BandMaxSAT: A Local Search MaxSAT Solver with Multi-armed Bandit

Jiongzhi Zheng[1,2] , **Kun He**[*1] , **Jianrong Zhou**[1] , **Yan Jin**[1] , **Chu-Min Li**[3] , **Felip Manya**[4]

[1]School of Computer Science and Technology, Huazhong University of Science and Technology
[2]Institute of Artificial Intelligence, Huazhong University of Science and Technology, China
[3]MIS, University of Picardie Jules Verne, France
[4]Artificial Intelligence Research Institute (IIIA), CSIC, Bellaterra, Spain
jzzheng@hust.edu.cn, brooklet60@hust.edu.cn

Abstract

We address Partial MaxSAT (PMS) and Weighted PMS (WPMS), two practical generalizations of the MaxSAT problem, and propose a local search algorithm for these problems, called BandMaxSAT, that applies a multi-armed bandit model to guide the search direction. The bandit in our method is associated with all the soft clauses in the input (W)PMS instance. Each arm corresponds to a soft clause. The bandit model can help BandMaxSAT to select a good direction to escape from local optima by selecting a soft clause to be satisfied in the current step, that is, selecting an arm to be pulled. We further propose an initialization method for (W)PMS that prioritizes both unit and binary clauses when producing the initial solutions. Extensive experiments demonstrate that BandMaxSAT significantly outperforms the state-of-the-art (W)PMS local search algorithm SATLike3.0. Specifically, the number of instances in which BandMaxSAT obtains better results is about twice that obtained by SATLike3.0. Moreover, we combine BandMaxSAT with the complete solver TT-Open-WBO-Inc. The resulting solver BandMaxSAT-c also outperforms some of the best state-of-the-art complete (W)PMS solvers, including SATLike-c, Loandra and TT-Open-WBO-Inc.

1 Introduction

As an optimization extension of the famous Boolean Satisfiability (SAT) decision problem, the Maximum Satisfiability (MaxSAT) problem aims at finding a complete assignment of the Boolean variables to satisfy as many clauses as possible in a given propositional formula in Conjunctive Normal Form (CNF) [Li and Manyà, 2021]. Partial MaxSAT (PMS) is a variant of MaxSAT where the clauses are divided into hard and soft. PMS aims at maximizing the number of satisfied soft clauses with the constraint that all the hard clauses must be satisfied. Associating a positive weight to each soft clause in PMS results in Weighted PMS (WPMS), whose goal is to maximize the total weight of satisfied soft clauses with the same constraint of PMS that all the hard clauses must be satisfied. Both PMS and WPMS, denoted as (W)PMS, have many practical applications such as planning [Bonet *et al.*, 2019], combinatorial testing [Ansótegui *et al.*, 2022], group testing [Ciampiconi *et al.*, 2020], and timetabling [Demirovic and Musliu, 2017].

In this paper, we focus on the local search approach, which is a well-studied category of incomplete (W)PMS algorithms and exhibits promising performance on random and crafted (W)PMS instances. Recent well-performing (W)PMS local search algorithms, such as Dist [Cai *et al.*, 2014], CCEHC [Luo *et al.*, 2017], SATLike [Lei and Cai, 2018] and SATLike3.0 [Cai and Lei, 2020], all start from an initial complete assignment and then flip the Boolean value of a selected variable per step to find better solutions. These local search algorithms follow similar procedures to escape from local optima. Note that a local optimum indicates that flipping any single variable cannot improve the current solution.

When falling into an infeasible local optimum (i.e., there are falsified hard clauses), these algorithms first randomly select a falsified hard clause and then satisfy it by flipping one of its variables. The random strategy for selecting the falsified hard clause is reasonable, since all the hard clauses should be satisfied. However, when falling into a feasible local optimum (i.e., there are no falsified hard clauses), these algorithms still use the random strategy to determine the soft clause to be satisfied in the current step, which may not be a good strategy for the following reasons: 1) different from hard clauses, not all the soft clauses should be satisfied. 2) the high degree of randomness may lead to a small probability for these algorithms to find a good search direction (satisfying a falsified soft clause corresponds to a search direction).

To handle the above issues, we propose a multi-armed bandit (MAB) local search algorithm, called BandMaxSAT, for (W)PMS. MAB is a basic model in the field of reinforcement learning [Slivkins, 2019; Lattimore and Szepesvári, 2020]. In an MAB reinforcement learning model, the agent needs to select to pull an arm (i.e., perform an action) at each decision step (i.e., state), which leads to some rewards. The agent uses the rewards to evaluate the benefit of pulling each arm and uses the evaluation values to decide the arm to be pulled in each step. In summary, the MAB can be used to help a program learn to select an appropriate item from multiple candidates. Therefore, we propose to apply an MAB to help the

*Corresponding author.

(W)PMS local search algorithm learn to select an appropriate soft clause (i.e., a high-quality search direction) to be satisfied, whenever the search falls into a feasible local optimum. Specifically, each arm in the bandit of BandMaxSAT corresponds to a soft clause in the input (W)PMS instance. Pulling an arm implies selecting the corresponding clause to be satisfied in the current step.

