NuWLS: Improving Local Search for (Weighted) Partial MaxSAT by New Weighting Techniques

Yi Chu[1], **Shaowei Cai**[2,3*], **Chuan Luo**[4]
[1]Institute of Software, Chinese Academy of Sciences, China
[2]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China
[3]School of Computer Science and Technology, University of Chinese Academy of Sciences, China
[4]School of Software, Beihang University, China
{chuyi,caisw}@ios.ac.cn, chuanluo@buaa.edu.cn

Abstract

Maximum Satisfiability (MaxSAT) is a prototypical constraint optimization problem, and its generalized version is the (Weighted) Partial MaxSAT problem, denoted as (W)-PMS, which deals with hard and soft clauses. Considerable progress has been made on stochastic local search (SLS) algorithms for solving (W)PMS, which mainly focus on clause weighting techniques. In this work, we identify two issues of existing clause weighting techniques for (W)PMS, and propose two ideas correspondingly. First, we observe that the initial values of soft clause weights have a big effect on the performance of the SLS solver for solving (W)PMS, and propose a weight initialization method. Second, we propose a new clause weighting scheme that for the first time employs different conditions for updating hard and soft clause weights. Based on these two ideas, we develop a new SLS solver for (W)PMS named NuWLS. Through extensive experiments, NuWLS performs much better than existing SLS solvers on all 6 benchmarks from the incomplete tracks of MaxSAT Evaluations (MSEs) 2019, 2020, and 2021. In terms of the number of winning instances, NuWLS outperforms state-of-the-art SAT-based incomplete solvers on all the 6 benchmarks. More encouragingly, a hybrid solver that combines NuWLS and an SAT-based solver won all four categories in the incomplete track of the MaxSAT Evaluation 2022.

Introduction

The maximum satisfiability (MaxSAT) problem is an optimization variant of the influential Boolean satisfiability (SAT) problem, and plays an important role in computer science. Given a propositional formula in conjunctive normal form (CNF), the objective of MaxSAT is to find an assignment that maximizes the number of satisfied clauses. The partial MaxSAT (PMS) problem is a variant of MaxSAT: In a partial CNF formula, clauses are divided into hard ones and soft ones, and the task is to find an assignment that satisfies all hard clauses and maximizes the number of satisfied soft clauses. In the weighted partial MaxSAT (WPMS) problem, each soft clause is assigned a positive integer as its weight, and the goal is to seek an assignment that satisfies all hard clauses and maximizes the total weight of satisfied soft clauses.

Real-world combinatorial problems usually involve both hard and soft constraints, so encoding such problems into (W)PMS is more flexible than encoding them into SAT and MaxSAT. Indeed, (W)PMS has been applied to solving many real-world problems, including scheduling (Thornton and Sattar 1998), timetabling (Cha et al. 1997), set covering (Naji-Azimi, Toth, and Galli 2010), computational protein design (Allouche et al. 2012), and FPGA routing (Fu and Malik 2006).

Related Work

There are two major categories of algorithms for solving (W)PMS: complete algorithms and incomplete algorithms. Complete algorithms can prove the optimality of the solutions. Among all complete methods, the method of iteratively calling SAT solvers (commonly referred to as the SAT-based method) has experienced significant improvement. The core-guided algorithm was first introduced for solving PMS in (Fu and Malik 2006), then the core-guided method was used for solving WPMS in (Ansótegui, Bonet, and Levy 2009; Ansótegui, Bonet, and Levy 2010). In recent years, much attention has been focused upon the improvement of the SAT-based method for (W)PMS, which has resulted in much excellent work (Davies and Bacchus 2011; Koshimura et al. 2012; Ansótegui, Bonet, and Levy 2013; Narodytska and Bacchus 2014; Martins et al. 2014; Ansótegui and Gabàs 2017; Berg, Demirovic, and Stuckey 2019; Joshi et al. 2019; Nadel 2019, 2020). In addition, in the literature (Li et al. 2021), the authors propose a branch-and-bound (BnB) MaxSAT solver named MaxCDCL by combining clause learning and an efficient bounding procedure significantly increases the speed of the BnB MaxSAT solvers.

Although incomplete algorithms, mainly stochastic local search (SLS) ones, cannot prove the optimality of the solutions, they are very useful for many industrial problems where the objective is not to find an optimal assignment but aims to seek a good-quality assignment within a reasonable time (Luo et al. 2015). Interestingly, there is a gold line through previous SLS solvers for (W)PMS, which involves methods to maintain the clause weights. Essentially, SLS algorithms introduce weights to clauses (rather than the original weights), and guide the search mainly by maintaining the weights. An early SLS method for PMS is a weighted ver-

*Corresponding author

sion of WalkSAT (Jiang, Kautz, and Selman 1995), in which the weight of each hard clause is set to the number of all soft clauses plus one. After that, Cha et al. observed that too large hard clause weights limit the search area of SLS algorithms for solving PMS, and proposed to set the weights of hard clauses to a hand-tuned optimal value (Cha et al. 1997). Thornton et al. further proposed two clause weighting strategies where the weight differential between hard and soft clauses is dynamically adjusted during the search (Thornton and Sattar 1998; Thornton et al. 2002).

A milestone SLS algorithm for PMS is Dist, which separates hard and soft scores, and employs a clause weighting scheme for hard clauses (Cai et al. 2014), leading to significant improvements over previous SLS methods. Three variants based on Dist are developed, including DistUP (Cai et al. 2016), CCEHC (Luo et al. 2017), and NuDist (Lei and Cai 2020), where DistUP is designed for PMS while CCEHC is designed for WPMS, and NuDist performs better than the former two algorithms on both PMS and WPMS.

Recently, SATLike (Lei and Cai 2018) made a breakthrough in the incomplete solving of (W)PMS. SATLike proposes a new clause weighting scheme for both hard and soft clauses, where a limit on the maximum weight is set for each soft clause. SATLike performs much better than the Dist series on benchmarks of recent MaxSAT Evaluations, and the latest version SATLike3.0 (Cai and Lei 2020) represents the state-of-the-art SLS algorithm for (W)PMS. Additionally, BandMaxSAT enhances the performance of SATLike3.0 by guiding the search direction with a multi-armed bandit (Zheng et al. 2022).

