A Tabu History driven Crossover Operator Design for Memetic Algorithm applied to Max-2SAT-Problems

M. Borschbach and A. Exeler
Institute for Computer Science
Einsteinstr. 62, University of Münster, D-48149 Germany
Markus.Borschbach@uni-muenster.de

ABSTRACT

The solution for the Max-2SAT is the starting point for a selection of these strategies by a brief review. Moreover, a memetic algorithm for Max-2SAT problems based on a specific crossover operator and an improved tabu search stage is presented. Simulation performed on several instances of Max-2SAT reference problems are used to evaluate the different memetic algorithm strategies applied in our approach and to compare it to the computational complexity of existing local search solutions.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods
General Terms: Design, Verification.
Keywords: Max-2SAT, Memetic Algorithms, Tabu Search

1. INTRODUCTION

The satisfiability problem (SAT) is one of the most studied problems in computer science and since SAT is NP-complete [6], it doesn't seem to be promising to find a complete algorithm that performs well on all sizes of problem instances, e.g. scheduling [13], graph theory, automated reasoning [1] and other domains like VLSI [3]. Its optimization variant Max-SAT consists of identifying a variable assignment which maximizes the number of satisfied clauses. Among the best exact algorithms is MaxSatz, which won most of the benchmarks for unweighted propositional formula in conjunctive normal form at the Max-SAT evaluation competition 2006. Max-2SAT, where the number of literals in each clause is limited by two, is NP-hard [9]. In contrast to this, 2SAT can be solved in linear time [2]. The Memetic Algorithm with Tabu History driven Crossover (MATHiC) proposed in this paper, consists of the following six evolutionary phases. The initial population is modified by initial distribution heuristics, added assignments of solutions with the highest number of satisfied clauses known in advance, selection constraints and a local search applied to each individual (phase one). Moreover, a tabu history local search (THLS) is applied in phase three to each individual generated by the recombination operator and in phase five by each individual generated by the mutation operator. In phase two each recombination (Tabu History Crossover (THC)) is driven by a specific problem constraint operator setup and in phase four each mutation also. Each generation is completed by an additional selection mechanism (six).

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-130-9/08/07.

2. OUTLINE & EVALUATION

The efficiency and operation of phases two and four (refer to section 1) of the proposed approach MATHiC (see [4] for algorithm details) are outlined in the following. The crossover operator used is very similar to the operators presented in [11] and in [5] and a simplified version will be used for empirical comparison. The advances made here for THC1 are as follows. Each individual got its own tabu history. At the start of the crossover operator the successor randomly inherits one of its predecessor tabu histories and every flip[1] made during the execution of the crossover operator is recorded in this inherited tabu history. Also a refined version of the selection strategy for the variable to flip is used. If a clause is unsatisfied in both predecessors simultaneously, it is only flipped if the improvement is maximized *and* it is the least recently flipped variable of the clause. If it maximizes the improvement but is not the least recently flipped variable, it is only flipped with a probability p and the least recently flipped variable is flipped with the probability $1-p$. In all simulation onsets in the following p is set to 0.5. The advanced crossover (THC2) uses an improved local search operator based on the flip history to guide the search on each individual. The last tl elements of the history list are declared *tabu* during the local search phase. The algorithm selects an allowed[2] variable which offers the best improvement. If there is more than one variable allowed, which offer a maximum improvement, the least recently flipped one is chosen. If there is a tabu variable that offers an improvement over the best solution so far, it is flipped instead (aspiration). If no improving variable was found, a variable that offers zero improvement is flipped (side step). The crossover operator repeats until n side steps[3] were done consecutively. In table 1 the THLS, the THC1 and THC2 are compared on instances from the spin glass problem (row 1 to 5, number of clauses increased from 120 to 231, number of literals from 224 to 440) and the break minimization problem (row 6 to 10, number of clauses increased from 27 to 343, number of literals from 162 to 2058) taken from the Max-Sat competition 2006. The necessary population size was set to 10, the generation termination condition to overall 100.000 flips and each simulation repeated at least 100 times. The THLS

[1]Inverted Binary Assignment of a variable.
[2]Each variable that is not tabu.
[3]Number of overall necessary variable assignments.

alone wasn't able to find an optimal solution for half of the instances in every run (denoted by SR, the precentage of satisfied runs). In combination with THC1, the success rate increases and except for one instance the algorithm finds an optimal solution in every run. But not only the success rate, also the average number of flips (denoted by AFS, average number of flips to solution) is increased, which leads to an extended runtime. When combined with the advanced crossover operator THC2, the success rate is also increased for every instance. Besides one instance, the AFDS is inferior to the results of the combination with THC1. Therefore, THC2 is a more efficient combination for THLS for these instances.

