Farsighted Probabilistic Sampling:  
A General Strategy for Boosting MaxSAT Local Search Solvers

Jiongzhi Zheng\textsuperscript{1,2}, Jianrong Zhou\textsuperscript{1}, Kun He\textsuperscript{†1},  
\textsuperscript{1}\textsuperscript{School of Computer Science and Technology, Huazhong University of Science and Technology}  
\textsuperscript{2}\textsuperscript{Institute of Artificial Intelligence, Huazhong University of Science and Technology, China}  
\{jzzheng, jrzhou, brooklet60\}@hust.edu.cn

Abstract  
Local search has been demonstrated as an efficient approach for both Partial MaxSAT (PMS) and Weighted PMS (WPMS), denoted as (W)PMS, two practical generalizations to the typical combinatorial problem of MaxSAT. In this work, we observe that most (W)PMS local search solvers usually flip a single variable per iteration. Such a mechanism may lead to relatively low-quality local optimal solutions, and may limit the diversity of the search directions to escape from local optima. To this end, we propose a general strategy called farsighted probabilistic sampling (FPS) to replace the single flipping mechanism to boost the (W)PMS local search algorithms. FPS considers the benefit of continuously flipping a pair of variables, so as to find higher-quality local optimal solutions. Moreover, FPS presents an effective approach to escape from local optima by preferring the best to flip among the best sampled single variable and the best sampled variable pair. Extensive experiments demonstrate that our proposed FPS strategy significantly improves the (W)PMS state-of-the-art (local search) solvers, and FPS has an excellent generalization to various (Max)SAT local search solvers.

1 Introduction  
Maximum Boolean Satisfiability (MaxSAT) is an optimization version of the well-known Boolean Satisfiability (SAT) decision problem. Given a propositional formula in the Conjunctive Normal Form (CNF), MaxSAT aims at maximizing the number of satisfied clauses. Partial MaxSAT (PMS) is a generalization of MaxSAT, whose clauses are divided into hard and soft. The goal of PMS is to maximize the number of satisfied soft clauses with the constraint that all the hard clauses must be satisfied. In a more generalized situation, the soft clauses have nonnegative weights, and the resulting problem is called the Weighted Partial MaxSAT (WPMS), whose goal is to maximize the total weight of satisfied soft clauses with the same constraint that all the hard clauses must be satisfied. Both PMS and WPMS, denoted as (W)PMS, have many practical applications, such as planning [Zhang and Bacchus, 2012], timetabling [Cha et al., 1997], routing [Xu et al., 2003], group testing [Ciamapici et al., 2020], etc.

Local search is a well-studied category of incomplete algorithms for (W)PMS. One of the most common frameworks of (W)PMS local search algorithms is to improve the initial solution by iteratively flipping a variable that is selected as important for the local search according to a selection strategy [Luo et al., 2015; Luo et al., 2017]. Another important technique in local search algorithms is the clause weighting scheme [Cai et al., 2014; Luo et al., 2017; Lei and Cai, 2018], which associates dynamic weights (different from the original clause weights in MaxSAT instances) to clauses and uses the dynamic weights to guide the search direction. Recently, many studies have tried to develop effective (W)PMS local search algorithms by proposing novel clause weighting schemes and variable selection strategies [Cai et al., 2014; Luo et al., 2017; Lei and Cai, 2018; Lei and Cai, 2020].

The clause weighting schemes used in the local search algorithms are effective for (W)PMS. However, the variable selection strategies with the common single flipping mechanism in recent effective (W)PMS local search algorithms [Cai et al., 2014; Luo et al., 2015; Luo et al., 2017; Lei and Cai, 2018; Lei and Cai, 2020] might not be good enough. On the one hand, these algorithms all flip a single variable per iteration, which may cause the algorithm easily fall into a local optimum (i.e., flipping any variable can not increase the total weight of satisfied clauses), thus the quality of the local optimal solution might not be good enough. On the other hand, these algorithms all apply a random walk strategy (i.e., satisfying a randomly selected falsified clause) to escape from the local optimum. The high degree of randomness of this strategy may make it hard for these algorithms to find a good search direction to escape from the local optimum.

To handle these issues, we propose a general farsighted probabilistic sampling (FPS) variable selection strategy to replace the single flipping mechanism in the (W)PMS local search solvers. FPS employs a two-level technique, that allows the algorithm to look-ahead and consider the benefit of continuously flipping a pair of variables, as well as the probabilistic sampling approach. First, when no single flipping can improve the current solution (i.e., reaching a local optimum of the single flipping mechanism), FPS tries to look-ahead to find a pair of variables that flipping both can increase the total
weight of satisfied clauses. In this way, the quality of the local optimal solution can be improved. Second, the two-level technique and the sampling strategy can provide more and better search directions to escape from local optima. If FPS fails to improve the current solution by flipping a pair of variables, it will choose the best to flip among the best sampled single variable and the best sampled pair of variables.