There are related studies that apply MAB to MaxSAT. For example, Goffinet and Ramanujan [2016] proposed an algorithm for MaxSAT, based on Monte-Carlo tree search, where a two-armed bandit is associated with each variable (node in the search tree) to decide the branching direction, i.e., which Boolean value assign to the variable. Their approach needs a local search solver to evaluate the quality of the branching nodes, thus the performance relies on the local search solver. Lassouaoui et al. [2019] proposed to use an MAB model to select the low-level heuristics in a hyper-heuristic framework for MaxSAT. Pulling an arm in their model implies selecting a corresponding low-level heuristic to optimize the current solution. They didn't compare with the state-of-the-art (in)complete MaxSAT algorithms, but only compared with other hyper-heuristic methods. Our work proposes a novel MAB model for (W)PMS that significantly improves (W)PMS state-of-the-art local search methods. To our knowledge, this is the first time that an MAB model is associated with the clauses in a (W)PMS local search solver.

Moreover, inspired by the studies for SAT and MaxSAT that prioritize both unit and binary clauses (i.e., clauses with exactly one and two literals, respectively) over other clauses [Chao and Franco, 1990; Chvátal and Reed, 1992; Li *et al.*, 2006], we propose a novel decimation approach that prefers to satisfy both unit and binary clauses, denoted as hybrid decimation (HyDeci), for generating the initial assignment in BandMaxSAT. The decimation method is a category of incomplete approaches that proceeds by assigning the Boolean value of some (usually one) variables sequentially and simplifies the formula accordingly [Cai *et al.*, 2017]. Decimation approaches that focus on unit clauses have been used in MaxSAT [Cai *et al.*, 2017; Cai and Lei, 2020]. However, it is the first time, to our knowledge, that a decimation method concentrating on both unit and binary clauses is used in MaxSAT. The experimental results demonstrate that considering both unit and binary clauses is better than only considering unit clauses.

To evaluate the performance of the proposed BandMaxSAT algorithm, we compare BandMaxSAT with the state-of-the-art (W)PMS local search algorithm SATLike3.0 [Cai and Lei, 2020]. The experiments show that BandMaxSAT significantly outperforms SATLike3.0 on both PMS and WPMS. Moreover, as one of the state-of-the-art (W)PMS solvers, SATLike-c [Lei *et al.*, 2021] combines SATLike3.0 with an effective complete solver, TT-Open-WBO-Inc [Nadel, 2019], and won three categories among the total four (PMS and WPMS categories, each associated with two time limits) of the incomplete track in the latest MaxSAT Evaluation (MSE2021). By combining BandMaxSAT with TT-Open-WBO-Inc, the resulting solver BandMaxSAT-c also outperforms some of the best state-of-the-art (W)PMS complete solvers, including SATLike-c, Loandra [Berg *et al.*, 2019], and TT-Open-WBO-Inc.

The main contributions of this work are as follows:

- We propose a multi-armed bandit (MAB) model that fits well with the MaxSAT local search algorithms, and an effective local search solver for (W)PMS, called BandMaxSAT, that applies the proposed bandit model to guide the search direction.
- We demonstrate that there is a great potential to use MAB in MaxSAT solving. Our proposed MAB model is general and could be applied to improve other MaxSAT local search algorithms.
- We propose a novel decimation method for (W)PMS, denoted as HyDeci, that prefers to satisfy both unit and binary clauses. HyDeci provides high-quality initial assignments for BandMaxSAT, and could be applied to improve other MaxSAT local search algorithms.
- Extensive experiments show that BandMaxSAT significantly outperforms the state-of-the-art (W)PMS local search algorithm SATLike3.0. Moreover, by combining BandMaxSAT with the complete solver TT-Open-WBO-Inc, the resulting solver BandMaxSAT-c also outperforms the state-of-the-art (W)PMS complete solvers.

2 Preliminaries

Given a set of Boolean variables $\{x_1, ..., x_n\}$, a literal is either a variable itself x_i or its negation $\neg x_i$; a clause is a disjunction of literals, i.e., $c_j = l_{j1} \vee ... \vee l_{jn_j}$, where n_j is the number of literals in clause c_j. A Conjunctive Normal Form (CNF) formula $\mathcal{F}$ is a conjunction of clauses, i.e., $\mathcal{F} = c_1 \wedge ... \wedge c_m$. A complete assignment A represents a mapping that maps each variable to a value of 1 (true) or 0 (false). A literal x_i (resp. $\neg x_i$) is true if the current assignment maps x_i to 1 (resp. 0). A clause is satisfied by the current assignment if there is at least one true literal in the clause.

Given a CNF formula $\mathcal{F}$,MaxSAT aims at finding an assignment that satisfies as many clauses in $\mathcal{F}$ as possible. Given a CNF formula $\mathcal{F}$ whose clauses are divided into hard and soft, PMS is a variant of MaxSAT that aims at finding an assignment that satisfies all the hard clauses and maximize the number of satisfied soft clauses in $\mathcal{F}$, and WPMS is a generalization of PMS where each soft clause is associated with a positive weight. The goal of WPMS is to find an assignment that satisfies all the hard clauses and maximizes the total weight of satisfied clauses in $\mathcal{F}$. In the local search algorithms for (Max)SAT, the flipping operator for a variable is an operator that changes its Boolean value.

