Evolved Preambles for MAX-SAT Heuristics

Luís O. Rigo Jr.
Valmir C. Barbosa*

Programa de Engenharia de Sistemas e Computação, COPPE
Universidade Federal do Rio de Janeiro
Caixa Postal 68511
21941-972 Rio de Janeiro - RJ, Brazil

Abstract

MAX-SAT heuristics normally operate from random initial truth assignments to the variables. We consider the use of what we call preambles, which are sequences of variables with corresponding single-variable assignment actions intended to be used to determine a more suitable initial truth assignment for a given problem instance and a given heuristic. For a number of well established MAX-SAT heuristics and benchmark instances, we demonstrate that preambles can be evolved by a genetic algorithm such that the heuristics are outperformed in a significant fraction of the cases.

Keywords: MAX-SAT, MAX-SAT heuristics, Initial truth assignments.

1 Introduction

Given a set $V$ of Boolean variables and a set of disjunctive clauses on literals from $V$ (i.e., variables or their negations), MAX-SAT asks for a truth assignment to the variables that maximizes the number of clauses that are satisfied (i.e., made true by that assignment). MAX-SAT is NP-hard [8] but can be approximated in polynomial time, though not as close to the optimum as one wishes (cf. [2] and references therein, as well as [5] for the restricted case of three-literal clauses). MAX-SAT has enjoyed a paradigmatic status over the years, not only because of its close relation to SAT, the first decision problem to be proven NP-complete, but also because of its importance to other areas (e.g., constraint satisfaction in artificial intelligence [6]).

Since NP-hardness is a property of worst-case scenarios, the difficulty of actually solving a specific instance of an NP-hard problem varies widely with both the instance’s size and internal structure. In fact, in recent years it has become

*Corresponding author (valmir@cos.ufrj.br).
increasingly clear that small changes in either can lead to significant variation in
an algorithm’s performance, possibly even to a divide between the instance’s be-
ing solvable or unsolvable by that algorithm given the computational resources
at hand and the time one is willing to spend \[13\]. Following some early ground-
work \[25\], several attempts have been made at providing theoretical foundations
or practical guidelines for automatically selecting which method to use given the
instance \[26, 23, 7, 12, 17, 19, 31\].

Here we investigate a different, though related, approach to method selection
in the case of MAX-SAT instances. Since all MAX-SAT heuristics require an
initial truth assignment to the variables, and considering that this is invariably
chosen at random, a natural question seems to be whether it is worth spending
some additional effort to determine an initial assignment that is better suited
to the instance at hand. Once we adopt this two-stage template comprising an
initial-assignment selection in tandem with a heuristic, fixing the latter reduces
the issue of method selection to that of identifying a procedure to determine an
appropriate initial assignment. We refer to this procedure as a preamble to the
heuristic. As we demonstrate in the sequel, for several state-of-the-art heuristics
and problem instances the effort to come up with an appropriate preamble pays
off in terms of better solutions for the same amount of time.

We proceed in the following manner. First, in Section 2 we define what
preambles are in the case of MAX-SAT. Then we introduce an evolutiona-
ry method for preamble determination in Section 3 and give computational
results in Section 4. We close with concluding remarks in Section 5.

2 MAX-SAT preambles

Let \( n \) be the number of variables in \( V \). Given a MAX-SAT instance on \( V \), a
preamble \( p \) of length \( \ell \) is a sequence of pairs \((v_1, a_1), (v_2, a_2), \ldots, (v_{\ell}, a_{\ell})\), each
representing a computational step to be taken as the preamble is played out. In
this sequence, and for \( 1 \leq k \leq \ell \), the \( k \)th pair is such that \( v_k \in V \) and \( a_k \) is one
of 2, 1, or 0, indicating respectively whether to leave the value of \( v_k \) unchanged,
to act greedily when choosing a value for \( v_k \), or to act contrarily to such greedy
assignment. A preamble need not include all \( n \) variables, and likewise a variable
may appear more than once in it.