There exist some related studies that apply the look-ahead or similar techniques for SAT [Li and Huang, 2005; Li et al., 2007; Wei et al., 2008; Cai and Su, 2013; Cai et al., 2013], and some studies that propose multiple flipping local search operators for SAT [Mali and Lipen, 2003] and (W)PMS [Reisch et al., 2020]. FPS is different from them and better than them. See details of related work in Appendix.

To evaluate the performance of FPS, we first apply FPS to the state-of-the-art (W)PMS local search algorithm, SATLike3.0 [Cai and Lei, 2020], by replacing its single flipping mechanism with FPS. The resulting algorithm is called MaxFPS. We compare MaxFPS with SATLike3.0 on all the benchmarks of PMS and WPMS from the incomplete track of the last four MaxSAT Evaluations. The results show that MaxFPS significantly outperforms SATLike3.0 for both the PMS and WPMS problems. Moreover, we do a further comparison with some of the state-of-the-art complete (W)PMS solvers, including SATLike-c\(^2\), Loandra [Berg et al., 2019], and TT-Open-WBO-Inc\(^2\). We replace the local search component (i.e., SATLike3.0) in SATLike-c with our MaxFPS algorithm, the resulting solver MaxFPS-c also outperforms these complete solvers for (W)PMS.

To evaluate the generalization of our proposed method, we further test our method on two other effective MaxSAT algorithms, Dist [Cai et al., 2014] and CCEHC [Luo et al., 2017], and even a well-known SAT solver, CCAnr [Cai et al., 2015]. Experiments show that these local search algorithms can all be improved significantly by replacing their single flipping mechanisms with our proposed FPS strategy, demonstrating the excellent generalization of our FPS strategy.

The main contributions of this work are as follows:

- We propose a general farsighted probabilistic sampling (FPS) strategy for boosting (W)PMS local search solvers. FPS can improve the local optimal solution of the single flipping mechanism by considering the benefit of continuously flipping a pair of variables.
- FPS could escape from the local optimum by preferring the best to flip among the best sampled single variable and the best sampled pair of variables. This mechanism can provide more and better search directions when the algorithm falls into a local optimum.
- Our method provides an effective way to apply the look-ahead strategy to boost (W)PMS local search solvers, and demonstrates that there is a great potential for the look-ahead strategy to be used for (W)PMS.

- Extensive experiments demonstrate that FPS significantly improves the state-of-the-art (W)PMS (local search) solvers, and FPS has good generalization and robustness for various (Max)SAT local search solvers.

2 Preliminaries

Given a set of Boolean variables \(\{x_1, ..., x_n\}\), a literal is either a variable \(x_i\) or its negation \(\neg x_i\), a clause \(c_i\) is a disjunction of literals (i.e., \(c_i = l_{i1} \lor l_{i2} \lor ... \lor l_{ij}\), where \(l_{ij}\) indicates the \(j\)-th literal in the \(i\)-th clause), and a conjunctive normal form (CNF) formula \(F\) is a conjunction of clauses (i.e., \(F = c_1 \land c_2 \land ... \land c_m\)). A complete assignment \(A\) is a mapping that assigns to each variable either 1 (true) or 0 (false). A literal \(x_i\) (resp. \(\neg x_j\)) is true if \(x_i = 1\) (resp. \(x_j = 0\)). A clause is satisfied if it has at least one true literal, and falsified otherwise. The flipping operator in local search algorithms for (Max)SAT on a variable is an operator to change the Boolean value of the variable.

Given a CNF formula, SAT aims at determining whether there is an assignment that satisfies all the clauses in the formula, and MaxSAT is an optimization extension of SAT that aims at finding an assignment that maximizes the number of satisfied clauses. The PMS, for which the clauses are divided into hard and soft, aims to find an assignment that satisfies all the hard clauses and satisfies as many soft clauses as possible. The WPMS, for which the soft clauses are associated with nonnegative weights, aims to find an assignment that satisfies all hard clauses and maximizes the total weight of the satisfied soft clauses, which is equivalent to minimize the total weight of the falsified soft clauses.

For a (W)PMS instance \(F\), an assignment \(A\) is regarded as feasible if it satisfies all the hard clauses in \(F\), and the cost of a feasible assignment \(A\), denoted by cost\((A)\), is defined to be the total weight of the falsified soft clauses. For convenience, the cost of any infeasible assignment is set to +\(\infty\).

The clause weighting techniques are widely used in the local search algorithms for (W)PMS. Local search algorithms with clause weighting techniques [Cai et al., 2014; Luo et al., 2015; Luo et al., 2017; Lei and Cai, 2018; Lei and Cai, 2020] modify the weights of clauses during the search, and apply the dynamic clause weights to guide the search direction. In this work, the commonly used scoring function score\((x)\) is defined as follows: score\((x)\) is the increase of the total dynamic weight of satisfied clauses caused by flipping \(x\).