Given a (W)PMS instance $\mathcal{F}$, a complete assignment A is feasible if it satisfies all the hard clauses in $\mathcal{F}$. The cost of A, denoted as $cost(A)$, is set to $+\infty$ for convenience if A is infeasible. Otherwise, $cost(A)$ is equal to the number of falsified soft clauses for PMS, and equal to the total weight of falsified soft clauses for WPMS.

In addition, the effective clause weighting technique is widely used in recent well-performing (W)PMS local search algorithms [Cai *et al.*, 2014; Luo *et al.*, 2017; Cai and Lei, 2020]. Algorithms with this technique associate dynamic weights (independent of the original soft clause weights in WPMS instances) to clauses and use the dynamic weights to guide the search direction. BandMaxSAT also applies the clause weighting technique, and maintains dynamic weights to both hard clauses and soft clauses with the clause weighting strategy used in SATLike3.0 [Cai and Lei, 2020].

Given a (W)PMS instance $\mathcal{F}$, the current assignment A, and the dynamic clause weights, the commonly used scoring function for a variable x, denoted as $score(x)$, is defined as the increment or reduction of the total dynamic weight of satisfied clauses caused by flipping x in A. Moreover, a local optimum for (W)PMS indicates that there are no variables with positive *score*. A local optimum is feasible if there are no falsified hard clauses, otherwise it is infeasible.

3 Methodology

The proposed local search algorithm BandMaxSAT consists of the proposed hybrid decimation (HyDeci) initialization process and the search process. During the local search process, we use a multi-armed bandit that is associated with the soft clauses to help BandMaxSAT learn to select good directions to escape from feasible local optima. This section first introduces the HyDeci method and the bandit model used in BandMaxSAT, and then the main process of BandMaxSAT.

3.1 Hybrid Decimation

HyDeci is an effective decimation method that prefers to satisfy both unit and binary clauses. Since the clauses with shorter lengths are easier to be falsified, preferring to satisfy shorter clauses can reduce the number of falsified clauses, which results in high-quality initial assignments. The procedure of HyDeci is shown in Algorithm 1. We use SIMPLIFY to refer to the process of simplifying the formula after assigning a value to a variable.

Algorithm 1: HyDeci($\mathcal{F}$)

Input: A (W)PMS instance $\mathcal{F}$
Output: A complete assignment A of variables in $\mathcal{F}$

while $\exists$ *unassigned variables* **do**
  if $\exists$ *hard unit clauses* **then**
    $c :=$ a random hard unit clause;
    satisfy c and SIMPLIFY;
  else if $\exists$ *soft unit clauses* **then**
    $c :=$ a random soft unit clause;
    satisfy c and SIMPLIFY;
  else if $\exists$ *hard binary clauses* **then**
    $c :=$ a random hard binary clause;
    $l :=$ a greedily selected unassigned literal in c;
    satisfy l and SIMPLIFY;
  else if $\exists$ *soft binary clauses* **then**
    $c :=$ a random soft binary clause;
    $l :=$ a greedily selected unassigned literal in c;
    satisfy l and SIMPLIFY;
  else
    $v :=$ a random unassigned variable;
    assign v a random value and SIMPLIFY;
return the resulting complete assignment A;

HyDeci generates the initial complete assignment iteratively. In each iteration, HyDeci assigns the value of exactly one variable. When there are unit clauses, HyDeci samples a random unit clause (hard clauses take precedence) and then satisfies it. When there is no unit clause but there are binary clauses, HyDeci first samples a random binary clause c (hard clauses take precedence), and then selects one of the two unassigned literals in c and satisfies it according to a greedy strategy, that is, preferring to satisfy the literal whose satisfaction leads to more satisfied soft clauses (or to a larger total weight of satisfied soft clauses). When there are no unit and binary clauses, HyDeci randomly selects an unassigned variable and randomly assigns a Boolean value to it.

The main improvement of the proposed HyDeci algorithm over the previous decimation approaches [Cai *et al.*, 2017; Cai and Lei, 2020] is that HyDeci not only concentrates on unit clauses but also on binary clauses. The experimental results demonstrate that considering both unit and binary clauses is better than only considering unit clauses to generate high-quality initial assignments.

3.2 Multi-armed Bandit Model for (W)PMS

We propose a multi-armed bandit model for (W)PMS to help BandMaxSAT learn to select the appropriate soft clause to be satisfied when falling into a feasible local optimum. Each arm of the bandit model corresponds to a soft clause. Pulling an arm implies selecting the corresponding soft clause to be satisfied in the current step. The bandit model maintains an estimated value $V(i)$ and a selected time $t(i)$ for each arm (i.e., soft clause) i. We initialize $V(i) = 1$ and $t(i) = 0$ for each arm i. The larger the estimated value of an arm, the more benefits of pulling the arm, i.e., satisfying the soft clause corresponding to the arm may yield better solutions.

