A Search Agent for a Max-2sat Memetic Algorithm Approach

Markus Borschbach and Andre Exeler
University of Münster
Dept. of Mathematics and Natural Science
Einsteinstr. 62, D-48149 Münster
GERMANY

Abstract: Various algorithms have been suggested for the Max-SAT problem. The solution for the Max-2SAT is the starting point for a selection of these approximation algorithms. This paper aims at introducing approaches for Max-2SAT by a brief review of the basic ideas. 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. The overall performance is verified by empirical simulation and is used to compare the developed approach to other state up-to-date and of the art algorithms.

Key–Words: Max2SAT, memetic algorithms, tabu search.

1 Introduction

The satisfiability problem (SAT) is one of the most studied problems in computer science and has a lot of applications in scheduling[3], graph theory, automated reasoning[2] and other domains like VLSI[1]. Since SAT is \( NP \)-complete[15], it doesn’t seem to be promising to find a complete algorithm that performs well on all sizes of problem instances.

Its optimization variant Max-SAT consists of identifying a variable assignment to a given propositional formula in conjunctive normal form (CNF) which maximizes the number of satisfied clauses. Max-SAT and its variant Max-2SAT, where the number of literals in each clause is limited by two, is \( NP \)-hard[11]. In contrast to this, 2SAT can be solved in linear time[18].

In recent years many approaches for Max-SAT have been proposed. These approaches can be divided into two classes, complete and incomplete algorithms. Complete algorithms explore the whole search space of variable assignments; often in an implicit way. For that reason, these approaches always provide an exact solution to the problem. Most of these algorithms for Max-SAT are based on the Davis-Putnam-Logemann-Loveland procedure (DPLL)[14] or similar branch-and-bound algorithms. These algorithms differ mainly by the underlying heuristic for the branching rules, transformation rules or the heuristics to compute the upper and lower bound for the solution.

Among the best exact algorithms is MaxSatZ, which won most of the benchmarks for unweighted CNF-formulas in the Max-SAT evaluation 06\(^1\). Incomplete algorithms do not perform examination of the search space in a systematic way. Therefore, they can not guarantee to find an exact solution. Many of the most successful solvers of the incomplete type of approaches for Max-SAT are stochastic local search algorithms (SLS). Since GSAT and Walksat have been suggested, a lot of improvements have been proposed to enhance the performance of local search methods. These local search strategies are briefly reviewed in the next section. The remainder of this paper is organized as follows. In section 3, a new combination of search strategies is suggested and section 4 presents the simulation and verification setup. The simulation results are part of section 5 and the paper is completed by the conclusion and outlook on future work and open problems.

2 Related Work

An early advancement of GSAT and Walksat (refer to the previous section) was given by the HSAT algorithm[10]. HSAT uses historical information to select deterministically which variable is flipped\(^2\) next. In every step a variable is selected which offers the maximum improvement of the number of satisfied clauses. If there is a choice of variables, HSAT flips the one that was least recently flipped.

Other local search methods use tabu search to prevent loops when moving away from local optima and to enhance the exploration capabilities of the algorithm. One of them is the iterated robust tabu search (IRoTS)[5]. IRoTS consists of a perturbation phase and a local search phase which are repeated until an acceptance criterion is met. Both phases are derivations of the so called RoTS algorithm[4] that flips in every step a non tabu variable that provides

\(^1\)http://www.iiia.csic.es/~maxsat06/

\(^2\)Alternating the value either from true to false or vice versa.
maximum improvement and declares it tabu for the next $t^3$ steps. An exception to this rule is only made if a variable that is declared tabu achieves an improvement over the best solution so far (called aspiration). A variable is forced to be flipped if it has not been flipped within the last $10n$ search steps\(^4\). In contrast to most of the other tabu search algorithms, the tabu list length isn’t fixed but changed every $n$ iterations within a randomly chosen interval $[t_{min}, t_{max}]$. The differences between the perturbation phase and the local search phase is the number of search steps proceeded in each run and the tabu list length. This approach is selected for comparison, because it might be considered as a $(1 + 1)$ evolutionary strategy consisting of a mutation operator (the perturbation phase) and a local search operator.