3 The Proposed FPS Strategy

In this section, we propose a general farsighted probabilistic sampling (FPS) strategy for (W)PMS local search solvers, to replace the widely-used single flipping mechanism. This section first introduces our proposed MaxFPS algorithm, which incorporates FPS with the state-of-the-art (W)PMS local search algorithm, SATLike3.0 [Cai and Lei, 2020], to show how our FPS strategy works during the local search process. Then we show how FPS can be generalized to improve other (Max)SAT local search solvers.

\(^1\)https://maxsat-evaluations.github.io/2021/mse21-solver-src/incomplete/satlike-c.zip

\(^2\)https://maxsat-evaluations.github.io/2021/mse21-solver-src/incomplete/TT-Open-WBO-Inc-21.zip
Algorithm 1: MaxFPS

Input: A (W)PMS instance $F$, cut-off time $cutoff$, number of sampled clauses $sc\_num$, BMS parameter for sampling the second-level variables $sv\_num$

Output: A feasible assignment $A$ of $F$, or no feasible assignment found

1. $A :=$ an initial assignment; $A^* := A$
2. while running time $< cutoff$ do
   3. if $A$ is feasible & $cost(A) < cost(A^*)$ then
      4. $A^* := A$
      5. $Vars := \text{PickV}ars(F, A, sc\_num, sv\_num)$
      6. $A := A$ with the variables in $Vars$ flipped;
   7. if $A^*$ is feasible then return $A^*$;
   8. else return no feasible assignment found;

Algorithm 2: PickVars($F, A, sc\_num, sv\_num$)

Input: (W)PMS instance $F$, an assignment $A$, number of sampled clauses $sc\_num$, BMS parameter for sampling the second-level variables $sv\_num$

Output: a set of selected variables $Vars$

1. if $\GoodV ars := \{x|\text{score}(x) > 0\} \neq \emptyset$ then
   2. $v :=$ a variable in $\GoodV ars$ picked by BMS with parameter $t$
   3. $Vars := \{v\}$
   4. else if $\exists$ falsified hard clauses then
      5. $SC :=$ the set of $sc\_num$ randomly selected falsified hard clauses;
      6. else if $\exists$ falsified soft clauses then
         7. Initialize the set of first-level variables $FV := \emptyset$
         8. for $i := 1$ to $sc\_num$ do
            9. $v :=$ a random variable in $SC$
            10. if $v \notin FV$ then $FV := FV \cup \{v\}$
            11. $v_1 := \arg \max_{v \in FV} \text{score}(v)$
            12. $S_1 := \text{score}(v_1)$; $s_2 := -\infty$
            13. for $i := 1$ to $|FV|$ do
               14. $score_i := \text{score}(FV, v_i)$
               15. if $\GoodV ars_i := \{x|\text{score}(x) > 0\} \neq \emptyset$ then
                  16. $v :=$ a second-level variable in $\GoodV ars_i$
                  17. if $score_i + \text{score}(v) > s_2$ then
                      18. $Vars := \{FV, v\}$; return $Vars$
               19. else if $score_i + \text{score}(v) > s_2$ then
                  20. $s_2 := score_i + \text{score}(v)$
                  21. $v_2 := FV_i, v_2 := v$
            22. update clause weights();
            23. if $s_1 > s_2$ then $Vars := \{v_1\}$
            24. else $Vars := \{v_2, v_2\}$
            25. return $Vars$

3.1 MaxFPS: Incorporating FPS into SATLike3.0

MaxFPS actually applies the approach of generating the initial solution and the clause weighting scheme in SATLike3.0 [Cai and Lei, 2020], and replaces the variable selection strategy in SATLike3.0 with FPS. MaxFPS employs a two-level variable selection approach, that considers the benefit of continuously flipping a pair of variables, to help the algorithm find higher-quality local optimal solutions and good search directions to escape from local optima. Moreover, the probabilistic sampling approach and an early-stop strategy are applied in MaxFPS to improve the efficiency as well as the performance of the local search process.

The main flow of MaxFPS is presented in Algorithm 1, which repeats to select a set of variables (i.e., $Vars$) by the proposed two-level sampling approach (i.e., the PickVars function in line 5) and then flips the variables in $Vars$, until the cut-off time is reached. The procedure of PickVars function is presented in Algorithm 2. The variables to be flipped are determined in the following three cases:

**Case 1.** When the current solution is not a local optimum of the single flipping mechanism, i.e., there is at least one variable with positive score ($\GoodV ars$ in line 1 is not empty), PickVars selects a variable with positive score by a probabilistic sampling strategy called Best from Multiple Selections (BMS) [Cai, 2015]. BMS chooses $t$ random variables (with replacement) and returns the one with the highest score. Note that the selection strategy, and the BMS parameter $t$, in this case are the same as those in SATLike3.0 [Cai and Lei, 2020]. MaxFPS does not look-ahead in this case because it is unnecessary and inefficient to look-ahead when flipping a single variable can improve the current solution.