In addition, researchers modify complete algorithms into incomplete ones by presenting the best solution found in real time. Loandra (Berg, Demirovic, and Stuckey 2019) and TT-Open-WBO-Inc (Nadel 2020) are two examples of SAT-based incomplete solvers of this type. It has been observed that SLS solvers and SAT-based solvers are complementary in solving (W)PMS benchmarks. This is also witnessed by the following facts: A hybrid solver named SATLike-c combining SATLike3.0 and TT-Open-WBO-Inc won the unweighted categories in the incomplete track of MaxSAT Evaluation 2021 (Lei et al. 2021); another hybrid solver named DT-HyWalk combining BandMaxSAT and TT-Open-WBO-Inc achieves the second place on the unweighted categories in the incomplete track of MaxSAT Evaluation 2022.[1]

Contributions

In this paper, we further study the clause weighting techniques for (W)PMS, and develop a more effective SLS solver. First, we identify an important issue that has been ignored in previous clause weighting methods. Our preliminary experiments show that the performance of SATLike3.0 varies as we change the initial weights of soft clauses. Specifically, we find that the value of initial weights of soft clauses has a clear effect on the performance of SATLike3.0. Based on this observation, we design a method to initialize soft clause weights.

[1]https://maxsat-evaluations.github.io/2022/rankings.html

Second, we propose a method to update clause weights during the local search, which makes a deeper distinction between hard and soft clauses. In the Dist series, the weighting schemes are applied only to hard clauses. For the first time, SATLike introduces a weighting scheme that works in different ways for hard and soft clauses. Although SATLike sets a limit on the maximum weight of soft clauses, it adopts the same activation condition for updating the weights of hard and soft clauses. In our proposed clause weighting scheme, the hard clause weights and soft clause weights are updated according to different activation conditions. Additionally, the maximum limit of the weights of soft clauses in our weighting scheme is related to the original weights of the soft clauses.

Based on these two ideas, we develop a new SLS algorithm named NuWLS (New Weighting based Local Search). We compare NuWLS with 4 state-of-the-art solvers (2 SLS ones and 2 SAT-based incomplete ones) on all 6 benchmarks from the incomplete tracks of MSEs 2019, 2020, and 2021. The experimental results show that NuWLS significantly outperforms other SLS solvers, and outperforms SAT-based solvers in terms of the number of winning instances on all 6 benchmarks. Furthermore, NuWLS can considerably enhance the complementary between SLS solvers and SAT-based solvers, and improve the performance of the hybrid solver in solving the (W)PMS benchmarks. In the incomplete track of MaxSAT Evaluation 2022, the hybrid solver that combines NuWLS and TT-Open-WBO-Inc won all 4 categories. In addition, our results indicate the effectiveness of the clause weights initialization method and the new clause weighting scheme.

Preliminaries

Given a set of n Boolean variables $V = \{x_1, x_2, \cdots, x_n\}$, and the set of $2n$ literals corresponding to these variables $L = \{x_1, \neg x_1, x_2, \neg x_2, \cdots, x_n, \neg x_n\}$, where a literal is either a variable or its negation. A clause is a disjunction of literals, *i.e.,* $c_i = l_{i_1} \vee l_{i_2}, \cdots, \vee l_{i_k}$, where k is the length of clause c_i. A CNF formula F is a conjunction of clauses, *i.e.,* $F = c_1 \wedge c_2 \wedge \cdots \wedge c_m$, where m is the number of clauses in F. An assignment is a mapping that assigns a Boolean value (*True* or *False*) to each variable. Given an assignment α, a clause c is satisfied if at least one literal in c is *True*, otherwise c is falsified.

Given a partial CNF formula F, where all clauses are divided into hard ones and soft ones, the objective of the PMS problem is to find an assignment that satisfies all hard clauses and maximizes the total number of satisfied soft clauses. For a weighted partial CNF formula F, where each soft clause c is associated with a positive integer $w_{ori}(c)$ as its weight, the objective of the WPMS problem is to find an assignment that satisfies all hard clauses and maximizes the total weight of satisfied soft clauses.

Given a (weighted) partial CNF formula F and an assignment α, if α satisfies all the hard clauses in F, then we say α is a feasible assignment (*i.e.,* feasible solution, solution for short). For PMS, notation $cost(\alpha)$ denotes the total number of falsified soft clauses under the assignment α. For WPMS,

$cost(\alpha)$ denotes the total weight of falsified soft clauses under α. We say a solution α_1 is better than another solution α_2, if $cost(\alpha_1) < cost(\alpha_2)$.

In local search algorithms for (W)PMS, given an assignment α, the algorithms use the $score(x)$ of each variable x to search. For a variable x, we use $make(x)$ to denote the total weight of falsified clauses that would become satisfied by flipping x, and $break(x)$ to denote the total weight of satisfied clauses that would become falsified by flipping x. Then, for a variable x, its score is $score(x)$=$make(x)$-$break(x)$, and x is called a decreasing variable if $score(x)$>0.

Given a (weighted) partial CNF formula F, local search algorithms for (W)PMS, which use weighting techniques, maintain two types of weights for each clause. For a clause c, $w_{ori}(c)$ denotes its original clause weight, which is given in F. The original clause weight is used to calculate $cost(\alpha)$. For a clause c, $w(c)$ denotes c's weight, which is used to calculate each variable's score. The algorithms need to initialize $w(c)$ of each clause c, and $w_{init}(c)$ denotes the initial clause weight of c. Also, avg_{oriw} denotes the average original soft clause weights, and avg_{initw} denotes the average initial soft clause weights.

Main Ideas

Previous SLS algorithms for (W)PMS did not consider the effect of initial soft clause weights. In this section, we first conduct an empirical study to analyze the effect of initial soft clause weights on the performance of the SLS algorithm. Second, a novel clause weights initialization method is proposed based on the empirical analysis. Third, we introduce a new, effective clause weighting scheme.