Another promising approach is guided local search [8] (GLS) that is based on HSAT as the underlying search method. After each HSAT-search step GLS modifies a vector of clause penalties $clp_i$ for each clause $C_i$ in the given formula. When trapped in a local optimum, all penalty values of the clauses are increased if the following holds: the clause $C_i$ is unsatisfied under the current assignment $a$ and maximizes the utility function $util(a, C_i) := w_i / (1 + clp_i)$.\(^5\) The clause penalties are used if the improvement of a flip is computed. A variant of the algorithm called GLS2 regularly decreases all clause penalties by a constant factor.

In contrast to the local search methods based on a single solution, some population based evolutionary approaches like GRASP[6], Genetic Algorithms [12] and Memetic Algorithms have been proposed. Some of the latter are [13], [16] and [9] which present specialized crossover operators for the Max-SAT problem.

### 3 Max-2SAT Memetic Algorithm

The term *Memetic Algorithms* (MA) is used for a broad class of metaheuristics, that can be described as a hybrid type of approaches combining the global search abilities of evolutionary algorithms and local search procedures.

Most of the MA implementations use a population based search and try to use a lot of problem specific knowledge based on heuristics or specialized operators, among others. In the context of memetic algorithms, the individuals of the population are also called *agents*, because of their competing, interacting and self-improving nature.

In the algorithm presented here (see figure 1 for the framework of a Memetic Algorithm with Tabu History Computed (MATHIC)), a population of agents that consist of an encoding of a solution to the problem and a flip history is used. While a solution to a Max-2SAT-problem is given by a variable assignment, a boolean array of length $n$ is used to store it. The flip history is stored in a list of length $n$ and is consisting of each variable used so far. At the beginning, the least recently flipped variable and in the end the most recently variable is stored. If a variable is flipped, it will be removed from the list and reinserted at the end of the list.

The number of satisfied clauses is used to evaluate the individuals. The improvement obtained by flipping a variable is determined by the difference between the number of variables satisfied after and before the flipping is processed. Therefore the difference is positive if the number of satisfied clauses is increased after the flip.

Starting with a randomly generated initial population of size $p$, the algorithm at first selects the mating-pool $M$ of individuals, that are used to generate the next generation. This selection is done using stochastic-universal-sampling [17]. The fitness function used here is the number of satisfied clauses minus the minimal number of clauses satisfied by an individual in the population (a fitness scaling). For each single crossover operation one successor individual is generated out of two predecessor individuals. To ensure a constant population size of $|P|$, every individual in the mating-pool is used twice for crossover (see figure 1) and afterwards removed from the mating pool (crossover-operator(M)). After this step the local-search-operator is applied to each of these new individuals. These improved individuals are reinserted into the population and the best individual of the old population is kept, if no better new individual was found.

---

\(^3\)In advance configured steps size.

\(^4\)n is the number of variables.

\(^5\)w$_i$ is 1 if $C_i$ is unsatisfied, else 0.
PROCEDURE THLS(individual X)
BEGIN
sidesteps := 0;
WHILE sidesteps < n DO
BEGIN
v := variable with max. improvement;
IF improvement(X,v) > 0 DO
sidesteps := sidesteps + 1;
ELSE IF improvement(X,v) = 0 AND v is not tabu DO
BEGIN
X.flip(v);
sidesteps := sidesteps + 1;
END
END
END
RETURN X;
END

Figure 3: Tabu History Local Search

3.1 The Crossover Operator

The advanced crossover operator takes two individuals and generates one new individual (called successor here or child in other literature) out of them (see figure 2 for Tabu History Crossover 2 (THC2) algorithm). To maximize the number of satisfied clauses in the successor, the operator examines all clauses that are simultaneously unsatisfied in both predecessors, and flips the variable at each clause that leads to the best overall improvement. The improvement used is calculated by summing the improvement of the flipping of the variable in both predecessors. The values of the flipped variables are passed to the successor. All other variables that were not addressed by this procedure are assigned by using a simple uniform crossover. Up to this point, the operator is very similar to the operators presented in [9] and in [16] and this simplified version (called THC1) will be used for empirical comparison. The advances made here are as follows. As mentioned before, 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 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 ist set to 0.5.

