entropy

MDPI

Article

Initial Solution Generation and Diversified Variable Picking in Local Search for (Weighted) Partial MaxSAT

Zaijun Zhang [1,2], Jincheng Zhou [2,3,*], Xiaoxia Wang [1,2], Heng Yang [1,2] and Yi Fan [1,2,4]

[1] School of Mathematics and Statistics, Qiannan Normal University for Nationalities, Duyun 558000, China
[2] Key Laboratory of Complex Systems and Intelligent Optimization of Guizhou Province, Duyun 558000, China
[3] School of Computer and Information, Qiannan Normal University for Nationalities, Duyun 558000, China
[4] Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China
* Correspondence: zjc81@sgmtu.edu.cn; Tel.: +86-138-0949-0127

Abstract: The (weighted) partial maximum satisfiability ((W)PMS) problem is an important generalization of the classic problem of propositional (Boolean) satisfiability with a wide range of real-world applications. In this paper, we propose an initialization and a diversification strategy to improve local search for the (W)PMS problem. Our initialization strategy is based on a novel definition of variables' structural entropy, and it aims to generate a solution that is close to a high-quality feasible one. Then, our diversification strategy picks a variable in two possible ways, depending on a parameter: continuing to pick variables with the best benefits or focusing on a clause with the greatest penalty and then selecting variables probabilistically. Based on these strategies, we developed a local search solver dubbed *ImSATLike*, as well as a hybrid solver *ImSATLike-TT*, and experimental results on (weighted) partial MaxSAT instances in recent MaxSAT Evaluations show that they outperform or have nearly the same performances as state-of-the-art local search and hybrid competitors, respectively, in general. Furthermore, we carried out experiments to confirm the individual impacts of each proposed strategy.

Keywords: maximum satisfiability; structural entropy; local search; heuristic search

Citation: Zhang, Z.; Zhou, J.; Wang, X.; Yang, H.; Fan, Y. Initial Solution Generation and Diversified Variable Picking in Local Search for (Weighted) Partial MaxSAT. *Entropy* **2022**, *24*, 1846. https://doi.org/10.3390/e24121846

Academic Editors: Marcin Sosnowski, Jaroslaw Krzywanski, Karolina Grabowska, Dorian Skrobek, Ghulam Moeen Uddin, Yunfei Gao, Anna Zylka, Anna Kulakowska and Bachil El Fil

Received: 9 November 2022
Accepted: 15 December 2022
Published: 18 December 2022

1. Introduction

The maximum satisfiability (MaxSAT) problem is an optimization version of the Boolean satisfiability (SAT) problem, which is a prototypical NP-complete problem. In the context of the SAT and MaxSAT problems, a propositional formula F is usually expressed in conjunctive normal form (CNF) [1], i.e., $F = \bigwedge_i \bigvee_j l_{ij}$, where each l_{ij} is a literal, which is either a Boolean variable or its negation. A CNF formula can be expressed as a set of clauses, where a clause is a disjunction of literals, and each CNF formula is a conjunction of clauses.

Given a formula in CNF, the MaxSAT problem is to seek an assignment that minimizes the number of unsatisfied clauses in the formula. The partial maximum satisfiability (PMS) problem generalizes the MaxSAT problem to involve both hard and soft clauses. It aims to find a solution that minimizes the number of violated soft clauses while satisfying all the hard ones. The weighted partial maximum satisfiability (WPMS) problem is a generalization of the PMS problem, which further associates each soft clause with a positive weight and tries to locate a solution that minimizes the total weight of violated soft clauses. The MaxSAT, PMS, and WPMS problems are all NP-hard, and it is well known that optimum solutions are hard to approximate [2]. Obviously, MaxSAT is a special case of PMS, where the set of hard clauses is empty, and PMS is a special case of WPMS, where each soft clause is associated with the same weight.

Like other combinatorial problems, real-world applications usually contain hard and soft constraints [3], and soft ones often have different importance. Encoding such problems into PMS and WPMS problems is natural and straightforward [4–7]. In fact, real-world

problems such as computational protein design [8,9], set covering [10], coalition structure generation [11], and large-scale road sensing through crowdsourced vehicles [12] can be encoded and solved as PMS or WPMS problems.

There are two popular kinds of algorithms for solving MaxSAT and also its extensions: complete and stochastic local search (SLS) algorithms. Complete algorithms are able to confirm the optimality of the returned solution at the end, but they may fail to return a high-quality one for large-scale instances within reasonable time [13]. These algorithms can further be classified into two main subcategories: branch and bound MaxSAT algorithms [14–17], which are based on David–Putnam–Loveland–Logemann (DPLL) procedures [18,19], and SAT-based ones [20–27], which call efficient conflict-driven clause learning (CDCL) SAT solvers [28,29] to solve a sequence of SAT problems. Considering recent MaxSAT Evaluations, we found that branch and bound algorithms are superior on crafted benchmarks, while SAT-based ones perform better on application benchmarks (https://maxsat-evaluations.github.io/2018/, accessed on 8 November 2022). Furthermore, SAT-based solvers, namely *Open-WBO* [30], *LinSBPS*, and *TT-Open-WBO-inc* [31], performed extraordinarily in incomplete solver tracks of MaxSAT Evaluations 2018 and 2019 (https://helda.helsinki.fi/bitstream/handle/10138/237139/mse18_proceedings.pdf?sequence=1, https://helda.helsinki.fi/bitstream/handle/10138/306989/mse19proc.pdf?sequence=1 accessed on 8 November 2022). On the other hand, SLS algorithms are often able to find satisfactory solutions within a reasonable time frame [3,32], although they do not guarantee the optimality of the solution they find. These algorithms are usually variants or refinements of two prototype solvers, i.e., *GSAT* [33] and *WalkSAT* [34].