Empirical Study on Initial Soft Clause Weights

In our preliminary experiments, we observe that the performance of the SLS algorithm varies when we change the initial soft clause weights. For some (W)PMS instances, setting initial soft clause weights to values greater than original weights could significantly improve the performance of SLS, while for some other (W)PMS instances, initial soft clause weights that are smaller than original weights are preferable. This interesting phenomenon motivates us to carry out experiments to comprehensively study the effect of the initial setting of soft clause weights on the performance of SLS.

Experimental Methodology. Given a (W)PMS instance F, for each soft clause c in F, most SLS algorithms directly use c's original weight as its initial weight, *i.e.,* $w_{init}(c) = w_{ori}(c)$. To study the effect of initial soft clause weights on the performance of SLS, it is advisable to evaluate the performance of SLS that sets each soft clause's initial weight to various values, *e.g.,* the original soft clause weight multiplied by different factors. For soft clause c, if factor equals to 1, then c's initial weight equals to its original weight, as most existing SLS algorithms do; if factor is smaller than 1, then c's initial weight is smaller than its original weight; otherwise (factor is greater than 1), then c's initial weight is greater than its original weight. After all experiments are finished, we can compare the performance with regard to different factors. Through this way, we can analyze the effect of different values of initial soft clause weights on the performance of SLS.

Experimental Setup. For this empirical study, we adopt two benchmarks, *i.e.,* the PMS and WPMS benchmarks from the incomplete track of MaxSAT Evaluation 2020. As introduced before, since SATLike3.0 stands for one of the state-of-the-art SLS algorithms for (W)PMS (Cai and Lei 2020), we adopt SATLike3.0 as the studied SLS algorithm.

Specifically, for each soft clause c, SATLike3.0 sets c's initial weight to c's original weight, *i.e.,* $w_{init}(c) = w_{ori}(c)$. To analyze the effect of initial soft clause weight, we modify SATLike3.0 by setting $w_{init}(c)$ to $w_{ori}(c)$ multiplied by a factor λ, as discussed in the preceding subsection, *i.e.,* $w_{init}(c) = w_{ori}(c) \times \lambda$. To comprehensively analyze the effect of λ with different values, in our empirical study it is advisable to set λ to diverse values. For the WPMS benchmark, λ's value domain is $\Gamma_W = \{$1e-10, 1e-9, $\cdots$, 1, 10, $\cdots$, 1e+5, 1e+6$\}$. For each value $\gamma \in \Gamma_W$, SATLike3.0 is evaluated by setting $\lambda = \gamma$ on all WPMS instances.

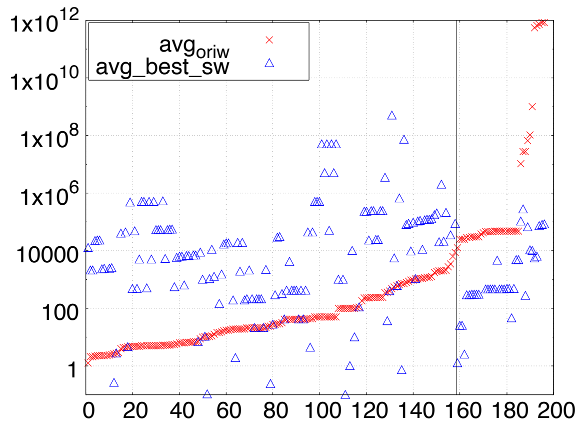
For PMS instances, it is clear that the average original soft clause weight of PMS instances is 1, which is much smaller than that figure of WPMS instances intuitively. Thus, for PMS instances, we only set the initial soft clause weights to the values not less than the original weights, so λ's value domain is $\Gamma_P = \{$1, 10, $\cdots$, 1e+5$\}$. For each value $\gamma \in \Gamma_P$, SATLike3.0 is evaluated with $\lambda = \gamma$ on all PMS instances.

In each experiment of the PMS and WPMS benchmarks, we use λ^* to denote the optimal value from λ's value domain that makes SATLike3.0 achieve the best performance, and use notation avg_best_sw to denote the optimal setting of average initial soft clause weights, *i.e.,* avg_best_sw = $avg_{oriw} \times \lambda^*$, recalling that avg_{oriw} denotes the average original soft clause weights.

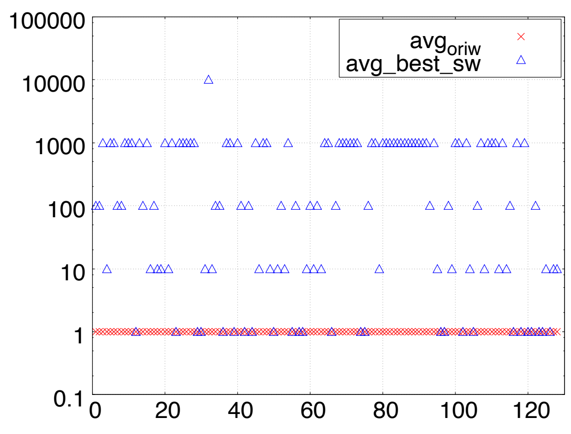
Observations and Insights. The results of this empirical study are illustrated in Figure 1. Figures 1(a) and 1(b) present the results on WPMS and PMS benchmarks, respectively (We ignore the instances in which SATLike3.0 could not find solutions). For each of Figures 1(a) and 1(b), the x-axis indicates the instances in the corresponding benchmark, which are sorted with the value of avg_{oriw} (*i.e.,* the average original soft clause weights) in ascending order. The y-axis presents the average value of clause weight. In both Figures 1(a) and 1(b), each instance is represented by two points with the same x-axis value. For each instance, the y-axis value of the *red x-cross shaped point* represents its value of avg_{oriw}, while the y-axis value of the *blue triangle shaped point* represents its values of avg_best_sw.

As can be observed in Figure 1(a), for those WPMS instances with relatively small-valued avg_{oriw}, setting initial soft clause weights greater than original weights results in better performance. On the other hand, for those WPMS instances with relatively large-valued avg_{oriw}, reducing initial soft clause weights achieves improvement. Also, according to Figure 1(b), for solving PMS instances, enlarging initial soft clause weights can lead to enhancement.