3.2 The Local Search Operator

The improved local search operator uses the flip history to guide the search on each agent. The last tl elements of the history list are declared tabu during the local search phase. The algorithm selects all the allowed6 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 algorithm repeats until n side steps were done consecutively.

Figure 2: Tabu History Crossover 2

4 Experimental Design

The source code was written in java and all tests were done on an Athlon XP 1700+ machine with 1GB RAM running on Linux. Most of the instances, i.e. the random instances, the MAX-CUT instances and the spin-glass instances, are taken from the MaxSAT evaluation 06 homepage7. The break minimization instances were introduced in [7] and are available on the website of Mutsunori Yagiura8. For the comparison with HSAT and IRoTS the implementation included in ucb sat9 was used with the default

---

6All variables that are not tabu.
7http://www.iiia.csic.es/~maxsat06/
8http://www.al cm.is.nagoya-u.ac.jp/~yagiura/sat/max2sat/
9http://www.satlib.org/ubcsat/
parameters. The comparisons with MaxSatz were done using the source code available on Jodie Planes’ homepage\textsuperscript{10}. The random MAX-CUT instances are grouped into 2 test suites named maxcut1 and maxcut2 consisting of 10 instances with 420 and 500 clauses and 60 variables in each instance. In each test, 50 runs are performed. In order to measure the complexity of the computation, the average number of flips to solution (AFS) - representing the number of flips required before reaching an optimum is used as a criterion of comparison. The success rate (SR), that is the percentage of runs where an optimum was found, and the average computation time (in seconds) is measured for comparison.

5 Simulation Results

To demonstrate the performance of our new crossover operator (THC2), we compared the tabu history local search without a crossover operator (THLS) to the combination with our simple crossover (THLS&THC1) and to the combination with our advanced crossover operator (THLS&THC2). When the local search operator is used only, all individuals in the mating-pool $M$ are directly improved by the local search operator and furthermore reinserted into the population $P$. The result of this comparison can be found in table 2. The THLS alone wasn’t able to find an optimal solution for half of the instances in every run. In combination with THC1, the success rate increases and except for the instance t7pm3-9999.spn the algorithm finds an optimal solution in every run. But not only the success rate, also the average number of flips 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. Although for the instances t6pm3-8888.spn, t7pm3-9999.spn and break_22_231_440 it is inferior to the results of the combination with THC1. But unlike the results with THC1, THLS&THC2 leads to a decreasing in the average number of flips to the solution for the break minimization instances. The increasing for the spin glass instances is not as big as in the combination with THC1. Therefore, THC2 is a more efficient combination for THLS.

5.1 Comparison with Local Search

In this section, our algorithm (THLS&THC2, referred as MATHiC) is compared to two local search algorithms, that use similar heuristics. HSAT is based on a flip history and IRoTS uses a tabu list to guide the search. The results of this comparison can be found in table 1 and table 3. 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 relatively small. Our approach found a n optimal solution in nearly all of the runs, but needs much more flips to find a solution. The results for HSAT show, that it isn’t a relyable solver for these instances, because it found an optimal solution seldomly, but if it finds one, it needs only a few flips to do so.