**Case 2.** When the current solution is a local optimum of the single flipping mechanism, PickVars tries to find a pair of variables that flipping both can improve the current solution (lines 5-24). The algorithm first samples $sc\_num$ (10 by default) falsified clauses $SC$ (hard clauses take precedence), and randomly samples a first-level variable from each sampled clause (lines 5-12). We denote $FV$ as the set of first-level variables. Note that $SC$ might contain duplicate clauses, as the sample is with replacement. Suppose clause $c$ appears $k$ times in $SC$, the algorithm actually randomly samples $k$ first-level variables in $c$ with replacement (lines 10-12).

Then the algorithm traverses each first-level variable, performs a pseudo flipping (which will not change the current assignment, but just to look-ahead), and selects a second-level variable by the BMS strategy with parameter $sv\_num$ (50 by default) if there is at least one variable with positive score after the pseudo flipping (lines 15-19).

During the search process, once a pair of variables (consisting of a first-level variable and a second-level variable selected by the BMS strategy) that flipping both can reduce the total weight of falsified clauses is found, an early-stop strategy is applied to terminate the search process and returns the
pair of variables (lines 20-21).

Case 3. When PickVars fails to determine the variables to be flipped in Cases 1 and 2, we say it falls into a local optimum. The algorithm first updates the clause weights (line 25) by the clause weighting strategy as in SATLike3.0, and then tries to escape from the local optimum by greedily selecting the variables to be flipped (lines 26-27). Specifically, the algorithm selects to flip the first-level variable with the highest score if the maximum score of the sampled first-level variables is larger than the maximum total score of the sampled pairs of variables (i.e., $s_1 > s_2$), and vice versa.

In summary, the selection strategy in Case 1 is commonly used in SATLike and SATLike3.0, while the strategies in Cases 2 and 3 can help the algorithm find better local optimal solutions and better search directions to escape from local optima than the single flipping mechanism, respectively.

3.2 Generalizing FPS to Other Solvers
Recent effective (Max)SAT local search solvers such as SATLike(3.0), Dist [Cai et al., 2014], CCEHC [Luo et al., 2017], and CCAnr [Cai et al., 2015] follow similar single flipping mechanisms. They maintain a set $V$ (e.g., the GoodVars in line 1 in Algorithm 2) recording variables that flipping any of them leads to a better solution than the current one. When $V \neq \emptyset$, i.e., the search does not reach a local optimum, they select to flip a variable of $V$. Otherwise, they first update the dynamic clause weight according to their clause weighting schemes, and then perform their local optimum escaping methods, which all contain the simple random walk strategy (i.e., satisfying a random selected falsified clause). These algorithms are mainly different on the initialization method, the clause weighting scheme, and the definition of $V$.

FPS can be simply applied to improve any of these algorithms as follows. First, when a local optimum for the single flipping mechanism is reached, try to look-ahead to further improve the local optimum as Case 2 does. Second, replace the random walk strategy with the local optimum escaping method in FPS as Case 3 does. Extensive experiments show that Dist, CCEHC, and CCAnr can all be improved significantly by applying FPS.

4 Experimental Results
For experiments, we first compare MaxFPS with the state-of-the-art (W)PMS local search algorithm, SATLike3.0. Then we replace the local search component of the powerful (W)PMS complete solver, SATLike-c, with our MaxFPS algorithm, and compare the resulting solver MaxFPS-c with some of the state-of-the-art (W)PMS complete solvers, SATLike-c, Loandra [Berg et al., 2019], and TT-Open-WBO-Inc. We further apply FPS to other (Max)SAT local search solvers based on single flipping mechanisms, including Dist [Cai et al., 2014], CCEHC [Luo et al., 2017], and CCAnr [Cai et al., 2015], to evaluate the generalization of FPS. Finally, we do ablation studies for clarity and analysis.

4.1 Experimental Setup
MaxFPS is implemented in C++ and compiled by g++. The parameters in the local search process in MaxFPS include the number of sampled clauses $sc_{num}$, and the BMS parameter for sampling the second-level variables $sv_{num}$. The larger the value of $sc_{num}$ or $sv_{num}$, the higher the quality of the local optimas for MaxFPS, and the lower the algorithm efficiency. Thus, we set the domain of these two parameters to $[5, 100]$ for both of them, and tune them on all the (W)PMS instances from the incomplete track of MaxSAT Evaluation 2017 (MSE2017). The default values of these two parameters are as follows: $sc_{num} = 10$, $sv_{num} = 50$. Other parameters include the BMS parameter $t$, and the parameters in the clause weighting scheme are set to the same as the default settings in SATLike3.0 [Cai and Lei, 2020].

For (W)PMS experiments, we select all benchmarks of (W)PMS problems from the incomplete track of the last four MSEs, i.e., MSE2018-MSE2021, for comparison and analysis. We use (W)PMS$_{y}$ to represent the (W)PMS benchmarks in MSE of year $y$ (e.g., WPMS$_{2021}$ represents the WPMS benchmark in MSE2021, and so on). For SAT experiments, we select all benchmarks of SAT problems from the main track of the last three SAT Competition (we cannot access SAT Competition 2018). We denote the SAT benchmarks in SAT Competition of year $y$ as SAT$_y$.