Hence, we summarize our insights as follows:



(a) WPMS Benchmark



(b) PMS Benchmark

Figure 1: The empirical analysis regarding the effect of initial soft clause weights on the performance of SLS for WPMS and PMS.

- Given a WPMS instance F, if F's average original soft clause weight is relatively large, then initializing soft clause weights as the values that are smaller than original weights is preferable; otherwise, enlarging initial soft clause weights based on original weights is preferable.
- For solving PMS, initializing soft clause weights as large-valued numbers can boost SLS algorithms.

Clause Weights Initialization Method

Based on the above empirical analysis, we propose a method for initializing clause weights. As observed from the empirical analysis, it is desirable to maintain the initial weights of soft clauses within a reasonable range. We propose a strategy to set the soft clause weights so as to make the average value of them reaches a preset parameter s_{avg}. A simple way to this end is to set

$$w_{init}(c) = w_{ori}(c) \times \frac{s_{avg}}{avg_{oriw}}$$

for each soft clause c.

Our local search algorithm restarts every L steps, and when it restarts, α and the clause weights are reset. In our clause weights initialization method, the hard clause weights are always initialized to 1. For the soft clauses, we distinguish two cases for the clause weights initialization in each round: If the algorithm finds a solution in the last round of local search, then the weight of each soft clause c is initialized to $w_{init}(c)$; otherwise, the weight of each soft clause c is initialized to 1, in order to put the priority on finding solutions. The weight of each soft clause c is initialized to 1 in the first round of local search. This procedure is called $init\_weight()$.

A New Clause Weighting Scheme

In this subsection, we propose a new clause weighting scheme named Dist-Weighting. Clause weighting is a popular diversification technique for SLS algorithms for (W)-PMS. Usually, SLS algorithms update clause weights when encountering local optima. Nevertheless, previous clause weighting schemes either update only the hard clause weights (Cai et al. 2014, 2016; Luo et al. 2017), or they update both hard and soft clause weights under the same conditions (Lei and Cai 2018; Cai and Lei 2020; Zheng et al. 2022). We propose to update hard and soft clause weights according to different activation conditions. Specifically, our new scheme is based on the following two ideas regarding both hard and soft clauses.

(1) For hard clauses: Since it is unreasonable to reduce the hard clause weights when the algorithm has not found any solutions yet, we introduce an additional condition for smoothing hard clause weights. Specifically, with a probability h_sp, if the algorithm has already found a solution in the current round of local search, then it smooths the hard clause weights.

(2) For soft clauses: When the cost of the current assignment α is smaller than that of the best solution α^* found so far (*i.e.*, $cost(\alpha) < cost(\alpha^*)$), the soft clause weights will not be updated since we encourage the algorithm to keep on searching around α. If the current assignment α is infeasible, the soft clause weights will not be increased. In addition, for each soft clause c, its maximum weight limit is set to $w_{init}(c) + \delta$.

Finally, we present our **Dist-Weighting** (Distinctively Weighting) scheme. The clause weights are updated when the algorithm encounters a local optimum, as in algorithm 1.

The NuWLS Algorithm

Based on the ideas in the preceding section, we develop a new SLS algorithm named NuWLS (New Weighting based Local Search). The pseudo-code of NuWLS is outlined in algorithm 2. We use α^* and $cost^*$ to denote the best found solution so far and its cost, respectively, and α denotes the current assignment which is maintained during the search.

In the beginning, α^* is empty and $cost^*$ is initialized as $+\infty$. NuWLS then iteratively calls the local search process

Algorithm 1: Dist-Weighting

```
Input: the current assignment $\alpha$; the best found
       solution so far $\alpha^*$.
1  prob := A random real number between 0 and 1;
2  if $prob < h\_sp$ and a solution has been found in the
     current round of local search then
3      for each satisfied hard clause $c$, $w(c)$ :=
         $w(c)$-$h\_inc$ if $w(c) > h\_inc$;
4  else
5      for each falsified hard clause $c$, $w(c)$ :=
         $w(c)$+$h\_inc$;
6  if $cost(\alpha) \geq cost(\alpha^*)$ then
7      if $prob < s\_sp$ then
8          for each satisfied soft clause $c$, $w(c)$ :=
             $w(c)$-$s\_inc$ if $w(c) > s\_inc$;
9      else if current assignment $\alpha$ is feasible then
10         for each falsified soft clause $c$,
             $w(c)$:=$w(c)$+$s\_inc$ if $w(c) < w_{init}(c)$+$\delta$;
11 return;
```

until reaching a preset time limit (lines 3-20). In the local search process, an initial complete assignment is generated by a unit propagation based procedure (Cai and Lei 2020) (line 3). NuWLS then calls $init\_weight$ to initialize the clause weights. (Note that the original weights in WPMS instances are only used in calculating the *cost* of assignments, and the weights used in guiding the search are always the ones introduced by the algorithm.) Following the initialization phase, NuWLS performs the search steps (lines 6-20). During the search, whenever the algorithm finds a solution with a lower cost than $cost^*$, then α^* and $cost^*$ are updated accordingly.

In each search step, NuWLS selects a variable and flips its value based on two situations. (1) If the set D of decreasing variables (*i.e.,* the variable x with $score(x)>0$) is not empty, a variable is selected from D. Since there may be many elements in D and traversing all of them would be time-consuming, a sampling strategy called BMS (Best from Multiple Selections) (Cai 2015) is adopted here. Through BMS, t variables are selected at random from the D set (t is a parameter in BMS) and then the variable with the highest score is chosen to flip. (2) When D is empty, which indicates that the search is stuck in a local optimum, NuWLS updates the weights of clauses according to the Dist-Weighting scheme. Then NuWLS randomly selects a falsified clause c, and picks the variable with the highest score from the selected clause c.

Finally, when the cutoff time is exceeded, NuWLS reports α^* and $cost^*$ if a solution is found; otherwise it reports "No solution found".