The rest of this subsection first introduces the method of selecting an arm to be pulled and then the method of updating the estimated values.

Arm Selection Strategy

BandMaxSAT uses the Upper Confidence Bound method [Hu *et al.*, 2019] to trade-off between exploration and exploitation and selects the arm to be pulled. Specifically, the upper confidence bound U_i on the estimated value V_i of arm i is calculated with the following equation:

$$U_i = V_i + \lambda \cdot \sqrt{\frac{ln(N)}{t(i)+1}}, \tag{1}$$

where N indicates the number of times fallen into a feasible local optimum and λ is the exploration bias parameter.

The procedure of selecting the arm is shown in Algorithm 2. Since our bandit contains a large number of arms (equal to the number of soft clauses), selecting the best among all the arms is inefficient. Therefore, BandMaxSAT first applies (line 3) the sampling strategy to randomly sample $ArmNum$ (20 by default) candidate arms and then selects the arm with the highest upper confidence bound among the candidates (lines 4-5). Similar sampling strategies have been used in

Algorithm 2: PickArm($ArmNum, N, \lambda$)

Input: Number of sampled arms $ArmNum$, number of times to fall into a local optimum N, exploration bias parameter λ
Output: The arm selected to be pulled c

initialize $U^* := -\infty$;
for $i := 1$ *to* $ArmNum$ **do**
  $j :=$ a random falsified soft clause;
  calculate U_j according to Eq. 1;
  if $U_j > U^*$ **then** $U^* := U_j$, $c := j$;
return c;

multi-armed bandit problems [Ou *et al.*, 2019] and some combinatorial optimization problems [Cai, 2015]. Note that the bandit aims at selecting a soft clause to be satisfied in the current step. Thus, the arms corresponding to the soft clauses that are satisfied by the current assignment will not be considered as candidates. The experimental results show that the sampling strategy in our bandit model can significantly improve the algorithm's performance.

Estimated Value Updating Strategy

Since BandMaxSAT pulls an arm of the bandit when it falls into a feasible local optimum, we apply the *cost* values of the feasible local optimal solutions before and after pulling an arm to calculate the reward of the action (i.e., pulling the arm). Suppose A' and A are the last and current feasible local optimal solutions respectively, and c is the last pulled arm, a simple reward for pulling c can be set to $r = cost(A') - cost(A)$. However, reducing the *cost* value from 20 to 10 is much harder and more meaningful than reducing it from 1000 to 990. Thus, the rewards of these two cases should not be the same. To address this issue, we define the reward as follows:

$$r(A, A', A^*) = \frac{cost(A') - cost(A)}{cost(A') - cost(A^*) + 1}, \tag{2}$$

where A^* is the best solution found so far. Suppose in Eq. 2 $cost(A') - cost(A)$ is constant, then the closer $cost(A')$ and $cost(A^*)$, the more rewards the action of pulling the last arm can yield, which is reasonable and intuitive.

Moreover, since the arms (i.e., soft clauses) are connected by the variables, we assume that the arms in our bandit model are not independent of each other. We also believe that the improvement (or deterioration) of A over A' may not only be due to the last action, but also due to earlier actions. Hence, we apply the delayed reward method [Arya and Yang, 2020] to update the estimated value of the last d (20 by default) pulled arms once a reward is obtained. Specifically, suppose that A' and A are the last and current feasible local optimal solutions respectively, A^* is the best solution found so far, and $\{a_1, ..., a_d\}$ is the set of the latest d pulled arms (a_d is the most recent one). Then, the estimated values of the d arms are updated as follows:

$$V_{a_i} = V_{a_i} + \gamma^{d-i} \cdot r(A, A', A^*), i \in \{1, ..., d\}, \tag{3}$$

where γ is the reward discount factor and $r(A, A', A^*)$ is calculated with Eq. 2.

Algorithm 3: BandMaxSAT

Input: A (W)PMS instance $\mathcal{F}$, cut-off time *cutoff*, BMS parameter k, reward delay steps d, reward discount factor γ, number of sampled arms $ArmNum$, exploration bias parameter λ
Output: A feasible assignment A of $\mathcal{F}$, or *no feasible assignment found*

1 $A :=$ HyDeci($\mathcal{F}$);
2 $A^* := A$, $cost(A') := +\infty$, $N := 0$;
3 **while** *running time < cutoff* **do**
4   **if** *A is feasible & cost(A) < cost(A*)* **then**
5     $A^* := A$;
6   **if** $D := \{x | score(x) > 0\} \neq \emptyset$ **then**
7     $v :=$ a variable in D picked by BMS(k);
8   **else**
9   update_clause_weights();
10   **if** $\exists$ *falsified hard clauses* **then**
11       $c :=$ a random falsified hard clause;
12   **else**
13   update_estimated_value(A, A', A^*, d, γ);
14       $N := N + 1$, $A' := A$;
15       $c :=$ PickArm($ArmNum, N, \lambda$);
16       $t(c) := t(c) + 1$;
17     $v :=$ the variable with the highest *score* in c;
18   $A := A$ with v flipped;
19 **if** A^* *is feasible* **then return** A^*;
20 **else return** *no feasible assignment found*;

3.3 Main Process of BandMaxSAT

Finally, we introduce the main process of BandMaxSAT, which is shown in Algorithm 3. BandMaxSAT first uses HyDeci to generate an initial assignment, and then repeatedly selects a variable and flips it until the cut-off time is reached.