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. For (W)PMS, each instance is calculated once by each algorithm with two different time limits (60 seconds and 300 seconds as in MSEs). For SAT, each instance is calculated once by each algorithm with a time limit of 3600 seconds. The best results for each benchmark are highlighted in bold.

4.2 MaxFPS versus SATLike3.0
We first compare MaxFPS with SATLike3.0 on all the (W)PMS benchmarks. The results for both 60 and 300 seconds of time limits are summarized in Table 1. Column #inst. indicates the number of instances of each benchmark, column #win. indicates the number of instances in which each solver finds the best solution among all solvers in the table, and column time indicates the average time of doing so over such “winning” instances. We could observe that the

| Benchmark | #inst. | MaxFPS | SATLike3.0 |
|-----------|--------|--------|------------|
| PMS$_{2018}$ | 153 | 74 | 11.498 | 56 | 18.897 |
| PMS$_{2019}$ | 299 | 165 | 10.579 | 141 | 13.090 |
| PMS$_{2020}$ | 262 | 149 | 10.168 | 107 | 12.042 |
| PMS$_{2021}$ | 155 | 73 | 10.520 | 59 | 10.065 |
| WPMS$_{2018}$ | 172 | 117 | 19.991 | 61 | 10.143 |
| WPMS$_{2019}$ | 297 | 205 | 18.980 | 101 | 14.547 |
| WPMS$_{2020}$ | 253 | 168 | 19.436 | 81 | 20.107 |
| WPMS$_{2021}$ | 151 | 79 | 27.210 | 55 | 27.554 |

Table 1: Comparison of MaxFPS and SATLike3.0 with 60 or 300 seconds of time limit on all the tested (W)PMS benchmarks.
results of MaxFPS are significantly better than those of SATLike-3.0 for both PMS and WPMS, in either of the two time limits. Specifically, the winning PMS instances for MaxFPS are about 17-39% (resp. 26-36%) greater than those for SATLike-3.0 with 60 (resp. 300) seconds of time limit, the winning WPMS instances for MaxFPS are about 44-107% (resp. 16-90%) greater than those for SATLike-3.0 with 60 (resp. 300) seconds of time limit, indicating a significant improvement.

Moreover, we present the results of MaxFPS on the instances (with a time limit of 300 seconds) that MaxFPS yields better solutions than the best-known solutions (BKS).

Table 2: Results of MaxFPS on the WPMS instances that MaxFPS yields better solutions than the best-known solutions (BKS).

### MaxFPS-c versus Complete Solvers

We then combine MaxFPS with the complete (W)PMS solver TT-Open-WBO-Inc (TT-OWI) as SATLike-c does (which combines SATLike-3.0 with TT-OWI), and compare the resulting solver MaxFPS-c with some of the state-of-the-art (W)PMS complete solvers, including SATLike-c, Loandra, and TT-OWI. We use the scoring function in MSE2021 as the metric for comparison, since it is more suitable for evaluating multiple solvers. The scoring function actually indicates how close the solutions are to the best-known solutions.

Specifically, suppose $C^*$ is the cost of the best-known solution of an instance recorded in MSES in Tables 2 and 3. The results are the number (or total weight) of falsified soft clauses of the solutions. Column Gap indicates the gap of the solution found by the $i$-th solver for an instance recorded in MSE2021. The scoring function indicated how close the solutions are to the best-known solutions.

![Graphic: Comparison of MaxFPS-c and Complete Solvers]

### Table 3: Results of MaxFPS on the WPMS instances that MaxFPS yields better solutions than the best-known solutions (BKS).}

![Table: Results of MaxFPS on the WPMS instances that MaxFPS yields better solutions than the best-known solutions (BKS).]

The excellent performance of our proposed method.

4.3 MaxFPS-c versus Complete Solvers

We then combine MaxFPS with the complete (W)PMS solver TT-Open-WBO-Inc (TT-OWI) as SATLike-c does (which combines SATLike-3.0 with TT-OWI), and compare the resulting solver MaxFPS-c with some of the state-of-the-art (W)PMS complete solvers, including SATLike-c, Loandra, and TT-OWI. We use the scoring function in MSE2021 as the metric for comparison, since it is more suitable for evaluating multiple solvers. The scoring function actually indicates how close the solutions are to the best-known solutions. Specifically, suppose $C^*$ is the cost of the best-known solution of an instance 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 $C_{i+1} = \min(C^*, C_1, C_2, C_3, C_4) + 1 \in [0, 1]$ (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.

The comparison results of MaxFPS-c and the other three complete solvers are shown in Table 4. With a time limit of 60 seconds, MaxFPS-c yields the highest score on all the benchmarks except PMS_2019 and PMS_2020, with a time

![Table: Comparison of MaxFPS-c with Complete Solvers]
limit of 300 seconds. MaxFPS-c yields the highest score on all the benchmarks except WPMS_2019. The results demonstrate that MaxFPS-c outperforms the three state-of-the-art (W)PMS complete solvers on most benchmarks, and our proposed approach is effective.