1.1. Local Search for MaxSAT

Recently, significant breakthroughs have been achieved by SLS algorithms for solving PMS and WPMS problems, resulting in state-of-the-art SLS algorithms, namely *Dist* [3] together with its improvement *DistUP* [35], *CCEHC* [36], and *SATLike* together with one of its variants *SATLike* 3.0 [37]. The *Dist* algorithm shows great success in solving PMS and won several categories in the incomplete solver track of the MaxSAT Evaluation 2014. Furthermore, it competes well with state-of-the-art complete algorithms on some classes of PMS application instances, such as advanced encryption standard and protein [3]. Furthermore, *Dist* can also be adapted to solve WPMS and is still one of the current best SLS algorithms for solving WPMS. The *DistUP* algorithm, an improvement of *Dist*, which incorporates unit propagation in its initialization procedure, shows improvement over *Dist* on industrial instances. However, *CCLS*, *Dist*, and *DistUP* are not dedicated to solving WPMS, and their performance for solving WPMS could be further improved. This motivates the design of a solver dubbed *CCEHC* [36], which is the state-of-the-art on WPMS instances. The *CCEHC* algorithm extends the framework of *CCLS* with an extra heuristic, which emphasizes hard clauses (EHCs). With a strong focus on hard clauses, the EHC heuristic has three components: a variable selection mechanism, which focuses on a forbidding mechanism called configuration checking based only on hard clauses, a weighting scheme for hard clauses, as well as an approach of a biased random walk. Later, *SATLike* and its variant *SATLike* 3.0 outperformed previous solvers in solving PMS and WPMS problems. Moreover, they are thought to be the first SLS solvers that compete well with SAT-based ones.

Despite the significant breakthroughs above, there is still a gap between the performances of SLS solvers and those of SAT-based ones. To make matters worse, the algorithms for the former ones are more complicated than those for the latter ones. We believe that these drawbacks may be due to certain structures of PMS and WPMS problems. For example, there are two kinds of clauses, hard and soft ones. Furthermore, these drawbacks could also be caused by improper selections of initial solutions (starting points of local search) or diversifying variables. In this sense, the detailed analysis of the structures of PMS and WPMS problems, as well as suitable initial solutions and diversifying variables may lead to significant improvements.

1.2. Our Contributions

In this work, we develop an SLS solver named *ImSATLike* together with a hybrid one dubbed *ImSATLike-TT* based on two novel strategies, i.e., generating a high-quality starting point for local search and selecting a promising variable for diversification. Firstly, our initial solution generation is based on a notion called *variable entropy*. The resulting solution is closer to high-quality feasible solutions compared to those generated in the most common and traditional approach, i.e., pure random assignments. Experiments showed that this strategy is able to improve the efficiency of locating a satisfactory solution. Secondly, when the search is trapped in local optima, it will focus more on three types of variables: (1) those in the whole formula, which has the greatest benefit; (2) those lying in a clause with the greatest penalty; (3) those causing the least clauses to become unsatisfied. Thirdly, we also develop a hybrid solver *ImSATLike-TT*, which combines *ImSATLike* with a state-of-the-art SAT-based solver *TT-Open-WBO-inc* [31], and this solver presents satisfactory performances on (weighted) partial MaxSAT instances in recent MaxSAT Evaluations.

The rest of this paper is organized as follows. Some necessary concepts and basic notations are introduced in Section 2. The strategy of generating an initial solution based on variables' structural entropy is introduced in Section 3. In Section 4, we introduce the diversifying variable selection strategy based on clause penalties. Our algorithm and the experimental evaluations are presented in Sections 5 and 6, respectively. In Section 7, we give some conclusions and the future work.

2. Preliminaries

Throughout this paper, we talk about propositional logic. Given a set of n Boolean variables (also called propositional atoms) $V = \{x_1, \cdots, x_n\}$, a literal l is either x_k or $\neg x_k$, where $k = 1, 2, \cdots, n$. A clause $C = l_1 \vee \cdots \vee l_s$ is a disjunction of literals, where s is called the (clause) length of C. Then, we use $\mathcal{V}(C)$ as the set of variables in C. In addition, if $l = x_k$ (respectively $l = \neg x_k$) is a literal in C, then we say that x_k occurs positively (respectively negatively) in C, and we can also say that C contains x_k's positive (respectively negative) occurrence.