When local optima are not reached, BandMaxSAT selects a variable to be flipped using the Best from Multiple Selections (BMS) strategy [Cai, 2015]. BMS chooses k random variables (with replacement) and returns one with the highest *score* (lines 6-7). When falling into a local optimum, BandMaxSAT first updates the dynamic clause weights (by the update_clause_weight() function in line 9) according to the clause weighting scheme in SATLike3.0 [Cai and Lei, 2020], and then selects the clause to be satisfied in the current step.

If the local optimum is infeasible, BandMaxSAT randomly selects a falsified hard clause as the clause to be satisfied in the current step (lines 10-11). If the local optimum is feasible (lines 12-15), BandMaxSAT first updates the estimated values of the latest d pulled arms according to Eq. 3, and then uses the PickArm() function (Algorithm 2) to select the soft clause to be satisfied in the current step. After determining the clause c to be satisfied, BandMaxSAT flips the variable with the highest *score* in c in the current step (line 16).

4 Experimental Results

We compare BandMaxSAT with the state-of-the-art (W)PMS local search algorithm, SATLike3.0 [Cai and Lei, 2020], as well as some of the best state-of-the-art complete solvers SATLike-c [Lei *et al.*, 2021], Loandra [Berg *et al.*, 2019] and TT-Open-WBO-Inc [Nadel, 2019]. The experimental results demonstrate the excellent performance of our proposed BandMaxSAT algorithm, that significantly improves the best (W)PMS local search solver. The results also show the effectiveness of each component in BandMaxSAT, including the HyDeci initialization method, the sampling strategy and the delayed reward method.

4.1 Experimental Setup

BandMaxSAT is implemented in C++ and compiled by g++. Our experiments were performed on a server using an Intel® Xeon® E5-2650 v3 2.30 GHz 10-core CPU and 256 GB RAM, running Ubuntu 16.04 Linux operation system. We tested the algorithms on all the (W)PMS instances from the incomplete track of last four MaxSAT Evaluations (MSE), i.e., MSE2018-MSE2021. Note that we denote the benchmark that contains all the PMS/WPMS instances from the incomplete track of MSE2021 as PMS_2021/WPMS_2021, and so on. Each instance is calculated once by each algorithm with a time limit of 300 seconds, which is consistent with the settings in MSEs and [Cai and Lei, 2020]. The best results in the tables appear in bold. Moreover, we use BandMS as a short name of BandMaxSAT in the tables.

The parameters in BandMaxSAT include the BMS parameter k, the reward delay steps d, the reward discount factor γ, the number of sampled arms $ArmNum$, and the exploration bias parameter λ. We use all the (W)PMS instances from the incomplete track of MSE2017 as the training set to tune these parameters. The parameter domains of these parameters are as follows: $[10, 50]$ for k, $[1, 50]$ for d, $[0.5, 1]$ for γ, $[10, 50]$ for $ArmNum$, and $[0.1, 10]$ for λ. Finally, the default settings of these parameters are as follows: $k = 15$, $d = 20$, $\gamma = 0.9$, $ArmNum = 20$, $\lambda = 1$. The code of BandMaxSAT is available at https://github.com/JHL-HUST/BandMaxSAT/.

4.2 Comparison of BandMaxSAT and SATLike3.0

We first compare BandMaxSAT with the state-of-the-art (W)PMS local search algorithm, SATLike3.0 [Cai and Lei, 2020], on all the tested instances. The results are shown in Table 1. Column *#inst.* indicates the number of instances in each benchmark. Column *#win.* indicates the number of instances in which the algorithm yields the best solution among all the algorithms in the table. Column *time* represents the average running time to yield the *#win.* instances.

As shown by the results in Table 1, BandMaxSAT (BandMS) significantly outperforms SATLike3.0 for both PMS and WPMS. Specifically, the *#win.* instances of BandMaxSAT is 62-131% greater than those of SATLike3.0 for WPMS, and 43-103% greater than those of SATLike3.0 for PMS, indicating a significant improvement.