### 4.4 Generalize to Other Solvers

We further test our proposed farsighted sampling strategy on other (Max)SAT local search algorithms based on the single flipping mechanism to evaluate the generalization of our method. We choose the PMS algorithm Dist [Cai et al., 2014], the WPMS algorithm CCEHC [Luo et al., 2017], and the well-known SAT algorithm CCAnr [Cai et al., 2015], and use Dist-FPS, CCEHC-FPS, CCAnr-FPS to represent the algorithms obtained by combining them with our method, respectively. The comparison results of Dist-FPS versus Dist, CCEHC-FPS versus CCEHC, CCAnr-FPS versus CCAnr are shown in Tables 5, 6, and 7, respectively.

From the results one can observe that, the three solvers can all be improved significantly by our FPS method. Note that we also set the parameters $sc_{num} = 10$ and $sv_{num} = 50$ in Dist-FPS, CCEHC-FPS, and CCAnr-FPS as in the MaxFPS solver, and did not retune the parameters for them. However, the results are still promising, indicating that our method has good generalization and robustness for (W)PMS and even for SAT. In addition, the results for PMS_2021 and WPMS_2021 in Tables 5 and 6 are not good. This is because our method is not good at calculating the instances of the decision-tree classes in these two benchmarks, which is the largest instance class in both PMS_2021 and WPMS_2021 (decision-tree class contains 23 among 155 instances in PMS_2021, and 24 among 151 instances in WPMS_2021). Similar results can be found in Table 1.

### 4.5 Ablation Study

Finally, we do ablation studies to evaluate the effectiveness of each part in our proposed farsighted probabilistic sampling (FPS) strategy, including the sampling strategy, the method of escaping from local optima, and the two-level look-ahead search process, by comparing MaxFPS with its several variants. Due to the limited space, we put the results in Appendix. The results demonstrate that these components in FPS are effective and reasonable.

### 5 Conclusion

In this work, we propose an effective general farsighted probabilistic sampling (FPS) strategy, that combines the look-ahead technique and probabilistic sampling method, to replace the mechanism of flipping a single variable per iteration step, that is widely used in (Max)SAT local search solvers. The proposed strategy can significantly improve the local optimum of the single flipping mechanism. Moreover, the look-ahead technique, as well as the probabilistic sampling method, can provide more and better search directions to escape from local optima. Look-ahead is not a new technique, but how to look-ahead is the magic. This paper proposes an effective way to apply this technique to boost MaxSAT or even SAT local search solvers. We also demonstrate that there is a great potential for the look-ahead technique to be used for (Max)SAT. Extensive experiments show that our proposed FPS strategy significantly improves the (W)PMS state-of-the-art (lo-
cal search) solvers. Moreover, the results also show that FPS can boost various (Max)SAT (local search) solvers, including SATLike3.0, SATLike-c, Dist, CCEHC, and CCAnr, indicating the excellent generalization of the proposed FPS strategy.

In future work, we plan to further explore the potential of the look-ahead strategy for (Max)SAT.

References

[Berg et al., 2019] Jeremias Berg, Emir Demirovic, and Peter J. Stuckey. Core-boosted linear search for incomplete MaxSAT. In Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 16th International Conference, CPAIOR 2019, volume 11494, pages 39–56, 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 and Su, 2013] Shaowei Cai and Kaile Su. Local search for boolean satisfiability with configuration checking and subscore. Artificial Intelligence, 204:75–98, 2013.

[Cai et al., 2013] Shaowei Cai, Kaile Su, and Chuan Luo. Improving WalkSAT for random k-satisfiability problem with k > 3. In Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2013, pages 145–151, 2013.

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

[Cai et al., 2015] Shaowei Cai, Chuan Luo, and Kaile Su. CCAnr: A configuration checking based local search solver for non-random satisfiability. In Theory and Applications of Satisfiability Testing, SAT 2015, volume 9340, pages 1–8, 2015.

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

[Cha et al., 1997] Byungki Cha, Kazuo Iwama, Yahiko Kambayashi, and Shuichi Miyazaki. Local search algorithms for partial MAXSAT. In Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, pages 263–268, 1997.

[Ciampiconi et al., 2020] Lorenzo Ciampiconi, Bishwamitra Ghosh, Jonathan Scarlett, and Kuldeep S. Meel. A MaxSAT-based framework for group testing. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, pages 10144–10152, 2020.

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

[Lei and Cai, 2020] Zhendong Lei and Shaowei Cai. Nudist: An efficient local search algorithm for (weighted) partial MaxSAT. The Computer Journal, 63(9):1321–1337, 2020.

[Li and Huang, 2005] Chu Min Li and Wenqi Huang. Diversification and determinism in local search for satisfiability. In Theory and Applications of Satisfiability Testing. 8th International Conference, SAT 2005, volume 3569, pages 158–172, 2005.

[Li et al., 2007] Chu Min Li, Wanxia Wei, and Harry Zhang. Combining adaptive noise and look-ahead in local search for SAT. In Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, volume 4501, pages 121–133, 2007.