A formula F in conjunctive normal form (CNF) is a conjunction of clauses, i.e., $F = C_1 \wedge \cdots \wedge C_t$, where t is called the number of clauses in F. Given a CNF formula F, we abuse $\mathcal{V}(F)$ to denote the set of variables in F, i.e., $\mathcal{V}(F) = \bigcup_{1 \leq j \leq t} \mathcal{V}(C_j)$. Furthermore, we use $\mathcal{C}(F)$ to denote the set of clauses in F, i.e., $\mathcal{C}(F) = \{C_1, \cdots, C_t\}$. In the MaxSAT problem, as well as its variants, clauses are usually partitioned into hard and soft ones, so we use $\mathcal{C}_h(F)$ and $\mathcal{C}_s(F)$ to denote the set of hard and soft clauses in F, respectively.

Two different variables, namely x and y, are said to be neighbors if there exists at least one clause C in $\mathcal{C}(F)$ s.t. both x and y occurring in C. We use $N(x, F)$ to denote the set of x's neighboring variables in F, i.e., $N(x, F) = \{y | x, y \text{ both occur in } C, C \in \mathcal{C}(F)\}$. Given a CNF formula F with a weighting function $\mathbb{W}_F : \mathcal{C}(F) \mapsto Z^+$, we say that $\langle F, \mathbb{W}_F \rangle$ is a MaxSAT formula (or we call it a MaxSAT instance). Without loss of generality for any unweighted soft clause $C^s \in \mathcal{C}(F)$, we let $\mathbb{W}_F(C^s) = 1$. We use $\mathbb{W}_F$ with subscripts here in order to distinguish between this weight notation and those below in graph theory.

Usually, SLS algorithms will first make a random guess to obtain a candidate solution, then they will change this solution by trial and error, so we introduce some related notions here. A complete assignment is a map $\alpha : \mathcal{V}(F) \mapsto \{0, 1\}$, which assigns a Boolean value (either 0 or 1) to each variable in the formula F, so for any variable x in F, either $\alpha(x) = 0$ or $\alpha(x) = 1$. In the context of SLS algorithms for MaxSAT, as well as its variants, a (candidate) solution is a complete assignment. In this sense, we say that x is flipped if we change the Boolean value of x from 0 to 1 or vice versa. More formally, this manipulation leads to another assignment $\alpha' = \alpha \backslash \{\langle x, \alpha(x) \rangle\} \cup \{\langle x, 1 - \alpha(x) \rangle\}$. In what follows, we will use the notions of assignment and solution interchangeably.

Given a CNF formula F and a complete assignment α that maps $\mathcal{V}(F)$ to $\{0, 1\}$, each clause in F under the assignment α has two possible states: satisfied and unsatisfied; a

clause C in F is satisfied if at least one literal in C takes the value 1 (*true*) under α; otherwise, C is unsatisfied.

Clauses in a (weighted) partial MaxSAT formula $\langle F, \mathbb{W}_F \rangle$ are partitioned into hard and soft ones, and each soft clause in the weighted case is further associated with a positive integer. Given $\langle F, \mathbb{W}_F \rangle$, the (weighted) partial maximum satisfiability ((W)PMS)) problem is to find a complete assignment that satisfies all hard clauses in F and minimizes the total weight/number of all unsatisfied soft clauses in F.

Given a (weighted) partial MaxSAT formula $\langle F, \mathbb{W}_F \rangle$, a complete assignment is feasible if it satisfies all hard clauses in F. The quality of a complete assignment α over $\langle F, \mathbb{W}_F \rangle$, denoted as $quality(\alpha, F, \mathbb{W}_F)$, is the total weight/number of all satisfied soft clauses in $\langle F, \mathbb{W}_F \rangle$ under α. An optimum assignment is a feasible assignment, namely α^*, s.t., for any feasible assignment α over $\langle F, \mathbb{W}_F \rangle$, $quality(\alpha^*, F, \mathbb{W}_F) \geq quality(\alpha, F, \mathbb{W}_F)$, that is an optimum assignment over $\langle F, \mathbb{W}_F \rangle$ is an feasible assignment with the minimum cost. In what follows, we usually suppress F and $\mathbb{W}_F$ in $quality(\alpha, F, \mathbb{W}_F)$ if understood from the context.

2.1. Variable Graph

The research community for complex networks has developed techniques of analysis and algorithms to study real-world graphs, and such approaches can be adopted by the SAT community. Inspired by the results on complex networks, Ref. [21] studied the community structure, or modularity, of industrial SAT instances, and they proposed a notion named the *variable graph*, which describes the interactions between Boolean variables in a SAT formula. Here, we extend the notion of the *variable graph* so that it works seamlessly in PMS and WPMS problems.

