noSAT-MaxSATv3

Ole Lübke
Institute for Software Systems
Hamburg University of Technology (TUHH)
Hamburg, Germany
ole.luebke@tuhh.de

***Abstract*—Using a SAT solver to solve MaxSAT is a well-established and efficient method. However, in resource-constrained computing environments, e.g., embedded systems, such algorithm designs can be hard to realize. With *noSAT-MaxSAT*, we explore alternatives that do not rely on an external, dedicated SAT solver. Compared to the previous version, the most notable changes in *noSAT-MaxSATv3* include an update of the base algorithm to NuWLS-2.0, new rules for restarting, the integration of a new Frequency Fitness Assignment-inspired variable selection heuristic in local optima, and the inclusion of fixed-precision integer arithmetic to increase precision in weight calculations without resorting to floating-point numbers.**

I. Introduction

Exact as well as anytime MaxSAT solvers usually include a SAT solver. For stochastic local search (SLS) algorithms, a SAT solver can provide a feasible model as a starting point [1]; linear search algorithms query a SAT solver iteratively; core-guided and implicit hitting set-based algorithms use a SAT solver to extract unsatisfiable subsets of the clauses of the formula (cores) [2]. Our goal is to develop an efficient MaxSAT solver that is suitable for deployment in resource-constrained computing environments. Consequently, the solver is developed under a set of constraints commonly found in embedded systems programming, outlined in section II, which practically rule out the inclusion of any complete SAT solving algorithm.

Compared to *noSAT-MaxSATv2* [3], in *noSAT-MaxSATv3* we removed the 2-SAT preprocessing step, which turned out to be effective only for very few instances; updated the base algorithm to the NuWLS-2.0 SLS algorithm [4]; introduced a new variable-selection heuristic for local optima inspired by Frequency Fitness Assignment (FFA) [5, 6]; improved precision in weight calculations using fixed-precision integer arithmetic; implemented new conditions for restarting. More details on these changes are provided in section III.

Finally, for comparison, we paired *noSAT-MaxSATv3* with the *CaDiCaL 1.9.5* SAT solver [7], resulting in the solver *noSAT-MaxSATv3-c*. It executes *CaDiCaL* once to obtain a feasible initial assignment for SLS (or determine unsatisfiability).

II. Requirements & Architecture

noSAT-MaxSAT is developed under the following requirements commonly found in programming embedded systems: It 1) is programmed in C; 2) is self-contained, i.e., it has no external dependencies (other than the C standard library); 3) does not allocate memory dynamically; 4) does not use floating-point operations; 5) does not contain unbounded loops, i.e., the maximum number of iterations of any loop is known and constant when entering it.

noSAT-MaxSAT is split into a library that fulfills these requirements, and an application that uses the library but is not bound by the above-mentioned restrictions to facilitate, e.g., parsing input instances of arbitrary size. Given the number of variables and clauses of a formula, the library can compute (an upper bound on) the amount of memory required for running the algorithm.

III. Implementation

Algorithm 1 noSAT-MaxSATv3

```
Require: Variable values are initialized (e.g., randomly)
 1: bestcost ← ∞; skip ← false; badtries ← 0
 2: for try ← 1 ... maxTries do
 3:     if ¬skip then
 4:         INITVARS(badtries ≥ maxBadtries)
 5:         DECIMATION
 6:         INIT
 7:     end if
 8:     skip ← false; impr ← false
 9:     for flip ← 1 ... maxFlips do
10:         impr ← current cost < bestcost
11:         skip ← skip ∨ impr
12:         v ← SELECTVARIABLE
13:         FLIPVARIABLE(v)
14:     end for
15:     impr ← current cost < bestcost
16:     skip ← skip ∨ impr
17:     badtries ← if impr then 0 else badtries + 1
18: end for
```

Algorithm 1 shows how NuWLS-2.0 [4] is implemented in *noSAT-MaxSATv3*, highlighting the new restart rules. In the following, we elaborate on these rules, the new variable selection heuristic, and the use of fixed-precision integer arithmetic.

A. New Restart Rules

In line 3 of Algorithm 1, the initialization block is only entered when no cost-improving assignment has been found

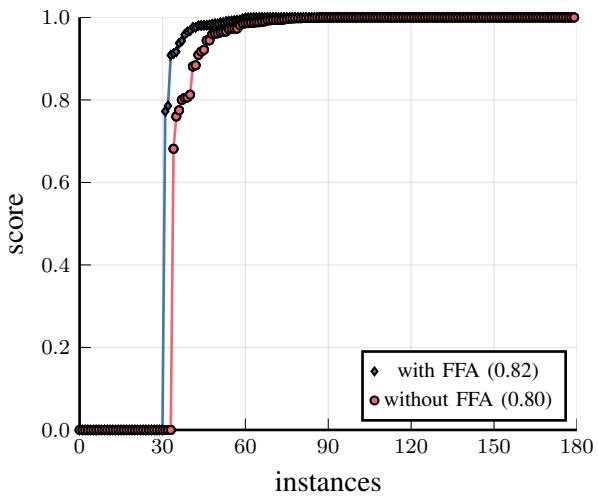

Fig. 1. Comparison of *noSAT-MaxSATv3* with and without the FFA-based variable selection heuristic. $score(s,i) = \frac{1+\min_{z\in\{\text{with FFA, without FFA}\}} o(z,i)}{1+o(s,i)}$, where $o(s,i)$ denotes the best objective value found by solver s on instance i within 60 seconds. The benchmark instances have been selected randomly from the incomplete/anytime weighted tracks of the MaxSAT Evaluations 2020 – 2023.