THLS		THLS&THC1		THLS&THC2	
SR	AFS	SR	AFS	SR	AFS
1.00	2694	1.00	2933	1.00	1581
1.00	3714	1.00	3954	1.00	2374
1.00	4990	1.00	5185	1.00	2755
0.94	5901	1.00	6170	1.00	3584
1.00	420	1.00	595	1.00	366
0.98	1148	1.00	1584	1.00	1048
0.98	2314	1.00	3194	1.00	1918
0.74	4998	1.00	7823	1.00	8018
0.06	5724	0.46	40818	0.30	39463

Table 1: Recombination Efficiency

Beside the Memetic Algorithm proposed here, other population based evolutionary approaches like GRASP [12], Genetic Algorithms [8] and other Memetic Algorithms exist. Some of the latter are [7], [5] and [11], which present specialized crossover operators for the Max-SAT problem. In contrast to the population based approaches, many local search methods based on evolving a single solution have been proposed. Based on random instances (140 variables and clauses scaled from 200 (first row) to 1000 (last row)) our algorithm (THLS&THC2, referred as MATHiC) is compared (see table 2) to two local search algorithms, that use similar local search heuristics. HSAT [10] is based on a flip

	HSAT		IRoTS		MATHiC	
suite	SR	AFS	SR	AFS	SR	AFS
random2	0,46	4837	1,00	1643	1,00	492
random4	0,54	1750	1,00	1031	1,00	591
random6	0,52	1658	1,00	772	1,00	482
random8	0,42	507	1,00	310	1,00	415
random10	0,29	505	1,00	476	1,00	490

Table 2: Comparison with Local Search Algorithms

history and IRoTS uses a tabu list to guide the search. The overall winner of this test is IRoTS, that found an optimal solution in every run on each instance. Also the number of flips IRoTS needed to find a solution is quite small. The results for HSAT show, that it isn't a reliable solver for these instances, because it found an optimal solution seldom, but if it finds one, it needs only a few flips to do so. Our approach found an optimal solution in all of the runs, but needs much less flips to find a solution for 200 up to 600 clauses. Because of the global and local nature of the crossover operator and the local search procedure, the overall impact is complementary and ensures a good compromise between exploration and exploitation of the search space. Our algorithm has been empirical verified on random and structured test set instances (see [4] for additional results) and has been compared to local search algorithms that make use of similar heuristics as our approach. The simulation results justifies the competitiveness of our approach. Although our algorithm needs marginally more time (same order of computational complexity) to find a solution on two bigger instances, the small variation of overall computational effort is very promising. As a work in progress, all at the end of section 1 mentioned phases are evaluated and an improved recombination design (THC3) is under consideration.

3. REFERENCES

[1] A. Armando, C. Castellini, E. Giunchiglia, F. Giunchiglia, and A. Tacchella. SAT-based decision procedures for automated reasoning: a unifying perspective, 2002.

[2] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979.

[3] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. *Lecture Notes in Computer Science*, 1579:193–207, 1999.

[4] M. Borschbach and A. Exeler. A search agent for a Max-2SAT memetic algorithm approach. In *Proceedings of Advances of Computer Application and COmputational Science (ACACOS), In: Selected Paper from the Conference in Hangzhou*, 2008.

[5] D. Boughaci, B. Benhamou, and H. Drias. Iga: An improved genetic algorithm for Max-SAT problems. In *Indian International Conference on Artificial Intelligence (IICAI)-07*, 2007.

[6] S. A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971.

[7] C. Fleurent and J. Ferland. Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26:619–652, 1996.

[8] J. Frank. A study of genetic algorithms to find approximate solutions to hard 3cnf problems, 1994.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[10] I. P. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for SAT. In *National Conference on Artificial Intelligence*, pages 28–33, 1993.

[11] F. Lardeux, F. Saubion, and J.-K. Hao. Gasat: A genetic local search algorithm for the satisfiability problem. *Evolutionary Computation*, 14(2):223–253, 2006.

[12] P. Pardalos, L. Pitsoulis, and M. Resende. A parallel grasp for Max-SAT problems, 1996.

[13] H. Zhang, D. Li, and H. Shen. A sat based scheduler for tournament schedules. In *SAT*, 2004.