The variable graph of a (weighted) partial MaxSAT formula $\langle F, \mathbb{W}_F \rangle$, denoted by $G(F, \mathbb{W}_F)$, is defined as $(V_F, E_F, \mathcal{W}_{\langle F, \mathbb{W}_F \rangle})$, which describes the interactions between any pair of distinct Boolean variables in F. First, V_F is a vertex set s.t. each vertex $v_i \in V_F$ representing a Boolean variable $x_i \in \mathcal{V}(F)$, i.e., there is a bijection $\phi : \mathcal{V}(F) \mapsto V_F$ for graph construction. In this sense, the inverse function ϕ^{-1} exists, and it maps vertices, namely v_i, back to their corresponding Boolean variables, namely x_i. Second, they defined E_F as $\{\{u, v\} | x = \phi^{-1}(u), y = \phi^{-1}(v)$ and $y = N(x, F)\}$, i.e., two vertices in a variable graph $G(F, \mathbb{W}_F)$ are connected if and only if their corresponding Boolean variables are neighbors in F. Third, the edge weight component $\mathcal{W}_{\langle F, \mathbb{W}_F \rangle}$, is defined as below.

$$\mathcal{W}_{\langle F, \mathbb{W}_F \rangle}(\{u, v\}) = \sum_{x,y \in C} \frac{1}{\binom{|C|}{2}},$$

where $x = \phi^{-1}(u), y = \phi^{-1}(v)$. In this formula, $|C|$ is the cardinality of C and $\binom{|C|}{2}$ means a combination of $|C|$ elements taken two elements at a time. The motivation is to give the same relevance to all clauses, so they pondered the contribution of a clause to an edge by $1/\binom{|C|}{2}$. This way, the sum of the weights of the edges generated by a clause is always 1. In this paper, we propose an extension to this weighting scheme that is tailored for PMS and WPMS.

2.2. Local Search for MaxSAT

The basic framework of SLS algorithms for solving (W)PMS can be described as follows. Initially, an SLS algorithm randomly generates an assignment of Boolean values to all variables; then, it repeatedly selects and flips a Boolean variable until the cutoff arrives; finally, it returns the best feasible assignment that has been found. During the search, most SLS algorithms alternate between two modes: greedy (intensification) mode and random (diversification) mode. In greedy modes, SLS algorithms prefer those flips that lead to a decreasing number of unsatisfied hard clauses and a decreasing total weight/number of unsatisfied soft clauses. In random modes, they tend to diversify the search by randomized strategies.

2.3. Clause Penalties in SATLike 3.0

Although greedy search helps find better solutions nearby, it can often be trapped in local optima, so various diversification strategies have been proposed to tackle this problem including dynamic local search. Usually, in the context of SAT/MaxSAT, SLS algorithms associate each clause in a CNF formula F with a penalty, in order to help focus more on those clauses that are rarely satisfied [38]. To be specific, if a clause, namely C, is often unsatisfied, they will increase C's penalty often. As a result, any solution that violates C will tend to have a great penalty. Alternatively, the search will have high priority to satisfy C. In this sense, each clause will have opportunities to be satisfied.

Below, we introduce the penalty management scheme in *SATLike* 3.0 [37], which is also adopted for our algorithms. It distinguishes between hard and soft clauses with three parameters: the change δ_h for hard clause penalties, the change δ_s for soft clause penalties, as well as Λ, which limits soft clause penalties. *SATLike* 3.0 uses Λ to prevent the penalties of soft clauses from being too large, in case they receive too much attention. Furthermore, δ_h is usually greater than δ_s because hard clauses should have greater impacts on the search, compared to soft ones:

1. Initially:
   (a) $penalty(C^h) \leftarrow 1$ for each hard clause C^h;
   (b) $penalty(C^s) \leftarrow \mathbb{W}_F(C^s)$ for each soft clause C^s.
2. At each local optimum:
   (a) with probability p:
      i. $penalty(C^h) \leftarrow penalty(C^h) + \delta_h$ for each violated hard clause C^h;
      ii. $penalty(C^s) \leftarrow penalty(C^s) + \delta_s$ for each violated soft clause C^s s.t. $penalty(C^s) < \Lambda$.
   (b) with probability $1 - p$:
      i. $penalty(C^h) \leftarrow penalty(C^h) - \delta_h$ for each satisfied hard clause C^h s.t. $penalty(C^h) > \delta_h$;
      ii. $penalty(C^s) \leftarrow w(C^s) - \delta_s$ for each satisfied hard clause C^s s.t. $penalty(C^s) > \delta_s$.

Like most SLS algorithms, *SATLike* 3.0 uses the traditional notion of a *score* to select variables to flip, in order to decrease the total penalties. Before introducing this notion, we first introduce the *cost* of an assignment over a MaxSAT formula, which sums up the penalties of all hard and soft clauses. Given an assignment α, the *cost* of α over $\langle F, \mathbb{W}_F \rangle$, denoted by $cost(\alpha, F, \mathbb{W}_F)$, is defined as the total penalties of all unsatisfied clauses. In this sense, the *score* of x under α over $\langle F, \mathbb{W}_F \rangle$, denoted by $score(x, \alpha, F, \mathbb{W}_F)$, is defined as the benefits of flipping x in F. More specifically,

$$score(x, \alpha, F, \mathbb{W}_F) = cost(\alpha, F, \mathbb{W}_F) - cost(\alpha', F, \mathbb{W}_F),$$

where α' is the same as α with x being flipped. Therefore, the scoring function measures that decrease of penalties that is caused by flipping x. *SATLike*, as well as its variants mainly rely on this scoring function and guide local search to seek a better solution. Last, we use $age(x)$ to denote the number of flips since the last time x was flipped.