The greedy action to which \( a_k \) sometimes refers assigns to \( v_k \) the truth value
that maximizes the number of satisfied clauses at that point in the preamble.
Algorithmically, then, playing out \( p \) is equivalent to performing the following
steps from some initial truth assignment to the variables in \( V \):

1. \( k := 1 \).
2. If \( a_k = 2 \), then proceed to Step 5.
3. Given the current values of all other variables, compute the number of clauses that get satisfied for each of the two possible assignments to \( v_k \).
4. If \( a_k = 1 \), then set \( v_k \) to the truth value yielding the greatest number of satisfied clauses. If \( a_k = 0 \), then do the opposite. Break ties randomly.

5. \( k := k + 1 \). If \( k \leq \ell \), then proceed to Step 2.

We use random initial assignments exclusively. A MAX-SAT preamble, therefore, can be thought of as isolating such initial randomness from the heuristic proper that is to follow the preamble. Instead of starting the heuristic at its usual random initial assignment, we start it at the assignment determined by running the preamble.

It is curious to note that, as defined, a preamble generalizes the sequence of steps generated by the simulated annealing method [16] when applied to MAX-SAT. In fact, what simulated annealing does in this case, following one of its variations [9, 3], is to choose \( v_k \) by cycling through the members of \( V \) and then let \( a_k \) be either 1 or 0 with the Boltzmann-Gibbs probability. At the high initial temperatures the two outcomes are nearly equally probable, but the near-zero final temperatures imply \( a_k = 1 \) (i.e., be greedy) with high probability. The generalization that comes with our definition allows for various possibilities of preamble construction, as the evolutionary procedure we describe next.

3 Methods

Given a MAX-SAT instance and heuristic \( H \), our approach is to evolve the best possible preamble to \( H \). We do so through a genetic algorithm of the generational type [24]. The description that follows refers to parameter values that were determined in an initial calibration phase. This phase used the heuristics \textit{gsat} [28], \textit{gwsat} [27], \textit{hsat} [10], \textit{husat} [11], \textit{gsat-tabu} [21], \textit{novelty} [22], \textit{walksat-tabu} [22], \textit{adaptnovelty+} [14], \textit{saps} [15], and \textit{sapsnr} [29] as heuristic \( H \), and also the instances \textit{C880mul}, \textit{am8_8}, \textit{c3540mul}, \textit{term1mul}, and \textit{vdamul} from [18]. Each of the latter involves variables that number in the order of \( 10^4 \) and clauses numbering in the order of \( 10^5 \). Moreover, not all optima are known (cf. Section 4).

The genetic algorithm operates on a population of 50 individuals, each being a preamble to heuristic \( H \). The fitness of individual \( p \) is computed as follows. First \( p \) is run from 10 random truth assignments to the variables, then \( H \) is run from the truth assignment resulting from the run of \( p \) that satisfied the most clauses (ties between runs of \( p \) are broken randomly). Let \( R(p) \) denote the number of clauses satisfied by this best run of \( p \) and \( S_H(p) \) the number of clauses satisfied after \( H \) is run. The fitness of individual \( p \) is the pair \( (S_H(p), R(p)) \). Whenever two individuals’ fitnesses are compared, ties are first broken lexicographically, then randomly. Selection is always performed from linearly normalized fitnesses, the fittest individual of the population being 20 times as fit as the least fit.

For each MAX-SAT instance and each heuristic \( H \), we let the genetic algorithm run for a fixed amount of time, during which a new population is
repeatedly produced from the current one and replaces it. The initial population comprises individuals of maximum length 1.5\(n\), each created randomly to contain at least 0.4\(n\) distinct variables. The process of creating each new population starts by an elitist step that transfers the 20% fittest individuals from the current population to the new one. It then repeats the following until the new population is full.

First a decision is made as to whether crossover (with probability 0.25) or mutation (with probability 0.75) is to be performed. For crossover two individuals are selected from the current population and each is partitioned into three sections for application of the standard two-point crossover operator. The partitioning is done randomly, provided the middle section contains exactly 0.4\(n\) distinct variables, which is always possible by construction of the initial population (though at times either of the two extreme sections may turn out to be empty). The resulting two individuals (whose lengths are no longer bounded by 1.5\(n\)) are added to the new population. For mutation a single individual is selected from the current population and 50% of its pairs are chosen at random. Each of these, say the \(k\)th pair, undergoes either a random change to both \(v_k\) and \(a_k\) (if this will leave the individual with at least 0.4\(n\) distinct variables) or simply a random change to \(a_k\) (otherwise). The mutant is then added to the new population.