<table>
<thead>
<tr>
<th>suite</th>
<th>HSAT SR</th>
<th>AFS</th>
<th>IRoTS SR</th>
<th>AFS</th>
<th>MATHiC SR</th>
<th>AFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxcut1</td>
<td>0.08</td>
<td>34</td>
<td>1.00</td>
<td>172</td>
<td>0.95</td>
<td>3068</td>
</tr>
<tr>
<td>maxcut2</td>
<td>0.10</td>
<td>29</td>
<td>1.00</td>
<td>256</td>
<td>0.91</td>
<td>3781</td>
</tr>
</tbody>
</table>

Table 1: Comparison with local search algorithms on random instances

5.2 Comparison with Complete Algorithms

In table 4 the algorithm is compared to MaxSatz, a state of the art complete Max-SAT solver. Both algorithms were run 50 times and the average number of false clauses (afc) and the average runtime (time) were measured. Although complete and incomplete algorithms for Max-SAT can’t be compared reasonable, because of their different asymptotic runtime\textsuperscript{11}, hence every incomplete algorithm with polynomial runtime exceeds MaxSatz on some instances, the result is noteworthy. Therefore, a comparison is done to show in general the different nature of these two classes of algorithms. As expected, MaxSatz finds an optimal solution in every run. On the biggest two spin glass instances the execution was aborted, because MaxSatz didn’t find a solution within 10 hours. In contrast to this, our algorithm didn’t accomplish to find an optimal solution in every run, but it always find a solution very close to the optimum and its runtime is substantially smaller.

6 Conclusions

In this paper, we proposed a new memetic algorithm for Max-2SAT problems. The proposed algorithms differs in the onset and use of a separate tabu history for each individual. Moreover, an advanced Max-2SAT crossover operator is used to generate new individuals. Furthermore, these individuals are improved

\textsuperscript{10}http://web.udl.es/usuaris/m4372594/

\textsuperscript{11}complete algorithms for Max-SAT will always have an exponential runtime unless $P = NP$
by a local search algorithm that combines the heuristics of tabu search and history search. 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 empirically verified on random and structured test set instances 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 more time to find a solution on some smaller instances, the comparable quality of the obtained solutions and the low number of flip-operators-computations necessary is very promising.

7 Future Work

Our future work will consist of refining the crossover operator by the introduction of a recombination scheme for the tabu histories. Also a mutation operator will be added to ensure an optimized diversity of the population during the search process. Furthermore, the algorithm will be generalized to work not only on Max-2SAT but on all Max-SAT instances.

References:

### Table 2: Comparison of the crossover operators on structured instances

<table>
<thead>
<tr>
<th>suite</th>
<th>n</th>
<th>m</th>
<th>THL</th>
<th>THL&amp;THC1</th>
<th>THL&amp;THC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>break_16,120,224</td>
<td>120</td>
<td>224</td>
<td>1.00</td>
<td>3309</td>
<td>1.00</td>
</tr>
<tr>
<td>break_18,183,288</td>
<td>183</td>
<td>288</td>
<td>1.00</td>
<td>3745</td>
<td>1.00</td>
</tr>
<tr>
<td>break_20,190,360</td>
<td>190</td>
<td>360</td>
<td>1.00</td>
<td>4801</td>
<td>1.00</td>
</tr>
<tr>
<td>break_22,231,440</td>
<td>231</td>
<td>440</td>
<td>0.86</td>
<td>6119</td>
<td>0.98</td>
</tr>
<tr>
<td>t3pm3-5555.spn</td>
<td>27</td>
<td>162</td>
<td>1.00</td>
<td>294</td>
<td>1.00</td>
</tr>
<tr>
<td>t4pm3-6666.spn</td>
<td>64</td>
<td>384</td>
<td>0.98</td>
<td>916</td>
<td>1.00</td>
</tr>
<tr>
<td>t5pm3-7777.spn</td>
<td>125</td>
<td>750</td>
<td>0.94</td>
<td>1891</td>
<td>1.00</td>
</tr>
<tr>
<td>t6pm3-8888.spn</td>
<td>216</td>
<td>1296</td>
<td>0.32</td>
<td>3901</td>
<td>0.98</td>
</tr>
<tr>
<td>t7pm3-9999.spn</td>
<td>343</td>
<td>2058</td>
<td>0.06</td>
<td>5724</td>
<td>0.55</td>
</tr>
</tbody>
</table>