2.4. SATLike 3.0

Below, we introduce a state-of-the-art solver, *SATLike* 3.0 (See Algorithm 1) [37], which performed well in recent MaxSAT Evaluations. It is named after SAT because it works somewhat like a SAT solver. The experimental results showed that it outperforms some SAT-based solvers in some industrial benchmarks.

Algorithm 1: SATLike 3.0.

input : A (W)PMS formula $\langle F, \mathbb{W}_F \rangle$ and the *cutoff*
output: A best solution found α^* or "NO SOLUTION FOUND"

```
1  Initiate penalties as mentioned in Section 2.3;
2  α ← init(F); // employ unit propagation for initialization
3  α* ← α;
4  quality* ← 0;
5  while elapsed time < cutoff do
6      if α is feasible and quality(α) > quality* then
7          α* ← α;
8          quality* ← quality(α);
9      if α* satisfies F then return α*;
10     G ← {x ∈ V(F) | score(x, α, F, W_F) > 0};
11     if G ≠ ∅ then
12         v ← x ∈ G with the greatest score, breaking ties in favor of the oldest one;
13     else
14         adjust penalties as mentioned in Section 2.3;
15         C ← a random unsatisfied clause;
16         v ← x in C with the greatest score, breaking ties in favor of the oldest one;
17     α ← α with v being flipped;
18 if α* is feasible then return α*;
19 else return "NO SOLUTION FOUND";
```

3. Initiating a Solution Based on Variables' Structural Entropy

In local search, an initial solution (starting point), which lies near a high-quality one, may cost significantly less steps (flips) to achieve that satisfactory solution. In our previous works [39,40], we confirmed that a variable's structural entropy significantly influences the probability that it will be flipped later in local search. Based on this result, we developed strategies for initiating solutions and such strategies greatly improve two state-of-the-art solvers, *Sparrow* and *CCASat*. However, most SLS PMS and WPMS solvers initiate a solution in a purely random way, which possibly generates a bad starting point. Hence, in this work, we extend our initiating strategy for SAT to (W)PMS problems, in order to improve efficiency. In this section, we introduce (1) a novel weighting scheme for the variable graph of a MaxSAT formula, (2) our definition of structural entropy, and finally, (3) our algorithm for constructing a good starting point.

3.1. A Novel Weighting Scheme Tailored for (W)PMS

We adopted the notion of the variable graph from [21], but assigned each edge a positive weight in a novel approach. Before introducing this approach, we first define the *relevance* of any pair of distinct variables in a MaxSAT formula.

Definition 1. *Given a (weighted) partial MaxSAT formula $\langle F, \mathbb{W}_F \rangle$, a clause $C \in \mathcal{C}(F)$, and a pair of variables x, y, we define the* relevance *between x and y in C over $\langle F, \mathbb{W}_F \rangle$, denoted by $t_{\langle F, \mathbb{W}_F, C \rangle}(x, y)$, as below, where $\mathtt{W}$ is the total weight of all soft clauses.*

$$t_{\langle F, \mathbb{W}_F, C \rangle}(x, y) = \begin{cases} \dfrac{1}{\binom{|C|}{2}} & \text{if } x, y \in C \text{ and } C \in \mathcal{C}_h(F); \\ \dfrac{\mathbb{W}_F(C)}{\mathtt{W}} \cdot \dfrac{1}{\binom{|C|}{2}} & \text{if } x, y \in C \text{ and } C \in \mathcal{C}_s(F); \\ 0 & \text{otherwise.} \end{cases}$$

Now, we discuss some special cases of this formula to present some intuition:

1. If either x or y is absent from C, we think that there is no connection between them in C, so the relevance between them in C is defined to be 0.
2. If C is long, we believe that the connection between x and y is weak, so $\binom{|C|}{2}$ will be big and the relevance tends to be small. Cases are analogous if C is short.
3. If C is soft, we guess that their connection is weak, so the coefficient $\frac{\mathbb{W}_F(C)}{\mathtt{W}}$ helps decrease the relevance value.
4. If C is a soft clause with a great weight, we feel that the connection between x and y is relatively big, then $\frac{\mathbb{W}_F(C)}{\mathtt{W}}$ will be relatively big as well and so will be the relevance.

Then, we define the total relevance between x and y in a clause set S over $\langle F, \mathbb{W}_F \rangle$ as the sum of relevance over all clauses in S, as is shown below.

$$t_{\langle F, \mathbb{W}_F, S \rangle}(x, y) = \sum_{C \in S} t_{\langle F, \mathbb{W}_F, C \rangle}(x, y)$$

Here, we abuse the notation in Definition 1 and write $t_{\langle F, \mathbb{W}_F, S \rangle}(x, y)$ to discuss cases about clause sets. Therefore, the total relevance between x and y in the clause set of F, i.e., $\mathcal{C}(F)$, is $t_{\langle F, \mathbb{W}_F, \mathcal{C}(F) \rangle}(x, y)$, which measures how closely related the two variables are in the involved MaxSAT formula.