The calibration phase referred to above also yielded three champion heuristics, viz. novelty, walksat-tabu, and adaptnovelty+. The results we give in Section 4 refer exclusively to these, used either in conjunction with the genetic algorithm as described above or by themselves. In the latter case each heuristic is run repeatedly, each time from a new random truth assignment to the variables, until the same fixed amount of time used for the genetic algorithm has elapsed. The result reported by the genetic algorithm refers to the fittest individual in the last population to have been filled during that time. As for the heuristic, in order to compare its performance with that of the genetic algorithm as fairly as possible the result that is reported is the best one obtained after every 50 repetitions.

All experiments were performed from within the UBCSAT environment \[30\]. Optima, whenever possible, were discovered separately via the 2010 release of the MSUnCore code to solve MAX-SAT exactly \[20\].

4 Computational results

In our experiments we tackled all 100 instances of the 2004 SAT competition \[18\], henceforth referred to as the 2004 dataset, and all 112 instances of the 2008 MAX-SAT evaluation \[1\], henceforth referred to as the 2008 dataset. The time allotted for each instance to the genetic algorithm or each of the three heuristics by itself was of 60 minutes, always on identical hardware and software, always with exclusive access to the system. We report exclusively on the hardest instances from either dataset, here defined to be those for which MSUnCore found no answer as a result of being stymied by the available 4 gigabytes of
RAM and the system’s inability to perform further swapping. There are 51 such instances in the 2004 dataset, 11 in the 2008 dataset, totaling 62 instances.

Our results are given in Tables 1 through 3 respectively for $H = \text{novelty}$, $H = \text{walksat-tabu}$, and $H = \text{adaptnovelty+}$. Each table contains a row for each of the 62 instances, with a horizontal line separating the instances of the 2004 dataset from those of the 2008 dataset. For each instance the number $n$ of variables is given, as well as the number of clauses ($m$) and results for the genetic algorithm and for the heuristic in question by itself. These results are the number of satisfied clauses and the time at which this solution was first found during the allotted 60 minutes. Missing results indicate either that no population could be filled during this time (in the case of the genetic algorithm) or that no batch of 50 runs of the heuristic could be finished.

Some entries in the tables are highlighted by a bold typeface to indicate that the genetic algorithm found a solution strictly better than the one found by the heuristic when used by itself, or a solution satisfying the same number of clauses but first encountered in a shorter time. In the former case only the number of satisfied clauses is highlighted, in the latter case the time is highlighted as well. The number of highlighted instances amounts to the ratios given in Table 4. Clearly, with the notable exception of $H = \text{walksat-tabu}$ on the 2008 dataset (on which the use of $H$ alone outperformed the genetic algorithm on all 11 instances), the genetic algorithm succeeds well on a significant fraction of the instances.

Revising these ratios to contemplate all instances from both datasets (i.e., include the results omitted from Tables 1 through 3) yields the ratios in Table 5. These show that the genetic algorithm fares even better when evaluated on all 100 instances of the 2004 dataset. They also show slightly lower ratios for the genetic algorithm on the 112-instance 2008 dataset for $H = \text{novelty}$ and $H = \text{adaptnovelty+}$. As for $H = \text{walksat-tabu}$, we see in Table 5 a dramatic increase from the 0.000 of Table 4 indicating that for this particular $H$ on the complete 2008 dataset the genetic algorithm does better than the heuristic alone only on the comparatively easier instances (and then for a significant fraction of them).

5 Concluding remarks

Given a heuristic for some problem of combinatorial optimization, a preamble such as we defined in Section 2 for MAX-SAT is a selector of initial conditions. As such, it aims at isolating the inevitable randomness of the initial conditions one normally uses with such heuristics from the heuristic itself. By doing so, preambles attempt to poise the heuristic to operate from more favorable initial conditions.