4.3 Comparison with Complete Solvers

We then combine our BandMaxSAT local search algorithm with the complete solver TT-Open-WBO-Inc (TT-OWI) as SATLike-c does (which combines SATLike3.0 with TT-OWI), and denote the resulting solver as BandMaxSAT-c (BandMS-c). We further compare BandMaxSAT-c with some of the best state-of-the-art complete (W)PMS solvers that showed excellent performance in recent MSEs, including SATLike-c, Loandra and TT-OWI (all of them are downloaded from MSE2021). We apply the scoring function used in MSE2021 to evaluate the performance of these four solvers, since the scoring function is suitable for evaluating and comparing multiple solvers. The scoring function actually indicates how close the solutions are to the best-known solutions. Specifically, suppose C_{BKS} is the cost of the best-known solution of an instance which is recorded in MSEs, C_i is the cost of the solution found by the i-th solver ($i \in \{1, 2, 3, 4\}$) in our experiments, the score of solver i for this instance is $\frac{min(C_{BKS}, C_1, C_2, C_3, C_4)+1}{C_i+1} \in [0, 1]$ (resp. 0) if the solution found by solver i is feasible (resp. infeasible). Finally, the score of a solver for a benchmark is the average value of the scores for all the instances in the benchmark.

Benchmark	#inst.	BandMS		SATLike3.0	
		#win.	time	#win.	time
WPMS_2018	172	**118**	114.76	51	73.68
WPMS_2019	297	**210**	108.47	103	75.50
WPMS_2020	253	**170**	137.45	88	81.81
WPMS_2021	151	**89**	145.87	55	89.70
PMS_2018	153	**110**	99.15	54	80.39
PMS_2019	299	**204**	67.93	143	53.67
PMS_2020	262	**174**	73.91	119	63.18
PMS_2021	155	**112**	68.66	60	51.77

Table 1: Comparison of BandMS and SATLike3.0.

Benchmark	BandMS-c	SATLike-c	Loandra	TT-OWI
WPMS_2018	**0.9041**	0.8901	0.8820	0.9026
WPMS_2019	0.8974	0.8819	0.8474	**0.9022**
WPMS_2020	**0.8707**	0.8613	0.8043	0.8655
WPMS_2021	**0.7844**	0.7809	0.7833	0.7729
PMS_2018	**0.8537**	0.8440	0.7811	0.8412
PMS_2019	**0.8793**	0.8745	0.7818	0.8713
PMS_2020	**0.8619**	0.8511	0.8216	0.8586
PMS_2021	**0.8592**	0.8538	0.7714	0.8436

Table 2: Comparison of BandMS-c and the state-of-the-art complete (W)PMS solvers, SATLike-c, Loandra, TT-Open-WBO-Inc. The results are expressed by the scoring function used in MSE2021.

The comparison results of these four solvers are shown in Table 2. BandMaxSAT-c yields the highest score on all the benchmarks except WPMS_2019, demonstrating the excellent performance of our proposed method.

4.4 Ablation Study

Finally, we perform ablation studies to analyze the effect of each component in BandMaxSAT. We first compare BandMaxSAT with two variants to evaluate the sampling strategy in the bandit model. The first one is $\text{BandMaxSAT}_{\text{sample-1}}$ (Sample-1 in brief), a variant of BandMaxSAT that sets the parameter $ArmNum$ to 1, which actually replaces the whole bandit model in BandMaxSAT with the simple random strategy used in Dist [Cai *et al.*, 2014], CCEHC [Luo *et al.*, 2017], SATLike [Lei and Cai, 2018] and SATLike3.0 [Cai and Lei, 2020]. The second one is $\text{BandMaxSAT}_{\text{sample-all}}$ (Sample-all in brief), a variant of BandMaxSAT that removes the sam-

Benchmark	#inst.	BandMS		Sample-1		Sample-all	
		#win.	time	#win.	time	#win.	time
WPMS_2018	172	**100**	102.04	62	66.47	65	74.15
WPMS_2019	297	**188**	100.35	117	73.21	118	78.91
WPMS_2020	253	**160**	127.43	96	81.86	100	94.54
WPMS_2021	151	**80**	134.94	53	114.89	46	90.91
PMS_2018	153	**102**	87.75	74	59.45	67	65.78
PMS_2019	299	**191**	65.35	164	53.26	130	59.26
PMS_2020	262	**168**	67.38	139	54.28	111	66.21
PMS_2021	155	**109**	65.61	83	36.24	72	49.41

Table 3: Comparison with variants Sample-1 and Sample-all.

Benchmark	#inst.	BandMS		$\text{BandMS}_{\text{no-delay}}$	
		#win.	time	#win.	time
WPMS_2018	172	**117**	108.96	85	64.60
WPMS_2019	297	**187**	106.78	165	87.30
WPMS_2020	253	**161**	130.91	133	91.50
WPMS_2021	151	**89**	140.16	64	112.62
PMS_2018	153	**113**	91.99	97	57.00
PMS_2019	299	**202**	66.14	191	58.40
PMS_2020	262	**181**	71.78	158	43.55
PMS_2021	155	**108**	60.98	97	48.91

Table 4: Comparison with variant $\text{BandMS}_{\text{no-binary}}$.