Finally, we are ready to define edge weights in $G(F, \mathbb{W}_F)$, i.e., $\mathcal{W}_{\langle F, \mathbb{W}_F \rangle}(u, v) = t_{\langle F, \mathbb{W}_F, \mathcal{C}(F) \rangle}(x, y)$, where $x = \phi^{-1}(u)$ and $y = \phi^{-1}(v)$. In this sense, the weight of an edge in our variable graph $G(F, \mathbb{W}_F)$ represents the relevance between their corresponding Boolean variables in the MaxSAT formula $\langle F, \mathbb{W}_F \rangle$.

3.2. Properties of Our Weighting Scheme

Now, we discuss the impacts of hard and soft clauses on the relevance between Boolean variables. First, we have a proposition below that shows that the contribution of a *single* hard binary clause to the relevance is no *smaller* than that made by all soft clauses.

Proposition 1. *Given a MaxSAT formula $\langle F, \mathbb{W}_F \rangle$, if there exists a binary hard clause C^h that contains variables x and y, then*

$$t_{\langle F, \mathbb{W}_F, \mathcal{C}_s(F) \rangle}(x, y) \leq t_{\langle F, \mathbb{W}_F, C^h \rangle}(x, y);$$

the equality relation holds if and only if all soft clauses are of length 2.

Proof. First, we amplify the left-hand side as below.

$$\begin{aligned}
& t_{\langle F, \mathbb{W}_F, \mathcal{C}_s(F) \rangle}(x, y) \\
= & \sum_{C \in \mathcal{C}_s(F)} t_{\langle F, \mathbb{W}_F, C \rangle}(x, y) \\
= & \sum_{C \in \mathcal{C}_s(F), |C| \geq 2} \frac{\mathbb{W}_F(C)}{\mathtt{W}} \cdot \frac{1}{\binom{|C|}{2}} \\
\leq & \sum_{C \in \mathcal{C}_s(F), |C| \geq 2} \frac{\mathbb{W}_F(C)}{\mathtt{W}} \cdot \frac{1}{\binom{|C^h|}{2}} \qquad (1) \\
= & \frac{1}{\binom{|C^h|}{2}} \cdot \sum_{C \in \mathcal{C}_s(F), |C| \geq 2} \frac{\mathbb{W}_F(C)}{\mathtt{W}} \\
\leq & \frac{1}{\binom{|C^h|}{2}} = t_{\langle F, \mathbb{W}_F, C^h \rangle}(x, y). \qquad (2)
\end{aligned}$$

1. As to (1), if there exists any soft clause whose length is greater than 2, then the equation there does not hold.

2. As to (2), if there exists any soft clause whose length is smaller than 2, then the equation there does not hold.
3. Obviously, if all soft clauses are of length 2, the equality relation in the proposition above holds.

According to the statements above, we have proven this proposition. □

Based on this proof, we have a corollary below.

Corollary 1. *Given a MaxSAT formula $\langle F, \mathbb{W}_F \rangle$ and an integer k s.t. $k \geq 2$, if:*

1. *there exists a hard clause C^h of length k that contains variables x and y;*
2. *all soft clauses are at least of length k.*

then

$$t_{\langle F, \mathbb{W}_F, \mathcal{C}_s(F) \rangle}(x, y) \leq t_{\langle F, \mathbb{W}_F, C^h \rangle}(x, y);$$

the equality relation holds if and only if all soft clauses are of length k.

3.3. Variables' Structural Entropies

Given an edge-weighted graph $G = (V, E, w_G)$, we use $N(u, G)$ to denote the set of u's neighbor in G and use $d(u, G)$ to denote the cardinality of $N(u, G)$, i.e., $d(u, G) = |N(u, G)|$. Moreover, we use $\omega(u, G)$ to denote u's weighted degree, i.e., $\omega(u, G) = \sum_{v \in N(u,G)} w_G(\{u, v\})$, suppressing G if understood from the context. Given $U \subseteq V$, we define the volume of U, denoted by $vol(U)$, as $\sum_{u \in U} \omega(u)$, and we abuse this notation to define $vol(G)$ as $vol(V)$.

Definition 2. *Given a (weighted) partial MaxSAT formula $\langle F, \mathbb{W}_F \rangle$ and its variable graph $G(F, \mathbb{W}_F) = (V_F, E_F, \mathcal{W}_{\langle F, \mathbb{W}_F \rangle})$, where $V_F = \{v_1, \cdots, v_n\}$, we define v_i's structural entropy as*

$$\mathcal{H}(v_i) = -p_i \log_2 p_i = -\frac{\omega(v_i)}{vol(G)} \log_2 \frac{\omega(v_i)}{vol(G)},$$

then the structural entropy of the variable graph G is defined as

$$\mathcal{H}(G) = \sum_{i=1}^{n} \mathcal{H}(v_i).$$

As is stated in [41], the structural information $\mathcal{H}(G)$ of a weighted and connected graph G measures the information required to determine the code of the vertices that are accessible from a random walk in G with its stationary distribution $(p_1, \cdots, p_n)$. On the other hand, as to a single vertex, namely v_i, $\mathcal{H}(v_i)$ represents the uncertainty information of a random walk with a stationary distribution to visiting v_i from its neighbors.