In this paper we have demonstrated the success of MAX-SAT preambles when they are discovered, given the MAX-SAT instance and heuristic of interest, via an evolutionary algorithm. As we showed in Section 4 for well established benchmark instances and heuristics the resulting genetic algorithm
Table 1: Results for $H = \text{novelty}$. Times are given in minutes.

| Instance                  | n      | m      | Genetic algorithm | novelty alone |
|---------------------------|--------|--------|-------------------|---------------|
| cr5000m1                  | 5248   | 33199  | 33176             | 17.965        |
| df2988m1                  | 9540   | 61421  | 61375             | 21.391        |
| dalnum1                   | 9426   | 59991  | 59972             | 59.961        |
| fgri1m1                   | 3230   | 20575  | 20574             | 0.543         |
| k2m1                      | 11680  | 74581  | 74524             | 1.729         |
| x1m1                      | 8760   | 55577  | 55570             | 1.550         |
| an_6                      | 2269   | 7814   | 7813              | 0.263         |
| an_7                      | 4264   | 14751  | 14744             | 55.270        |
| an_8                      | 7631   | 25538  | 25331             | 37.914        |
| li-test-61                | 11908  | 41393  | 40982             | 1.827         |
| li-exam-61                | 28147  | 108436 | 108011            | 47.997        |
| li-exam-62                | 28147  | 108436 | 107999            | 56.149        |
| li-exam-63                | 28147  | 108436 | 107998            | 29.129        |
| li-test4-100              | 36809  | 142491 | 141844            | 34.560        |
| li-test4-101              | 36809  | 142491 | 141858            | 40.733        |
| li-test4-94               | 36809  | 142491 | 141863            | 2.556         |
| li-test4-95               | 36809  | 142491 | 141850            | 58.823        |
| li-test4-96               | 36809  | 142491 | 141858            | 6.038         |
| li-test4-97               | 36809  | 142491 | 141855            | 29.113        |
| li-test4-98               | 36809  | 142491 | 141862            | 40.632        |
| li-test4-99               | 36809  | 142491 | 141859            | 12.537        |
| gripper10u                | 2312   | 18666  | 18663             | 0.877         |
| gripper11u                | 3084   | 26019  | 26017             | 15.150        |
| gripper12u                | 3352   | 29412  | 29409             | 14.485        |
| gripper13u                | 4268   | 38965  | 38961             | 9.830         |
| gripper14u                | 4584   | 43390  | 43386             | 23.381        |
| bc56-sensors-1-k391-unsat  | 561371 | 1778987 | 1600025         | 15.997      |
| bc56-sensors-2-k592-unsat  | 850288 | 294319 | 236874            | 48.252      |
| bc57-sensors-1-k303-unsat  | 435701 | 1379987 | 1261961         | 39.150      |
| dne-03-1-k247-unsat       | 261352 | 77077  | 736229            | 49.430      |
| motors-stuck-1-k407-unsat | 654766 | 2068742| 1842393           | 55.420      |
| motors-stuck-2-k314-unsat | 505536 | 1306817| 1445274           | 53.360      |
| valves-gates-1-k617-unsat | 985042 | 3113540| 2714448           | 58.357      |
| 6pipe                     | 15800  | 394739 | 394717            | 42.251      |
| 7pipe                     | 23910  | 731118 | 731118            | 54.921      |
| cmb1                      | 5910   | 16804  | 16749             | 43.330      |
| dp12u11                   | 11137  | 30792  | 30785             | 11.240      |
| f2clk10                   | 34678  | 101319 | 100629            | 18.230      |
| f2fsk300                  | 194762 | 530713 | 506270            | 9.520      |
| homer17                   | 286    | 1732   | 1738              | 0.126      |
| homer18                   | 308    | 2030   | 2024              | 0.131      |
| homer19                   | 330    | 2340   | 2332              | 0.142      |
| homo20                    | 440    | 4220   | 4202              | 0.170      |
| k2f1igit2                     | 3771   | 270136 | 269918            | 47.860      |
| k2f1igit2 vaccination       | 5028   | 307674 | 307563            | 43.461      |
| k2f1igit2 vaccination2      | 9882   | 295998 | 295657            | 54.870      |
| kfl1digit2                 | 13176  | 345426 | 345230            | 39.450      |
| kfl1digit2 kfl2digit2       | 10056  | 271393 | 271296            | 17.291      |
| shal                      | 61377  | 255417 | 251863            | 54.40       |
| sha2                      | 61377  | 255417 | 251915            | 19.47       |
| rdeco/lf_block-JStblock    | 797340 | 119376 | 1100325           | 5.471       |
| rdeco/lf_dimacs            | 237763 | 939784 | 896327            | 16.880      |
| rdeco/lf_dimacsJStblck     | 1198012| 3865513| 3350748           | 11.820      |
| rdeco/lf_dimacsJStblck     | 1186710| 3829036| 3220154           | 37.064      |
| sfl1A3I3Istblock-lugly_dimacs | 870975 | 3812147| 3416609           | 43.833      |
| wu8ms01_dimacs            | 463080 | 1759150| 1624751           | 17.008      |
| wu8ms04_dimacs            | 463080 | 1759150| 1624017           | 4.333       |
| wu8ms0-problem_dimacs     | 2691648| 8517027 |7159756         | 56.460      |
| wu8ms0-problem_dimacs     | 2785108| 8812799 |7401876         | 43.930      |
| wuconnax1_dimacs          | 277950 | 1221020 |1168336          | 1.808      |