Benchmark	#inst.	BandMS		$\text{BandMS}_{\text{no-binary}}$	
		#win.	time	#win.	time
WPMS_2018	172	**105**	110.11	86	94.17
WPMS_2019	297	**190**	104.24	165	93.96
WPMS_2020	253	**161**	129.54	135	103.39
WPMS_2021	151	**89**	146.36	76	119.98
PMS_2018	153	**100**	107.80	98	79.07
PMS_2019	299	**203**	76.28	201	63.91
PMS_2020	262	**178**	80.20	176	63.13
PMS_2021	155	**102**	72.00	92	42.79

Table 5: Comparison with variant $\text{BandMS}_{\text{no-binary}}$.

Benchmark	#inst.	$\text{BandMS}_{\text{fast}}$		$\text{BandMS}_{\text{no-binary-fast}}$	
		#win.	time	#win.	time
WPMS_2018	172	**91**	9.70	89	3.15
WPMS_2019	297	162	10.80	**165**	7.31
WPMS_2020	253	**150**	13.18	136	11.36
WPMS_2021	151	**90**	18.49	69	15.62
PMS_2018	153	**94**	6.15	90	4.64
PMS_2019	299	**188**	5.06	168	4.72
PMS_2020	262	**166**	4.62	147	3.75
PMS_2021	155	**89**	7.17	79	5.13

Table 6: Comparison with $\text{BandMS}_{\text{fast}}$ and $\text{BandMS}_{\text{no-binary-fast}}$.

pling strategy in the bandit model, i.e., selecting the arm to be pulled by traversing all the available arms. The results are shown in Table 3.

The results in Table 3 show that our bandit model significantly outperforms the random strategy that is widely used in recent (W)PMS local search algorithms, and can greatly improve the performance. Moreover, the sampling strategy used in our bandit model is effective and necessary.

We then compare BandMaxSAT with its another variant $\text{BandMaxSAT}_{\text{no-delay}}$, that sets the parameter d to 1, to evaluate the delayed reward method. The results are shown in Table 4. The results indicate that the delayed reward method fits well with the problems, and the method can help BandMaxSAT evaluate the quality of the arms better.

We further do two groups of experiments to evaluate the proposed HyDeci initialization method. The first group compares BandMaxSAT with its variant $\text{BandMaxSAT}_{\text{no-binary}}$, that does not prioritize binary clauses in HyDeci (i.e., remove lines 8-15 in Algorithm 1). The second group compares two variants of BandMaxSAT. They are, $\text{BandMaxSAT}_{\text{fast}}$, a variant of BandMaxSAT that outputs the first feasible solution it found (within a time limit of 300 seconds), and $\text{BandMaxSAT}_{\text{no-binary-fast}}$, a variant of $\text{BandMaxSAT}_{\text{no-binary}}$ that outputs the first feasible solution it found. We actually use $\text{BandMaxSAT}_{\text{fast}}$ and $\text{BandMaxSAT}_{\text{no-binary-fast}}$ to roughly evaluate the quality of the initial assignments. The results of these two groups are shown in Tables 5 and 6, respectively.

From the results in Tables 5 and 6 we can see that:

(1) $\text{BandMaxSAT}_{\text{fast}}$ outperforms $\text{BandMaxSAT}_{\text{no-binary-fast}}$ on all the benchmarks except WPMS_2019, indicating that our method that prioritizes both unit and binary clauses can yield better initial assignments than the method that only prioritizes unit clauses.

(2) BandMaxSAT outperforms $\text{BandMaxSAT}_{\text{no-binary}}$ for WPMS, indicating that our HyDeci method is effective and can improve the BandMaxSAT for WPMS. For PMS, These two algorithms have close performance, indicating that the local search process in BandMaxSAT is robust for PMS, as $\text{BandMaxSAT}_{\text{no-binary}}$ can obtain PMS solutions with similar quality to BandMaxSAT, with worse initial assignments.

5 Conclusion

This paper proposes a multi-armed bandit local search solver called BandMaxSAT for Partial MaxSAT (PMS) and Weighted PMS (WPMS). The proposed bandit model can help the local search learn to select an appropriate soft clause to be satisfied in the current step when the algorithm falls into a feasible local optimum. We further apply the sampling strategy and the delayed reward method to improve our bandit model. As a result, the bandit model fits well with (W)PMS. Moreover, we propose an effective initialization method, called HyDeci, that prioritizes both unit and binary clauses when generating the initial assignments. HyDeci can improve BandMaxSAT by providing high-quality initial assignments, and could be useful to improve other local search MaxSAT solvers. Extensive experiments on all the (W)PMS instances from the incomplete tracks of the last four MSEs demonstrate that BandMaxSAT significantly outperforms the state-of-the-art local search (W)PMS algorithm SATLike3.0. Moreover, by combining BandMaxSAT with the complete solver TT-Open-WBO-Inc, the resulting solver BandMaxSAT-c outperforms the complete solvers SATLike-c, Loandra, and TT-Open-WBO-Inc.

The key issue in designing a local search algorithm is how to escape from local optima to find new high-quality search directions. In future work, we plan to generalize our bandit model to improve other local search algorithms for various NP-hard problems that need to select one among multiple candidates to escape from local optima.

Acknowledgements

This work is supported by National Natural Science Foundation of China (62076105) and Grant PID2019-111544GB-C21 funded by MCIN/AEI/10.13039/501100011033.