Experimental Evaluations

We conduct experiments to compare NuWLS against state-of-the-art solvers on extensive PMS and WPMS instances.

Algorithm 2: NuWLS

```
Input: (W)PMS instance $F$, cutoff.
Output: The best solution found and its $cost$, or "No
        solution found".
1  $\alpha^* := \emptyset$; $cost^* := +\infty$;
2  while elapsed time<cutoff do
3      $\alpha$ := an initial complete assignment;
4      $init\_weight()$;
5      $L$ = 10000000;
6      for step = 0; step < L; step++ do
7          if $\alpha$ is feasible and $cost^* > cost(\alpha)$ then
8              $\alpha^*$:=$\alpha$; $cost^*$:=$cost(\alpha)$; $L$=step+10000000;
9              if $cost^*$==0 then
10                 return $\alpha^*$ and $cost^*$;
11         if $(D = \{x|score(x) > 0\}) \neq \emptyset$ then
12             $v$ := a variable in $D$ selected by BMS strategy;
13         else
14             update clause weights by Dist-Weighting;
15             if $\exists$ falsified hard clauses then
16                 $c$ := a random falsified hard clause;
17             else
18                 $c$ := a random falsified soft clause;
19             $v$ := the variable with highest score in $c$;
20         $\alpha := \alpha$ with $v$ flipped;
21 if $\alpha^* \neq \emptyset$ then return $\alpha^*$ and $cost^*$;
22 else return No solution found;
```

Experimental Preliminaries

Benchmarks. We evaluate NuWLS on 6 benchmarks, *i.e.,* PMS and WPMS benchmarks from the incomplete tracks of MaxSAT Evaluations (MSEs) 2019, 2020, and 2021, denoted by `PMS_2019`, `PMS_2020`, `PMS_2021`, `WPMS_2019`, `WPMS_2020`, and `WPMS_2021`, respectively.

Competitors. NuWLS is compared with 4 state-of-the-art non-hybrid incomplete solvers: 2 SLS solvers (*i.e.,* BandMaxSAT and SATLike3.0) and 2 SAT-based solvers (*i.e.,* Loandra and TT-Open-WBO-Inc). BandMaxSAT and SATLike3.0 represent the state-of-the-art SLS solvers for (W)-PMS. The source codes of these two SLS solvers are publicly available.[2,3] Loandra is the best solver for PMS from the incomplete track of MSE 2019,[4] and performs best among the non-hybrid solvers in the incomplete track of MSE 2021.[5] TT-Open-WBO-Inc is the most competitive non-hybrid solver for PMS from the incomplete track of MSE 2020, and it is the best solver for solving WPMS from the incomplete tracks of MSEs 2019 and 2020.[4,6] For the SAT-based solvers, their source codes are downloaded from MSE 2020.[7]

[2]https://github.com/JHL-HUST/BandMaxSAT

[3]http://lcs.ios.ac.cn/%7ecaisw/MaxSAT.html

[4]https://maxsat-evaluations.github.io/2019/rankings.html

[5]https://maxsat-evaluations.github.io/2021/rankings.html

[6]https://maxsat-evaluations.github.io/2020/rankings.html

[7]https://maxsat-evaluations.github.io/2020/descriptions.html

Parameter Settings. There are 7 parameters in NuWLS: (1) s_{avg}, the average initial soft clause weights, using it to calculate $w_{init}(c)$ for each soft clause c; (2) h_inc, the increment for hard clause weight, used in the Dist-Weighting scheme; (3) s_inc, the increment for soft clause weight, used in Dist-Weighting; (4) h_sp, the smooth probability of hard clause weights, used in Dist-Weighting; (5) s_sp, the smooth probability of soft clause weights, used in Dist-Weighting; (6) δ, whose value plus $w_{init}(c)$ is the maximum weight limit of each soft clause c, used in Dist-Weighting; and (7) t, the number of samples, used in BMS strategy. Actually, most parameters are inherited from SATLike3.0, and our method introduces two additional parameters. The source code of NuWLS is available online.[8]

Following the empirical analysis in Section 'Empirical Study on Initial Soft Clause Weights', we set s_{avg} to avg_best_sw, and h_inc and s_inc are set to the ones used in SATLike3.0. To configure the rest of the parameters for NuWLS, we used an automatic configurator called SMAC (version:2.10.03) (Hutter, Hoos, and Leyton-Brown 2011). We randomly chose about 20% instances from PMS and WPMS benchmarks for training sets. We allowed the total budget of 60000 CPU seconds and the cutoff time of 100 seconds for each solver run. In order to make our comparisons fair, two SLS competitors (*i.e.,* BandMaxSAT and SATLike3.0) were automatically configured using SMAC with the same configuration protocol. We did not configure Loandra and TT-Open-WBO-Inc, since both of them do not expose any configurable parameters.

The parameter settings of NuWLS are as follows: For PMS instances, s_{avg} = 1000, h_inc=1, s_inc=1, h_sp=2e-4, s_sp=6.6e-5, δ=183, and t=96; for WPMS instances, s_{avg}=3000, h_inc=30, s_inc=10, h_sp=6.8e-5, s_sp = 9.9e-7, δ=200, and t=25. Also, we conduct experiments to analyze the effect of different parameter settings of NuWLS, and our results show that NuWLS exhibits robustness across all benchmarks under such parameter settings, which are around the one configured by SMAC within a certain range. For more details, please refer to the technical appendix. The parameter settings of SATLike3.0 and BandMaxSAT are listed in the technical appendix. [8]

Experimental Setup. NuWLS and its all competitors are implemented in C++, and compiled with g++ (version 8.3.1) using the option '-O3'. We conduct all experiments on a workstation running CentOS with the AMD EPYC7702 2.00GHz CPU and 2TB RAM.

In our experiments, each solver performs one run on each instance. For each run, we record the best solution and use the *runsolver* tool (version: 3.4.0) (Roussel 2011) to obtain the time for finding the best solution. The cutoff time is set to 300 CPU seconds for each run, following the rules of the incomplete tracks of MaxSAT Evaluations. For each solver on each benchmark, we report the number of winning instances where the solver finds the best solution among all competing solvers in the same table, denoted by '#win.', and the average time for obtaining the best solution, denoted by 'time' (the unit is CPU second; if not specified the time unit is CPU second). The number of instances in each benchmark is indicated by '#inst.' The results highlighted in **bold** indicate the best performance for the corresponding benchmark.

