 2000*, pages 283-292. IOS Press, 2000.

[8] F. Hutter, D. Tompkins, and H. Hoos. Scaling and probabilistic smoothing: Efficient dynamic local search for SAT. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP-02),* pages 233-248. 2002.

[9] F. Lardeaux, and F. Saubion. GASAT: A Genetic Local Search Algorithm for the Satisfiability Problem. *Evolutionary Computation,* 14(2):223-253, 2006.

[10] B. Selman, H. Kautz, and B. Cohen. Local Search Strategies for Satisfiability Testing. In *Proceedings of 2nd DIMACS Challenge on Cliques, Coloring and Satisfiability.* 1993.

[11] D. Singer. Parallel Resolution of the Satisfiability Problem. A Survey. *Technical Report: LITA 2007-101.* Université Paul Verlaine - Metz. 2007.

[12] K. Smyth, H. Hoos, and T. Stützle. Iterated Robust Tabu Search for MAX-SAT. In *Proceedings of the 16th Canadian Conference on Artificial Intelligence (AI 2003),* 129-144. 2003.

[13] D. Tompkins, and H. Hoos. UBCSAT: An Implementation and Experimentation Environment for SLS Algorithms for SAT and MAX-SAT. In *7th International Conference on Theory and Applications of Satisfiability Testing (SAT2004).* 2004.

[14] M. Villagra, and B. Barán. Ant Colony Optimization with Adaptive Fitness Function for Satisfiability Testing. *Logic, Language, Information and Computation,* volume 4576 of *Lecture Notes in Computer Science (LNCS)*, pages 352-361. Springer, 2007.