[Luo et al., 2015] Chuan Luo, Shaowei Cai, Wei Wu, Zhong Jie, and Kaile Su. CCLS: An efficient local search algorithm for weighted maximum satisfiability. IEEE Transactions on Computers, 64(7):1830–1843, 2015.

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

[Mali and Lipen, 2003] Amol Dattatraya Mali and Yevgeny Lipen. MFSA T: A SAT solver using multi-flip local search. In 15th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2003, pages 84–93, 2003.

[Reisch et al., 2020] Julian Reisch, Peter Großmann, and Natalia Kliewer. Stable resolving - a randomized local search heuristic for MaxSAT. In KI 2020: Advances in Artificial Intelligence - 43rd German Conference on AI, volume 12325, pages 163–175, 2020.

[Wei et al., 2008] Wanxia Wei, Chu Min Li, and Harry Zhang. A switching criterion for intensification and diversification in local search for SAT. Journal on Satisfiability, Boolean Modeling and Computation, 4(2-4):219–237, 2008.

[Xu et al., 2003] Hui Xu, Rob A. Rutenbar, and Kareem A. Sakallah. sub-SAT: A formulation for relaxed boolean satisfiability with applications in routing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 22(6):814–820, 2003.

[Zhang and Bacchus, 2012] Lei Zhang and Fahiem Bacchus. MAXSAT heuristics for cost optimal planning. In Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2012, pages 1846–1852, 2012.
In the Appendix, we first provide the related work about the studies that apply the look-ahead or similar techniques, as well as the multiple flipping operators for (Max)SAT, then present ablation studies for evaluating the effectiveness of each part in the proposed farsighted probabilistic sampling (FPS) strategy.

6 Related Work

The look-ahead technique [Li and Huang, 2005; Li et al., 2007; Wei et al., 2008] and an approach using the second level score [Cai and Su, 2013; Cai et al., 2013] are proposed for SAT. To evaluate the benefit of flipping a variable, these two approaches consider the immediate benefit of the flipping, as well as the future benefit that might be obtained after the flipping. However, these algorithms still prioritize the immediate benefit of flipping a single variable, and always perform a single flipping per iteration. For example, the look-ahead technique is used to select one from the two best single flipping variables [Li et al., 2007; Wei et al., 2008] and the second level score technique is used only for breaking ties [Cai and Su, 2013; Cai et al., 2013]. Thus the look-ahead and second level score techniques play a minor role in selecting the flipping variable.

Intuitively, when the algorithms fall into a local optimum of the single flipping mechanism, i.e., flipping any variable cannot improve the current solution, prioritizing the immediate benefit than the future benefit is short-sighted, and may lead to a poor search direction. In comparison, our proposed FPS strategy prefers to yield a better solution without preferring the immediate benefit or the future benefit, which leads to better solutions and better search directions.

There are also some studies proposing multiple flipping local search operators for SAT [Mali and Lipen, 2003] and (W)PMS [Reisch et al., 2020]. But they use an exhaustive method rather than probability sampling in the local search process, thus are less efficient when solving large instances. In contrast, our proposed FPS strategy combines the advantages of look-ahead and probability sampling, as well as the clause weighting scheme in SATLike3.0, thus it is more effective and efficient for solving the (W)PMS.

In summary, our proposed FPS strategy is different from the methods described above. Extensive experiments demonstrate that our FPS significantly improves the (W)PMS (local search) state-of-the-art, and can be generalized to various (W)PMS or even SAT local search algorithms.

7 Ablation Study

We do ablation studies to evaluate the effectiveness of each part in FPS, including the sampling strategy, the local optimum escaping method, and the two-level look-ahead search process, by comparing MaxFPS with its several variants.

We first compare MaxFPS with its two variants, MaxFPS$_1$ and MaxFPS$_2$, to show the effectiveness of the probabilistic sampling approach in MaxFPS. MaxFPS$_1$ sets the number of sampled clauses $\text{sc}_{\text{num}}=1$, while in MaxFPS$_2$, if there is at least one falsified hard clause, the set of the first-level variables contains all the variables in falsified hard clauses, otherwise in falsified soft clauses. The results of MaxFPS and these two variants are summarised in Table 8.

From the results we observe that MaxFPS significantly outperforms MaxFPS$_1$, demonstrating that sampling multiple first-level variables is more effective than only selecting one first-level variable. This is because the sampled variables from various falsified variables increase the opportunity to improve the local optimum for the single flipping mechanism, and provide more and better directions to escape from the local optimum. We can also observe that MaxFPS significantly outperforms MaxFPS$_2$, demonstrating that the look-ahead from all variables in falsified clauses is inefficient, and the sampling approach in MaxFPS can improve the efficiency.

We then compare MaxFPS with MaxFPS$_3$ to show the effectiveness of the method of escaping from local optima in MaxFPS (i.e., selecting variables in Case 3 in Section 3). MaxFPS$_3$ replaces the variable selection strategy in Case 3 (lines 26-27 in Algorithm 2) with the random walk 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] to escape from local optima. Specifically, if MaxFPS$_3$ cannot decide the variables to be flipped in Cases 1 and 2, the algorithm first randomly selects a falsified clause (hard clauses take precedence), then selects the variable with the highest score in this clause.