### Table 3: Comparison with local search algorithms on structured instances

<table>
<thead>
<tr>
<th>inst.</th>
<th>n</th>
<th>m</th>
<th>HSA T</th>
<th>IRoTS</th>
<th>MATHiC</th>
</tr>
</thead>
<tbody>
<tr>
<td>break_16,120,224</td>
<td>120</td>
<td>224</td>
<td>1.00</td>
<td>610</td>
<td>1.00</td>
</tr>
<tr>
<td>break_18,183,288</td>
<td>183</td>
<td>288</td>
<td>0.50</td>
<td>1661</td>
<td>1.00</td>
</tr>
<tr>
<td>break_20,190,360</td>
<td>190</td>
<td>360</td>
<td>0.84</td>
<td>4585</td>
<td>1.00</td>
</tr>
<tr>
<td>break_22,231,440</td>
<td>231</td>
<td>440</td>
<td>0.32</td>
<td>3225</td>
<td>1.00</td>
</tr>
<tr>
<td>t3pm3-5555.spn</td>
<td>27</td>
<td>162</td>
<td>0.56</td>
<td>11</td>
<td>1.00</td>
</tr>
<tr>
<td>t4pm3-6666.spn</td>
<td>64</td>
<td>384</td>
<td>0.28</td>
<td>71</td>
<td>1.00</td>
</tr>
<tr>
<td>t5pm3-7777.spn</td>
<td>125</td>
<td>750</td>
<td>0.5</td>
<td>337</td>
<td>1.00</td>
</tr>
<tr>
<td>t6pm3-8888.spn</td>
<td>216</td>
<td>1296</td>
<td>0.12</td>
<td>230</td>
<td>1.00</td>
</tr>
<tr>
<td>t7pm3-9999.spn</td>
<td>343</td>
<td>2058</td>
<td>0.00</td>
<td>–</td>
<td>1.00</td>
</tr>
</tbody>
</table>

### Table 4: Comparison with a complete algorithm

<table>
<thead>
<tr>
<th>inst.</th>
<th>n</th>
<th>m</th>
<th>MaxSat</th>
<th>MATHiC</th>
</tr>
</thead>
<tbody>
<tr>
<td>break_16,120,224</td>
<td>120</td>
<td>224</td>
<td>16.00</td>
<td>16.00</td>
</tr>
<tr>
<td>break_18,183,288</td>
<td>183</td>
<td>288</td>
<td>20.00</td>
<td>20.00</td>
</tr>
<tr>
<td>break_20,190,360</td>
<td>190</td>
<td>360</td>
<td>24.00</td>
<td>24.00</td>
</tr>
<tr>
<td>break_22,231,440</td>
<td>231</td>
<td>440</td>
<td>29.00</td>
<td>29.00</td>
</tr>
<tr>
<td>t3pm3-5555.spn</td>
<td>27</td>
<td>162</td>
<td>17.00</td>
<td>17.00</td>
</tr>
<tr>
<td>t4pm3-6666.spn</td>
<td>64</td>
<td>384</td>
<td>38.00</td>
<td>38.00</td>
</tr>
<tr>
<td>t5pm3-7777.spn</td>
<td>125</td>
<td>750</td>
<td>78.00</td>
<td>78.00</td>
</tr>
<tr>
<td>t6pm3-8888.spn</td>
<td>216</td>
<td>1296</td>
<td>–</td>
<td>10h</td>
</tr>
<tr>
<td>t7pm3-9999.spn</td>
<td>343</td>
<td>2058</td>
<td>–</td>
<td>10h</td>
</tr>
</tbody>
</table>

---

*RECENT ADVANCES IN SYSTEMS, COMMUNICATIONS & COMPUTERS, Selected Papers from the WSEAS Conferences in Hangzhou, China, April 6-8, 2008*