Times are given in minutes.
Table 2: Results for $H = \text{walksat-tabu}$. Times are given in minutes.

| Instance       | $n$ | $m$ | Genetic algorithm | walksat-tabu alone |
|----------------|-----|-----|-------------------|--------------------|
| $c3500m1$      | 5248| 33199| 33164             | 35106              |
| $c6298m1$      | 9540| 61421| 61392             | 61389              |
| $dalu4u1$      | 9426| 59991| 59896             | 59910              |
| $frg1m1$       | 3230| 20575| 20570             | 20570              |
| $k2m1$         | 11580| 74581| 74340             | 74341              |
| $x1m1$         | 8760| 55571| 55561             | 55562              |
| $an_{8, 6}$    | 2269| 7814 | 7809              | 7810               |
| $an_{7, 7}$    | 4264| 14751| 14732             | 14729              |
| $an_{9, 6}$    | 7361| 25538| 25466             | 25473              |
| $homer17$      | 11908| 41391| 41183             | 41189              |
| $li-exam-61$   | 28147| 108436| 107983          | 107982             |
| $li-exam-62$   | 28147| 108436| 107974          | 107992             |
| $li-exam-63$   | 28147| 108436| 107980          | 107978             |
| $li-exam-64$   | 28147| 108436| 107985          | 107985             |
| $li-test4-100$ | 36809| 142491| 141790          | 141782             |
| $li-test4-101$ | 36809| 142491| 141804          | 141800             |
| $li-test4-94$  | 36809| 142491| 141806          | 141780             |
| $li-test4-95$  | 36809| 142491| 141791          | 141782             |
| $li-test4-96$  | 36809| 142491| 141804          | 141782             |
| $li-test4-97$  | 36809| 142491| 141777          | 141791             |
| $li-test4-98$  | 36809| 142491| 141774          | 141781             |
| $li-test4-99$  | 36809| 142491| 141774          | 141781             |
| $gripper10u$   | 2312| 18666| 18662            | 18662              |
| $gripper11u$   | 3084| 26019| 26014            | 26014              |
| $gripper12u$   | 3352| 29412| 29406            | 29406              |
| $gripper13u$   | 268| 38965| 38959            | 38959              |
| $gripper14u$   | 4584| 43390| 43383            | 43382              |
| $bc56-sensors-1-x391-unsat$ | 561371| 1778987| 1600024          | 1599963             |
| $bc56-sensors-2-x592-unsat$ | 850398| 2944319| 2361372          | 2361225             |
| $bc57-sensors-1-x303-unsat$ | 435701| 1379987| 1264444          | 1264039             |
| $dne-03-1-k247-unsat$ | 261352| 77077| 740068           | 739924              |
| $motors-stuck-1-x407-unsat$ | 654766| 2068742| 1840274          | 1839998             |
| $motors-stuck-2-x314-unsat$ | 505536| 1306817| 1445995          | 1445961             |
| $valves-gates-1-x617-unsat$ | 985042| 3113540| 2705763          | 2705761             |
| $7pipe$        | 15800| 394739| 394727           | 394727              |
| $7pipe$        | 23910| 731118| 751110           | 751110              |
| $comb1$        | 5914| 16804| 16713            | 16717              |
| $dp21u1$       | 11137| 30792| 30775            | 30773              |
| $f2cl1m20$     | 34678| 101319| 100087           | 100075              |
| $fitfo300$     | 194762| 530713| 509196           | 509152              |
| $homer17$      | 286| 1732| 1738             | 1738               |
| $homer18$      | 308| 2030| 2024             | 2024               |
| $homer19$      | 330| 2430| 2332             | 2332               |
| $homer20$      | 440| 4220| 4202             | 4202               |
| $k2fi1$        | 3771| 270136| 269839           | 269835              |
| $k2fi1$        | 5028| 307674| 307490           | 307485              |
| $k2fi1m20$     | 9882| 299998| 299544           | 299554              |
| $k2fi1m20$     | 13176| 345426| 345065           | 345044              |
| $k2fi1m20$     | 10056| 271393| 271301           | 271301              |
| $shal$         | 61377| 255417| 251374          | 251364              |
| $shal$         | 61377| 255417| 251390          | 251390              |
| $rsdecoder6-dimacs$ | 767340| 1106376| 1028280         | 1028258             |
| $rsdecoder6-dimacs$ | 237763| 933975| 895865           | 895990              |
| $rsdecoder-problem.dimacs$ | 1198012| 3865513| 3334250         | 3334450             |
| $rsdecoder-problem.dimacs$ | 1186710| 3829036| 3304293         | 3304476             |