during the last iteration of the outer loop. If so, in line 4, a random variable assignment is generated when no cost-improving assignment has been found for $maxBadtries$ iterations of the outer loop. When there are any falsified hard clauses, Algorithm 1 is executed on all hard clauses to find a feasible initial assignment. In line 5, decimation [8] is used to improve the assignment.

The original implementation of NuWLS-2.0 [4] effectively resets $flip$ to 0 whenever a better assignment is found, to continue searching in the same, promising direction. Resetting loop counters is not allowed in *noSAT-MaxSAT* due to requirement 5. Instead, the $skip$ flag is set to skip the initialization procedures for the next try and continue searching in the same direction. When no improvement was found for $maxBadtries$ tries, Algorithm 1 restarts with a random assignment to explore different areas of the search space.

B. Variable Selection Inspired by FFA

Frequency Fitness Assignment (FFA) is a heuristic used in evolutionary algorithms which prefers individuals with rarely-encountered fitness values over individuals with high fitness values [5]. In SLS algorithms for MaxSAT such as NuWLS, usually the variable with the highest *score* is selected for flipping, where the score indicates how much the total cost of the current assignment improves when the variable is flipped. With FFA we would select the variable with the least-frequently encountered score instead. However, maintaining a dictionary of score values and their frequencies can impose a significant overhead. In contrast, an array of variable flip counts can be maintained with negligible, constant overhead, and selecting rarely-flipped variables over variables with a high score can actually lead to discovering better solutions [6]. Therefore, whenever there are no variables with a positive score (i.e., a local optimum is reached), the clause weights are updated, and a falsified clause is selected randomly (preferring hard clauses) as in NuWLS-2.0 [4]. Then, instead of selecting the variable with the least bad score, we select the least frequently flipped variable from that clause. Figure 1 shows that *noSAT-MaxSATv3* performs slightly better with the FFA-based heuristic.

C. Fixed-Precision Integer Arithmetic

The weighting scheme of NuWLS-2.0 [4] requires calculating the average weight of all soft clauses. In contrast to the original implementation, we cannot use floating-point numbers due to requirement 4. However, we found that the loss of precision under usual truncating integer arithmetic can severely degrade the performance of the solver. As a remedy, we scale all values involved in weight and score computations by left-shifting by x bits (multiplying with 2^x) where $x = 64 - b_{reserved} - b_{weightsum}$, $b_{reserved}$ is a number of reserved bits to prevent overflows (currently 32), and $b_{weightsum}$ is the number of bits required to store the sum of all weights.

References

[1] Zhendong Lei and Shaowai Cai. "SATLike-c: Solver Description". In: *MaxSAT Evaluation 2020: Solver and Benchmark Descriptions*. 2020, p. 23.

[2] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. "Maximum Satisfiability". In: *Handbook of Satisfiability*. 2nd ed. Vol. 336. Front. Artif. Intell. Appl. 2021, pp. 929–991. DOI: 10.3233/FAIA201008.

[3] Ole Lübke and Sibylle Schupp. "noSAT-MaxSATv2". In: *MaxSAT Evaluation 2023: Solver and Benchmark Descriptions*. 2023, pp. 27–28.

[4] Yi Chu, Shaowei Cai, and Chuan Luo. "NuWLS-c-2023: Solver Description". In: *MaxSAT Evaluation 2023: Solver and Benchmark Descriptions*. 2023, pp. 23–24.

[5] Thomas Weise, Mingxu Wan, Pu Wang, Ke Tang, Alexandre Devert, and Xin Yao. "Frequency Fitness Assignment". In: *IEEE Trans. Evol. Comp.* 18.2 (2014), pp. 226–243. DOI: 10.1109/TEVC.2013.2251885.

[6] Thorge Becker. "Application of Frequency Fitness Assignment for Solving MaxSAT Problems". Project Thesis. Hamburg University of Technology, 2023. URL: https://www.sts.tuhh.de/pw-and-m-theses/2023/becker23.pdf.

[7] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. "CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020". In: *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*. 2020, pp. 51–53.

[8] Shaowei Cai, Chuan Luo, and Haochen Zhang. "From Decimation to Local Search and Back: A New Approach to MaxSAT". In: *Proc. 26th Int. Jt. Conf. Artif. Intell.* 2017, pp. 571–577. DOI: 10.24963/ijcai.2017/80.