The comparison results of MaxFPS and MaxFPS$_3$ are shown in Table 9. The results show that our proposed method of escaping from local optima that selecting the best to flip among the best sampled single variable and the best sampled pair of variables is significantly better than the random walk strategy that is widely used in recent (W)PMS local search algorithms. This is because our method can provide more and better directions to escape from local optima.

Finally, we compare MaxFPS with another two variants to analyze the reasonableness of the two-level search process in MaxFPS. MaxFPS$_4$ extends the two-level search approach in MaxFPS to three levels. In MaxFPS$_4$, the third-level variables are selected by the same approach of selecting the second-level variables in MaxFPS, and the second-level variables in MaxFPS$_4$ are sampled from the neighbors of the corresponding first-level variable. Another alternative MaxFPS$_5$ is a variant of MaxFPS without the early-stop strategy. The results of MaxFPS and these two variants with 60 or 300 seconds of time limit are summarised in Table 10.

From the results we can see that MaxFPS performs better than MaxFPS$_4$, indicating that MaxFPS with three-level search is worse than MaxFPS with two-level search, since the three-level search process takes a much longer time and hence is inefficient. We also observe that MaxFPS signifi-
Table 8: Comparison of MaxFPS and its variants, MaxFPS<sub>1</sub>, MaxFPS<sub>2</sub>, with 60 or 300 seconds of time limit on all the tested (W)PMS benchmarks.

| Benchmark | #inst. | MaxFPS<sub>1</sub> | MaxFPS<sub>2</sub> | MaxFPS<sub>3</sub> |
|-----------|--------|------------------|------------------|------------------|
| PMS<sub>2018</sub> | 153 | 14.317 | 16.859 | 20.178 |
| PMS<sub>2019</sub> | 299 | 10.363 | 11.807 | 14.814 |
| PMS<sub>2020</sub> | 262 | 10.072 | 11.223 | 14.104 |
| PMS<sub>2021</sub> | 155 | 10.283 | 7.604 | 11.556 |
| WPM<sub>2018</sub> | 172 | 19.810 | 17.062 | 15.800 |
| WPM<sub>2019</sub> | 297 | 9.160 | 7.215 | 8.570 |
| WPM<sub>2020</sub> | 253 | 29.184 | 21.252 | 17.707 |
| WPM<sub>2021</sub> | 151 | 29.184 | 21.252 | 17.707 |

Table 9: Comparison of MaxFPS and its variant MaxFPS<sub>4</sub>, with 60 or 300 seconds of time limit on all the tested (W)PMS benchmarks.

| Benchmark | #inst. | MaxFPS<sub>4</sub> | MaxFPS<sub>5</sub> | MaxFPS<sub>6</sub> |
|-----------|--------|------------------|------------------|------------------|
| PMS<sub>2018</sub> | 153 | 53.306 | 53.215 | 55.684 |
| PMS<sub>2019</sub> | 299 | 46.602 | 52.164 | 63.701 |
| PMS<sub>2020</sub> | 262 | 37.845 | 40.377 | 52.697 |
| PMS<sub>2021</sub> | 155 | 29.184 | 21.252 | 17.707 |
| WPM<sub>2018</sub> | 172 | 70.396 | 76.653 | 51.797 |
| WPM<sub>2019</sub> | 297 | 88.019 | 61.962 | 67.961 |
| WPM<sub>2020</sub> | 253 | 96.278 | 82.994 | 71.885 |
| WPM<sub>2021</sub> | 151 | 110.312 | 108.481 | 110.710 |

Table 10: Comparison of MaxFPS and its alternatives, MaxFPS<sub>4</sub>, MaxFPS<sub>5</sub>, with 60 or 300 seconds of time limit on all the tested (W)PMS benchmarks.

| Benchmark | #inst. | MaxFPS<sub>4</sub> | MaxFPS<sub>5</sub> | MaxFPS<sub>6</sub> |
|-----------|--------|------------------|------------------|------------------|
| PMS<sub>2018</sub> | 153 | 15.584 | 10.021 | 55.684 |
| PMS<sub>2019</sub> | 299 | 11.072 | 141 | 52.697 |
| PMS<sub>2020</sub> | 262 | 10.853 | 118 | 52.697 |
| PMS<sub>2021</sub> | 155 | 11.094 | 57 | 69.637 |
| WPM<sub>2018</sub> | 172 | 17.524 | 128 | 51.797 |
| WPM<sub>2019</sub> | 297 | 18.443 | 107 | 67.961 |
| WPM<sub>2020</sub> | 253 | 18.443 | 107 | 67.961 |
| WPM<sub>2021</sub> | 151 | 110.312 | 110.710 | 51.797 |

significantly outperforms MaxFPS<sub>5</sub>, demonstrating that the early-stop strategy could significantly improve the efficiency and performance of MaxFPS.