| $sUb111B BLACKBOX$ | 870975| 3812147| 3386126         | 3386203             |
| $ub_48m1$      | 463080| 1759150| 1611716         | 1611839             |
| $ub_6ms4$      | 463080| 1759150| 1610945         | 1611361             |
| $ub_8ms-problem.dimacs$ | 2691648| 8517027| 7132324         | 7133580             |
| $ub_8ms-problem.dimacs$ | 2785108| 8812799| 7374846         | 7374877             |
| $ub_comma1.dimacs$ | 277950| 1221020| 1156736        | 1156771             |
| $ub_comma2.dimacs$ | 277950| 1221020| 1156718        | 1156841             |
| Instance                      | $n$ | $m$ | Genetic algorithm  | adaptynovelty+ alone |
|-------------------------------|-----|-----|-------------------|---------------------|
| c360m1                        | 5248 | 33199 | 33182             | 32.030 |
| c628m1                        | 9540 | 61421 | 61382             | 61.382 |
| dalon1                        | 9426 | 59991 | 59930             | 59.920 |
| f8g1m1                        | 3230 | 20575 | 20574             | 0.527 |
| k2m1                          | 11680 | 74581 | 74417             | 52.360 |
| x1m1                          | 8760 | 55571 | 55570             | 55.570 |
| anu6                          | 2260 | 7814  | 7813              | 0.102 |
| anu2                          | 4264 | 14751 | 14750             | 14750 |
| gripper10u                    | 2312 | 18666 | 18665             | 18665 |
| gripper11u                    | 3084 | 26019 | 26018             | 26018 |
| gripper12u                    | 3352 | 29412 | 29410             | 29411 |
| gripper13u                    | 4268 | 38965 | 38963             | 38963 |
| bc56-sensors-1-k391-unsat     | 56137 | 177897 | 1623357          | 1623430 |
| bc56-sensors-2-k592-unsat     | 850298 | 2944319 | 2394308       | 2393994 |
| bc57-sensors-1-k303-unsat     | 435701 | 1379987 | 1282208          | 1282338 |
| dme-03-1-k247-unsat           | 261352 | 770377 | 746902           | 746935 |
| motors-stock-1-k407-unsat     | 654766 | 2068742 | 1866990         | 1867266 |
| motors-stock-2-k314-unsat     | 505536 | 1596817 | 1418989         | 141907 |
| valves-gates-1-k617-unsat     | 985042 | 3113540 | 2742641       | 2742310 |
| 6pipe                         | 15800 | 394739 | 393808           | 393771 |
| 7pipe                         | 23910 | 731118 | 749636           | 749636 |
| comb1                         | 5910  | 16804 | 16756            | 16759 |
| dp12u1                        | 11137 | 30792 | 30723            | 30722 |
| f2cl2                        | 34678 | 101319 | 100431           | 100435 |
| fifd3                        | 194762 | 530713 | 516316           | 516380 |
| homer17                       | 386   | 1742  | 1738             | 1738 |
| homer18                       | 308   | 2030  | 2024             | 2024 |
| homer19                       | 330   | 2340  | 2332             | 2332 |
| homer20                       | 440   | 4220  | 4202             | 4202 |
| k2f1                    | 37730 | 2701364 | 2699251        | 269930 |
| k2f1                    | 5028  | 307074 | 307560           | 307559 |
| k2f1                    | 9882  | 299998 | 2995466          | 299546 |
| k2f1                    | 13176 | 345246 | 345123           | 345150 |
| shal                          | 61377 | 255417 | 252622           | 252613 |
| lce                            | 61337 | 255417 | 252622           | 252613 |
| lce2                            | 61337 | 255417 | 252622           | 252613 |
| rddecodeR1_blackbox_J2block   | 707340 | 1096376 | 10924736        | 1092489 |
| rsdecode42dimacs              | 237763 | 933978 | 904911           | 904906 |
| rddecodeproblem_dimacs_R8     | 1198012 | 3865513 | 3374308        | 3374204 |
| rddecodeproblem_dimacs_R9     | 1186710 | 3829036 | 3343967        | 3343653 |
| rsdecodedimacs                | 879775 | 3812147 | 3431928          | 3431455 |
| rsdecodedimacs                | 463080 | 1759150 | 1637425         | 1637317 |
| rsdecodedimacs                | 463080 | 1759150 | 1637425         | 1637317 |
| rsdecodedimacs                | 2691648 | 8517027 | 7189090          | 7189224 |
| rsencodedimacs                | 2781508 | 8817999 | 7432769       | 7432463 |
| rsencodedimacs                | 277950 | 1221020 | 1175833       | 1175942 |