[8]https://github.com/filyouzicha/NuWLS

Benchmark	#inst.	NuWLS		BandMaxSAT		SATLike3.0	
		#win.	time	#win.	time	#win.	time
PMS_2019	299	**192**	73.62	138	69.98	136	56.02
PMS_2020	262	**164**	70.12	113	77.39	107	58.09
PMS_2021	155	**86**	55.15	84	92.35	49	54.72
WPMS_2019	282	**207**	**102.45**	83	104.15	63	110.14
WPMS_2020	253	**190**	116.75	58	113.67	51	115.35
WPMS_2021	151	**80**	118.96	37	136.63	38	117.39

Table 1: Experimental results of NuWLS and the state-of-the-art SLS solvers on (W)PMS benchmarks.

Benchmark	#inst.	NuWLS		Loandra		TT-OWI	
		#win.	time	#win.	time	#win.	time
PMS_2019	299	**172**	**73.62**	137	105.78	145	78.17
PMS_2020	262	**154**	**70.12**	107	119.01	110	81.34
PMS_2021	155	**73**	**55.15**	59	125.73	65	81.60
WPMS_2019	282	**132**	102.45	110	165.98	116	62.18
WPMS_2020	253	**134**	116.75	81	179.03	98	71.68
WPMS_2021	151	**64**	118.96	60	176.67	52	93.99

Table 2: Experimental results of NuWLS and the state-of-the-art SAT-based solvers (*i.e.,* Loandra and TT-Open-WBO-Inc (TT-OWI for short)) on (W)PMS benchmarks.

Comparisons against SLS Solvers

The results of NuWLS and its SLS competitors on all benchmarks are reported in Table 1, showing that NuWLS performs best on all benchmarks. On PMS_2019 benchmark, the number of winning instances of NuWLS is 1.39 times that of the second best solver BandMaxSAT. On PMS_2020 and PMS_2021 benchmarks, this figure representing performance improvement is 1.45 and 1.02, respectively.

NuWLS also performs best in WPMS solving. On WPMS_2019 benchmark, the number of winning instances of NuWLS is 2.49 times that of the sub-optimal solver. On WPMS_2020 and WPMS_2021 benchmarks, this figure representing performance improvement is 3.28 and 2.11, respectively.

Comparisons against SAT-based Solvers

Table 2 reports the results of NuWLS and its SAT-based competitors on all benchmarks. On all the 3 PMS benchmarks, NuWLS achieves the largest number of winning instances with the shortest average run time.

For solving WPMS, in terms of the number of winning instances, NuWLS outperforms its SAT-based competitors on all the 3 benchmarks. The results indicate that NuWLS exhibits competitive performance in comparison to state-of-the-art SAT-based solvers.

Benchmark	#inst.	VBS_{nuwls}		VBS_{bandms}		$VBS_{satlike3.0}$	
		#win.	time	#win.	time	#win.	time
PMS_2019	299	**268**	**60.49**	249	67.42	236	63.83
PMS_2020	262	**228**	71.26	207	79.78	193	70.35
PMS_2021	155	**139**	**71.84**	125	91.56	119	79.30
WPMS_2019	282	**259**	97.02	214	94.70	205	96.86
WPMS_2020	253	**230**	113.29	163	110.64	165	100.51
WPMS_2021	151	**137**	145.88	105	136.06	108	126.73

Table 3: Experimental results of VBS_{nuwls}, $VBS_{bandmaxsat}$ (VBS_{bandms} for short), and $VBS_{satlike3.0}$ on all the (W)PMS benchmarks.

Complementarity between SLS Solvers and SAT-based Solvers

In this subsection, we conduct experiments to investigate the complementarity between SLS solvers and SAT-based solvers in solving (W)PMS.

To investigate the complementarity between SLS solvers and SAT-based solvers, we construct three perfect portfolio selectors: Given a set of base solvers Θ, for each instance, the solution of the perfect portfolio selector constructed on Θ is the best one of the solutions reported by all solvers in Θ. These three perfect portfolio selectors are built by the set of NuWLS, Loandra, and TT-Open-WBO-Inc, the set of BandMaxSAT, Loandra, and TT-Open-WBO-Inc, and the set of SATLike3.0, Loandra, and TT-Open-WBO-Inc, dubbed by VBS_{nuwls}, $VBS_{bandmaxsat}$ and $VBS_{satlike3.0}$ respectively. Then we conduct experiments to evaluate the performance of these three perfect portfolio selectors on all benchmarks, and the related results are presented in Table 3.

As can be clearly observed in Table 3, regarding the metric of the number of winning instances, VBS_{nuwls} stands out as the best, and achieves much better performance than the other two selectors. The results demonstrate that, compared to existing SLS solvers, NuWLS is able to considerably enhance the complementarity between SLS solvers and SAT-based solvers, which indicates that a portfolio selector, which combines NuWLS and SAT-based solvers, would advance the state of the art in (W)PMS solving.

Improving Hybrid Solver Through NuWLS

As mentioned in the section of related work, solvers combining SLS solvers and SAT-based solvers have achieved the best performance in (W)PMS solving.

Since our experimental results in Table 1 demonstrate that NuWLS performs much better than state-of-the-art SLS solvers for solving (W)PMS on all benchmarks, and Table 3 shows that NuWLS enhances the complementary between SLS solvers and SAT-based solvers, it is interesting to investigate whether NuWLS could improve the performance of the state-of-the-art hybrid solver.

We first modify SATLike-c by replacing SATLike3.0 with NuWLS, resulting in a new hybrid solver dubbed NuWLS-c. On all the 6 benchmarks, we compare NuWLS-c with two state-of-the-art hybrid solvers: SATLike-c, which combines an SLS solver (*i.e.,* SATLike3.0) with an SAT-based solver (*i.e.,* TT-Open-WBO-Inc); DT-HyWalk, which combines several SLS solvers (mainly BandMaxSAT and FPS (Zheng, Zhou, and He 2021)) and an SAT-based solver (*i.e.,*TT-Open-WBO-Inc).