Then, given a MaxSAT formula $\langle F, \mathbb{W}_F \rangle$ and a variable, namely x, we abuse the notation above to define x's structural entropy in $\langle F, \mathbb{W}_F \rangle$ as the structural entropy of its corresponding vertex in $G(F, \mathbb{W}_F)$.

Now, we present some properties of our definition of structural entropy to help understand its insights intuitively.

Proposition 2. *Let $f(x) = -x \log_2 x$ with $x \in (0, 1)$; we have:*

1. *$f(x) > 0$ for any $x \in (0, 1)$;*
2. *$f(x)$ is strictly monotonically increasing (respectively decreasing) in $(0, 1/e)$ (respectively $(1/e, 0)$), where e is Euler's constant and $e \approx 2.71828 \cdots$.*

Therefore, given a vertex, namely v_i, with $\mathcal{H}(v_i) = -p_i \log_2 p_i$, if $\mathcal{H}(v_i)$ is relatively small, then p_i is relatively close to 0 or 1. Similarly, if $\mathcal{H}(v_i)$ is relatively large, then the value of p_i is near $1/e$. In our algorithm, we first assign Boolean variables whose corresponding vertex has relatively small structural entropy. Now, we explain the motivation as follows.

Vertices with relatively small structural entropy correspond to Boolean variables that are of *much* or *little* influence on other variables in the CNF formula. Below, we discuss these cases in details:

1. Assigning highly influential variables tend to satisfy relatively many clauses or help satisfy clauses with great weights.
2. Variables of little influence often occur in few clauses or in clauses of small weights, so we simply assign them to satisfy such clauses.

3.4. Initiating Solutions

In this subsection, given a CNF formula F, we use $\mathcal{C}(F, x)$ to denote the set of clauses in F (including both hard and soft ones) that contain x as one of its literals. Similarly, we define the notation of $\mathcal{C}(F, \neg x)$. Then, our procedure for initiating a solution is described in Algorithm 2, which is named variables' structural entropy-based initialization (VSEI). The motivation of VSEI is as follows. The smaller a variable's structural entropy is, the more stable its truth value is, hence the smaller the probability that it will be flipped later [40]. That is, a variable with smaller structural entropy should probably be assigned earlier, compared to those variables with greater ones.

The main idea of Algorithm 2 is as follows. When initiating a solution, we repeated the following operations: picking a variable that is unassigned with the smallest structural entropy and, then, mapping it to 0 or 1, depending on the number of its positive and negative occurrences in clauses that have not been satisfied yet. To be specific, suppose we have picked a variable x and x's positive occurrences is more than its negative ones, then we assign 1 to x; otherwise, we assign 0 to x. In a nutshell, we assign values to variables greedily in order to satisfy as many clauses as possible at the end of initialization.

Algorithm 2: VSEI.

input : A (W)PMS formula $\langle F, \mathbb{W}_F \rangle$
output: An initial solution

```
1  NonAssignedSet ← V(F); // the set of unassigned variables
2  NonSatClSet ← C(F); // the set of clauses not satisfied yet
3  while NonAssignedSet ≠ ∅ do
4      x ← a variable in NonAssignedSet with the smallest structural entropy,
        breaking ties randomly;
5      if |NonSatClSet ∩ C(F, x)| > |NonSatClSet ∩ C(F, ¬x)| then
6          α(x) ← 1;
7          NonSATClSet ← NonSATClSet\C(F, x);
8      else
9          α(x) ← 0;
10         NonSATClSet ← NonSATClSet\C(F, ¬x);
11     NonAssignedSet ← NonAssignedSet\{x};
12 return α;
```

4. Diversifying Variable Selection Based on Clause Penalties

Each time the search encounters a local optimum, i.e., there are no variables whose flip leads to a penalty decrease, we will call Algorithm 3 to pick a variable and flip it. More specifically, this algorithm first adjusts penalties like *SATLike* 3.0 (Line 1), then it picks a variable in two possible ways depending on a parameter p: (1) continuing to choose one with the best score and the best age (Line 3); (2) focusing on an unsatisfied clause with the greatest penalty (Line 5) and performing probabilistic selections on it (Line 6). Considering its most distinguishing features, we name it probabilistic selection for great penalties (PSGP).

Algorithm 3: PSGP.

input : $score(x)$ and $age(x)$ for all x's in F, the current unsatisfied clause set U
output: A variable to be flipped

1 adjust penalties as mentioned in Section 2.3;
2 **begin** with probability p,
3 $v \leftarrow$ a variable in F with the greatest score, breaking ties in favor of the oldest one;
4 **begin** with probability $1-p$,
5 $C \leftarrow$ a clause in U with the greatest penalty, breaking ties randomly;
6 $v \leftarrow$ a variable x in C with a probability proportional to $\delta^{-break(x,\alpha)}$;
7 **return** v;

In Line 3, we insist on picking the globally best variables, and this may bring some benefits. The reason is that Line 1 just changes the penalties of some clauses, which, in turn, affects the score of some variables involved.