Table 3: Results for $H = \text{adaptnovelty}^+$. Times are given in minutes.
Table 4: Success ratios of the genetic algorithm as per Tables 1 through 3.

| Instance set                  | novelty | walksat-tabu | adapt+         |
|-------------------------------|---------|--------------|----------------|
| 2004 dataset                  | 0.540   | 0.667        | 0.588          |
| 2008 dataset                  | 0.545   | 0.000        | 0.545          |
| 2004 & 2008 datasets combined | 0.541   | 0.548        | 0.581          |

Table 5: Success ratios of the genetic algorithm over all 100 instances of the 2004 dataset and all 112 instances of the 2008 dataset.

| Instance set                  | novelty | walksat-tabu | adapt+         |
|-------------------------------|---------|--------------|----------------|
| 2004 dataset                  | 0.600   | 0.700        | 0.630          |
| 2008 dataset                  | 0.518   | 0.518        | 0.509          |
| 2004 & 2008 datasets combined | 0.557   | 0.604        | 0.566          |

can outperform the heuristics themselves when used alone. We believe further effort can be profitably spent on attempting similar solutions to other problems that, like MAX-SAT, can be expressed as an unconstrained optimization problem on binary variables. Some of them are the maximum independent set and minimum dominating set problems on graphs, both admitting well-known formulations of this type.

Acknowledgments

The authors acknowledge partial support from CNPq, CAPES, and a FAPERJ BBP grant.

References

[1] J. Argelich, C. M. Li, F. Manyà, and J. Planes. Third Max-SAT evaluation. URL http://www.maxsat.udl.cat/08/, 2008.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties. Springer-Verlag, Berlin, Germany, 1999.

[3] V. C. Barbosa. Massively Parallel Models of Computation. Ellis Horwood, Chichester, UK, 1993.

[4] V. C. Barbosa and E. Gafni. A distributed implementation of simulated annealing. J. Parallel Distrib. Comput., 6:411–434, 1989.

[5] E. Dantsin, M. Gavrilovich, E. Hirsch, and B. Konev. MAX SAT approximation beyond the limits of polynomial-time approximation. Ann. Pure Appl. Logic, 113:81–94, 2001.
[6] R. Dechter. *Constraint Processing*. Morgan Kaufmann, San Francisco, CA, 2003.