The experimental results are reported in Table 4. Given a solver S and an instance I, we use $Score$ to denote the ratio between the best solution found by solver S on instance I and the best known solution of instance I. Then, given a solver S and a benchmark B, notation '$Score(avg)$' denotes the average value of $Score$ achieved by solver S on all instances in benchmark B. Calculating $Score(avg)$ of each solver on each benchmark is the metric to measure the performance of solvers in incomplete tracks of recent MaxSAT Evaluations. Note that the definition of $Score$ and $Score(avg)$ (metric) in this paragraph has no relationship to the definition of the $score$ of each variable in Section 'Preliminaries'.

From Table 4, regarding $Score(avg)$, NuWLS-c outperforms all the competitors on all 6 benchmarks, indicating that NuWLS can considerably advance the state-of-the-art in hybrid (W)PMS solving. Regarding the metric of the number of winning instances, NuWLS-c outperforms all other solvers on 4 out of 6 benchmarks. On the other 2 benchmarks, NuWLS-c ranks second and a bit worse than the best. In addition, the average run time of NuWLS-c is shorter than that of its competitors on 5 of 6 benchmarks.

Competition Results of NuWLS-c in MSE 2022. To show the superiority of NuWLS-c, we submit NuWLS-c to the incomplete track of MSE 2022 (the latest edition of MaxSAT Evaluations). Encouragingly, according to the official results of MSE 2022[1], NuWLS-c won all four categories in the incomplete track of MSE 2022, confirming the effectiveness of NuWLS-c.

Effects of Clause Weights Initialization Method and Dist-Weighting Scheme

To study the effects of initial clause weights and Dist-Weighting scheme, we develop two alternative versions of NuWLS, *i.e.,* NuWLS-alt1 and NuWLS-alt2. In the NuWLS-alt1 solver, for each soft clause c, $w_{init}(c)$ is set to $w_{ori}(c)$, and the maximum weight limit of c is set to $w_{ori}(c) + \delta$. For NuWLS-alt2, with probability h_sp, the solver decreases the weights of satisfied hard and soft clauses; otherwise it increases the weights of falsified hard and soft clauses. Also, the parameters of these two solvers were automatically configured by SMAC through the same protocol presented in Section 'Experimental Preliminaries'. For details, please refer to the technical appendix.[8]

Table 5 presents the results of NuWLS, NuWLS-alt1, and NuWLS-alt2 on all benchmarks. From Table 5, NuWLS outperforms its two alternative versions. The number of winning instances of NuWLS-atl1 is much smaller than that of NuWLS, indicating that the clause weights initialization method has significant effects, especially on WPMS benchmarks. Also, the number of winning instances of NuWLS-atl2 is much smaller than that of NuWLS, which confirms the effectiveness of the Dist-Weighting scheme.

Benchmark	#inst.	NuWLS-c			DT-HyWalk			SATLike-c		
		$Score(avg)$	#win.	time	$Score(avg)$	#win.	time	$Score(avg)$	#win.	time
PMS_2019	299	**0.911**	214	**54.58**	0.902	**232**	70.25	0.881	169	66.11
PMS_2020	262	**0.897**	176	**59.15**	0.883	**190**	69.04	0.866	130	62.64
PMS_2021	155	**0.899**	**115**	67.21	0.887	111	68.45	0.881	90	65.87
WPMS_2019	282	**0.915**	**190**	**55.94**	0.914	173	66.00	0.905	145	69.51
WPMS_2020	253	**0.909**	**170**	**59.54**	0.877	131	70.79	0.886	114	76.18
WPMS_2021	151	**0.863**	**94**	**78.66**	0.797	70	89.91	0.807	58	100.87

Table 4: Experimental results of NuWLS-c, DT-HyWalk, and SATLike-c on (W)PMS benchmarks.

Benchmark	#inst.	NuWLS		NuWLS-alt1		NuWLS-alt2	
		#win.	time	#win.	time	#win.	time
PMS_2019	299	**178**	73.62	165	67.43	139	59.50
PMS_2020	262	**149**	70.12	141	73.77	122	54.91
PMS_2021	155	**89**	55.15	87	73.22	64	53.26
WPMS_2019	282	**180**	102.45	87	97.97	123	97.93
WPMS_2020	253	**161**	116.75	72	117.84	96	107.22
WPMS_2021	151	**69**	118.96	55	117.86	42	126.81

Table 5: Experimental results of NuWLS, NuWLS-alt1, and NuWLS-alt2 on (W)PMS benchmarks.

Conclusions and Future Work

This paper focused on improving the performance of SLS algorithms for solving (W)PMS via new weighting techniques. We first explored the effect of the initial soft clause weights on the SLS algorithm, which has not been explored in the previous work, and proposed a method for initializing soft clause weights. Second, we proposed a new weighting scheme that updates the weights of hard and soft clauses under different circumstances. We then developed an SLS algorithm called NuWLS. Experimental results demonstrate that NuWLS significantly outperforms previous SLS algorithms on standard (W)PMS benchmarks, and further enhances the performance of the hybrid solver for solving (W)-PMS.

Future work will focus on balancing hard and soft clause weights considering exploration and exploitation. A part of the future work will explore efficient techniques combining local search with complete methods.

Acknowledgments

This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No. XDA0320000 and XDA0320300, the National Natural Science Foundation of China under Grant 62122078, and Grant 62202025.

References

Allouche, D.; Traoré, S.; André, I.; de Givry, S.; Katsirelos, G.; Barbe, S.; and Schiex, T. 2012. Computational Protein Design as a Cost Function Network Optimization Problem. In *Proceedings of CP 2012*, 840–849.

Ansótegui, C.; Bonet, M. L.; and Levy, J. 2009. Solving (weighted) partial MaxSAT through satisfiability testing. In *International conference on theory and applications of satisfiability testing*, 427–440. Springer.