References

[Ansótegui *et al.*, 2022] Carlos Ansótegui, Felip Manyà, Jesus Ojeda, Josep M. Salvia, and Eduard Torres. Incomplete MaxSAT approaches for combinatorial testing. *Journal of Heuristics*, (in press), 2022.

[Arya and Yang, 2020] Sakshi Arya and Yuhong Yang. Randomized allocation with nonparametric estimation for contextual multi-armed bandits with delayed rewards. *Statistics & Probability Letters*, 164:108818, 2020.

[Berg *et al.*, 2019] Jeremias Berg, Emir Demirovic, and Peter J. Stuckey. Core-boosted linear search for incomplete MaxSAT. In *The Sixteenth International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, volume 11494, pages 39–56, 2019.

[Bonet *et al.*, 2019] Blai Bonet, Guillem Francès, and Hector Geffner. Learning features and abstract actions for computing generalized plans. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 2703–2710, 2019.

[Cai and Lei, 2020] Shaowei Cai and Zhendong Lei. Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability. *Artificial Intelligence*, 287:103354, 2020.

[Cai *et al.*, 2014] Shaowei Cai, Chuan Luo, John Thornton, and Kaile Su. Tailoring local search for partial MaxSAT. In *The Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2623–2629, 2014.

[Cai *et al.*, 2017] Shaowei Cai, Chuan Luo, and Haochen Zhang. From decimation to local search and back: A new approach to MaxSAT. In *The Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 571–577, 2017.

[Cai, 2015] Shaowei Cai. Balance between complexity and quality: Local search for minimum vertex cover in massive graphs. In *The Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 747–753, 2015.

[Chao and Franco, 1990] Ming-Te Chao and John V. Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k satisfiability problem. *Information Sciences*, 51(3):289–314, 1990.

[Chvátal and Reed, 1992] Vasek Chvátal and Bruce A. Reed. Mick gets some (the odds are on his side). In *The Thirty-Third Annual Symposium on Foundations of Computer Science*, pages 620–627, 1992.

[Ciampiconi *et al.*, 2020] Lorenzo Ciampiconi, Bishwamittra Ghosh, Jonathan Scarlett, and Kuldeep S. Meel. A MaxSAT-based framework for group testing. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 10144–10152, 2020.

[Demirovic and Musliu, 2017] Emir Demirovic and Nysret Musliu. MaxSAT-based large neighborhood search for high school timetabling. *Computers & Operations Research*, 78:172–180, 2017.

[Goffinet and Ramanujan, 2016] Jack Goffinet and Raghuram Ramanujan. Monte-carlo tree search for the maximum satisfiability problem. In *The Twenty-Second International Conference on Principles and Practice of Constraint Programming*, volume 9892, pages 251–267, 2016.

[Hu *et al.*, 2019] Yi-Qi Hu, Yang Yu, and Jun-Da Liao. Cascaded algorithm-selection and hyper-parameter optimization with extreme-region upper confidence bound bandit. In *The Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 2528–2534, 2019.

[Lassouaoui *et al.*, 2019] Mourad Lassouaoui, Dalila Boughaci, and Belaid Benhamou. A multilevel synergy thompson sampling hyper-heuristic for solving Max-SAT. *Intelligent Decision Technologies*, 13(2):193–210, 2019.

[Lattimore and Szepesvári, 2020] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[Lei and Cai, 2018] Zhendong Lei and Shaowei Cai. Solving (weighted) partial MaxSAT by dynamic local search for SAT. In *The Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 1346–1352, 2018.

[Lei *et al.*, 2021] Zhendong Lei, Shaowei Cai, Fei Geng, Dongxu Wang, Yongrong Peng, Dongdong Wan, Yiping Deng, and Pinyan Lu. Satlike-c: Solver description. *MaxSAT Evaluation 2021*, pages 19–20, 2021.

[Li and Manyà, 2021] Chu Min Li and Felip Manyà. MaxSAT, hard and soft constraints. In *Handbook of Satisfiability, second edition*, pages 903–927. IOS Press, 2021.

[Li *et al.*, 2006] Chu Min Li, Felip Manyà, and Jordi Planes. Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In *The Twenty-First National Conference on Artificial Intelligence*, pages 86–91, 2006.

[Luo *et al.*, 2017] Chuan Luo, Shaowei Cai, Kaile Su, and Wenxuan Huang. CCEHC: An efficient local search algorithm for weighted partial maximum satisfiability. *Artificial Intelligence*, 243:26–44, 2017.

[Nadel, 2019] Alexander Nadel. Anytime weighted maxsat with improved polarity selection and bit-vector optimization. In *2019 Formal Methods in Computer Aided Design*, pages 193–202, 2019.

[Ou *et al.*, 2019] Mingdong Ou, Nan Li, Cheng Yang, Shenghuo Zhu, and Rong Jin. Semi-parametric sampling for stochastic bandits with many arms. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 7933–7940, 2019.

[Slivkins, 2019] Aleksandrs Slivkins. Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, 12(1-2):1–286, 2019.