[7] E. Fink. How to solve it automatically: Selection among problem-solving methods. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, pages 128–136, Menlo Park, CA, 1998. AAAI Press.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[9] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-6:721–741, 1984.

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

[11] I. P. Gent and T. Walsh. Unsatisfied variables in local search. In J. Hallam, editor, *Hybrid Problems, Hybrid Solutions*, pages 73–85, Amsterdam, The Netherlands, 1995. IOS Press.

[12] C. P. Gomes and B. Selman. Algorithm portfolios. *Artif. Intell.*, 126:43–62, 2001.

[13] A. K. Hartmann and M. Weigt. *Phase Transitions in Combinatorial Optimization Problems: Basics, Algorithms and Statistical Mechanics*. Wiley-VCH, Weinheim, Germany, 2005.

[14] H. H. Hoos. An adaptive noise mechanism for WalkSAT. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 655–660, Menlo Park, CA, 2002. AAAI Press.

[15] F. Hutter, D. A. D. Tompkins, and H. H. Hoos. Scaling and probabilistic smoothing: Efficient dynamic local search for SAT. In P. van Hentenryck, editor, *Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes in Computer Science*, pages 233–248, Berlin, Germany, 2002. Springer-Verlag.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[17] M. G. Lagoudakis, M. L. Littman, and R. E. Parr. Selecting the right algorithm. In *Proceedings of the 2001 AAAI Fall Symposium Series: Using Uncertainty within Computation*, pages 74–75, Menlo Park, CA, 2001. AAAI Press.
[18] D. Le Berre and L. Simon. The SAT 2004 competition. In H. H. Hoos and D. G. Mitchell, editors, *Theory and Applications of Satisfiability Testing*, volume 3542 of *Lecture Notes in Computer Science*, pages 321–344, Berlin, Germany, 2005. Springer-Verlag.

[19] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, and Y. Shoham. Boosting as a metaphor for algorithm design. In F. Rossi, editor, *Principles and Practice of Constraint Programming*, volume 2833 of *Lecture Notes in Computer Science*, pages 899–903, Berlin, Germany, 2003. Springer-Verlag.

[20] V. Manquinho, J. Marques-Silva, and J. Planes. Algorithms for weighted Boolean optimization. In O. Kullmann, editor, *Theory and Applications of Satisfiability Testing*, volume 5584 of *Lecture Notes in Computer Science*, pages 495–508, Berlin, Germany, 2009. Springer-Verlag.

[21] B. Mazure, L. Sais, and E. Gregoire. Tabu search for SAT. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 281–285, Menlo Park, CA, 1997. AAAI Press.

[22] D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 321–326, Menlo Park, CA, 1997. AAAI Press.

[23] S. Minton. Automatically configuring constraint satisfaction programs: A case study. *Constraints*, 1:7–43, 1996.

[24] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, MA, 1996.

[25] J. R. Rice. The algorithm selection problem. In M. Rubinoff and M. C. Yovits, editors, *Advances in Computers*, volume 15, pages 65–118. Academic Press, New York, NY, 1976.

[26] S. Russell and D. Subramanian. Provably bounded-optimal agents. *J. Artif. Intell. Res.*, 2:575–609, 1995.

[27] B. Selman and H. Kautz. Domain-independent extensions to GSAT: Solving large structured variables. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 290–295, San Mateo, CA, 1993. Morgan Kaufmann.

[28] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 459–465, Menlo Park, CA, 1992. AAAI Press.

[29] D. A. D. Tompkins and H. H. Hoos. Warped landscapes and random acts of SAT solving. In *Proceedings of the Eighth International Symposium on Artificial Intelligence and Mathematics*, Piscataway, NJ, 2004. RUTCOR.
[30] D. A. D. Tompkins and H. H. Hoos. UBCSAT: An implementation and experimentation environment for SLS algorithms for SAT and MAX-SAT. In H. H. Hoos and D. G. Mitchell, editors, Theory and Applications of Satisfiability Testing, volume 3542 of Lecture Notes in Computer Science, pages 306–320, Berlin, Germany, 2005. Springer-Verlag.

[31] V. Vassilevska, R. Williams, and S. L. M. Woo. Confronting hardness using a hybrid approach. In Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 1–10, New York, NY, 2006. ACM Press.