Ansótegui, C.; Bonet, M. L.; and Levy, J. 2010. A New Algorithm for Weighted Partial MaxSAT. In Fox, M.; and Poole, D., eds., *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press.

Ansótegui, C.; Bonet, M. L.; and Levy, J. 2013. SAT-based MaxSAT algorithms. *Artificial Intelligence*, 196: 77–105.

Ansótegui, C.; and Gabàs, J. 2017. WPM3: An (in)complete algorithm for weighted partial MaxSAT. *Artificial Intelligence*, 250: 37–57.

Berg, J.; Demirovic, E.; and Stuckey, P. J. 2019. Core-Boosted Linear Search for Incomplete MaxSAT. In *Proceedings of CPAIOR 2019*, 39–56.

Cai, S. 2015. Balance between Complexity and Quality: Local Search for Minimum Vertex Cover in Massive Graphs. In *Proceedings of IJCAI 2015*, 747–753.

Cai, S.; and Lei, Z. 2020. Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability. *Artificial Intelligence*, 287: 103354.

Cai, S.; Luo, C.; Lin, J.; and Su, K. 2016. New local search methods for partial MaxSAT. *Artificial Intelligence*, 240: 1–18.

Cai, S.; Luo, C.; Thornton, J.; and Su, K. 2014. Tailoring Local Search for Partial MaxSAT. In *Proceedings of AAAI 2014*, 2623–2629.

Cha, B.; Iwama, K.; Kambayashi, Y.; and Miyazaki, S. 1997. Local Search Algorithms for Partial MAXSAT. In *Proceedings of AAAI 1997*, 263–268.

Davies, J.; and Bacchus, F. 2011. Solving MAXSAT by Solving a Sequence of Simpler SAT Instances. In *Proceedings of CP 2011*, 225–239.

Fu, Z.; and Malik, S. 2006. On Solving the Partial MAX-SAT Problem. In *Proceedings of SAT 2006*, 252–265.

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential Model-Based Optimization for General Algorithm Configuration. In *Proceedings of LION 2011*, 507–523.

Jiang, Y.; Kautz, H.; and Selman, B. 1995. Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT. In *1st International Joint Workshop on*

Artificial Intelligence and Operations Research, volume 20. Citeseer.

Joshi, S.; Kumar, P.; Rao, S.; and Martins, R. 2019. Open-WBO-Inc: Approximation Strategies for Incomplete Weighted MaxSAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1): 73–97.

Koshimura, M.; Zhang, T.; Fujita, H.; and Hasegawa, R. 2012. QMaxSAT: A partial Max-SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 8(1-2): 95–100.

Lei, Z.; and Cai, S. 2018. Solving (Weighted) Partial MaxSAT by Dynamic Local Search for SAT. In *Proceedings of IJCAI 2018*, 1346–1352.

Lei, Z.; and Cai, S. 2020. NuDist: An Efficient Local Search Algorithm for (Weighted) Partial MaxSAT. *The Computer Journal*, 63(9): 1321–1337.

Lei, Z.; Cai, S.; Geng, F.; Wang, D.; Peng, Y.; Wan, D.; Deng, Y.; and Lu, P. 2021. SATLike-c: Solver Description. In *Proceedings of MaxSAT Evaluation 2021: Solver and Benchmark Descriptions*, 19–20.

Li, C.; Xu, Z.; Coll, J.; Manyà, F.; Habet, D.; and He, K. 2021. Combining Clause Learning and Branch and Bound for MaxSAT. In Michel, L. D., ed., *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, volume 210 of *LIPIcs*, 38:1–38:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Luo, C.; Cai, S.; Su, K.; and Huang, W. 2017. CCEHC: An efficient local search algorithm for weighted partial maximum satisfiability. *Artificial Intelligence*, 243: 26–44.

Luo, C.; Cai, S.; Wu, W.; Jie, Z.; and Su, K. 2015. CCLS: an efficient local search algorithm for weighted maximum satisfiability. *IEEE Transactions on Computers*, 64(7): 1830–1843.

Martins, R.; Joshi, S.; Manquinho, V.; and Lynce, I. 2014. Incremental cardinality constraints for MaxSAT. In *Proceedings of CP 2014*, 531–548.

Nadel, A. 2019. Anytime weighted MaxSAT with improved polarity selection and bit-vector optimization. In *Proceedings of FMCAD 2019*, 193–202.

Nadel, A. 2020. TT-Open-WBO-Inc-20: an Anytime MaxSAT Solver Entering MSE'20. In *Proceedings of MaxSAT Evaluation 2020: Solver and Benchmark Descriptions*, 21–22.

Naji-Azimi, Z.; Toth, P.; and Galli, L. 2010. An electromagnetism metaheuristic for the unicost set covering problem. *European Journal of Operational Research*, 205(2): 290–300.

Narodytska, N.; and Bacchus, F. 2014. Maximum Satisfiability Using Core-Guided MaxSAT Resolution. In *Proceedings of AAAI 2014*, 2717–2723.

Roussel, O. 2011. Controlling a Solver Execution with the runsolver Tool. *Journal on Satisfiability, Boolean Modeling and Computation*, 7(4): 139–144.

Thornton, J.; Bain, S.; Sattar, A.; and Pham, D. N. 2002. A Two Level Local Search for MAX-SAT Problems with Hard and Soft Constraints. In *Proceedings of AI 2002*, 603–614.

Thornton, J.; and Sattar, A. 1998. Dynamic Constraint Weighting for Over-Constrained Problems. In *Proceedings of PRICAI 1998*, 377–388.

Zheng, J.; He, K.; Zhou, J.; Jin, Y.; Li, C.-M.; and Manya, F. 2022. BandMaxSAT: A Local Search MaxSAT Solver with Multi-armed Bandit. In *The Thirty-First International Joint Conference on Artificial Intelligence*.

Zheng, J.; Zhou, J.; and He, K. 2021. Farsighted Probabilistic Sampling based Local Search for (Weighted) Partial MaxSAT. *arXiv preprint arXiv:2108.09988*.