In Line 6, each variable namely x in C is picked with a probability proportional to $\delta^{-break(x,\alpha)}$, where $break(x,\alpha)$ is the number of clauses (including both hard and soft ones) that will become unsatisfied if x is flipped, given the current assignment α. Hence, this probability distribution always prefers small-*break* variables. Here, the parameter δ controls how concentrated this distribution is at small-*break* variables. Obviously, the greater δ is, the greater the probability difference between small-*break* and big-*break* variables. This distribution is inherited from *ProbSAT* [42], which was a simple and elegant local search SAT solver with a probabilistic selection as its single strategy.

5. ImSATLike

In this section, we introduce our novel Algorithm 4 as a whole, which works on (W)PMS instances. In the initialization procedure, it adopts variables' structural entropy to generate a good starting point of local search. Then, during local search, each time it meets a local optimum, it will still pick the globally best variables or it will focus on an unsatisfied clause with the greatest penalty and choose variables by probabilistic selection. Since our algorithm is based on *SATLike* 3.0, we call it *ImSATLike*.

There are two main differences between our algorithm and *SATLike* 3.0: (1) *SATLike* 3.0 employs unit propagation to generate an initial solution, while our solver initiates starting points by variables' structural entropy; (2) in diversification, *SATLike* 3.0 picks a random unsatisfied clause and performs greedy selection, while our solver still possibly continues our greedy strategy or focuses on a clause with the greatest penalty and exploits probabilistic selection.

Algorithm 4: ImSATLike.

input : A (W)PMS formula $\langle F, \mathbb{W}_F \rangle$ and the *cutoff*
output: A best solution found α^* or "NO SOLUTION FOUND"

Initiate penalties as mentioned in Section 2.3
$\alpha \leftarrow$ an assignment generated by VSEI (Algorithm 2);
$\alpha^* \leftarrow \alpha$;
$quality^* \leftarrow 0$;
while *elapsed time* $<$ *cutoff* **do**
  if α *is feasible and* $quality(\alpha) > quality(\alpha^*)$ **then**
    $\alpha^* \leftarrow \alpha$;
    $quality^* \leftarrow quality(\alpha)$;
  if α^* *satisfies* F **then return** α^*;
  $G \leftarrow \{x | score(x, \alpha, F, \mathbb{W}_F) > 0\}$;
  if $G \neq \emptyset$ **then**
    $v \leftarrow x \in G$ with the greatest *score*, breaking ties in favor of the oldest one;
  else
    $v \leftarrow$ a variable returned by PSGP (Algorithm 3);
  $\alpha \leftarrow \alpha$ with v being flipped;
if α^* *is feasible* **then return** α^*;
else return "NO SOLUTION FOUND";

6. Experimental Evaluations

To evaluate the performance of our algorithm, we compared it to *SATLike* and its improvement *SATLike* 3.0 on (W)PMS instances, which was used in MaxSAT Evaluations 2018 and 2019. To be specific, these instances came from four benchmarks, namely `ms18_wt`, `ms19_wt`, `ms18_unwt`, and `ms19_unwt`, among which the former (respectively latter) two contain all weighted (respectively unweighted) partial MaxSAT instances used in 2018 and 2019, respectively. For each instance, namely $\mathcal{I}$, among $\mathcal{I}$'s feasible solutions, there is a quality that is known to be the best, and we call it $\mathcal{I}$'s best-known (solution) quality. Given a solver $\mathcal{A}$ and an instance $\mathcal{I}$, we say that $\mathcal{A}$ *successfully* solves $\mathcal{I}$ in a particular run if $\mathcal{A}$ locates a solution of that best-known quality in that run.

SATLike 3.0 not only outperforms *CCEHC* and *Dist*, but also beats their respective improvements, *DeciDist* and *DeciCCEHC*, which alternates between decimation and local search [43]. Hence, it is the current best local search solver for (W)PMS problems. In addition, we also compared our solver to two SAT-based ones, Open-WBO-inc [30] and LinSBPS (https://helda.helsinki.fi/bitstream/handle/10138/237139/mse18_proceedings.pdf?sequence=1, accessed on 8 November 2022), which were the top two solvers in MaxSAT Evaluation 2018.

SATLike 3.0 was downloaded from the web pages of MaxSAT Evaluation 2018 (https://maxsat-evaluations.github.io/2018/, accessed on 8 November 2022), and we adopted its default parameter settings in the following experiments. Based on this version, we developed *ImSATLike* with two extra parameters: p and δ in Algorithm 3, which were set to 0.3 and 2.06, respectively.

In the following tables, we use *ins_class* to denote instance sets, #win to denote the number of instances that were successfully solved, #ins to denote the number of instances in each instance set, and *time* to denote the average running time to locate a solution. The best #win and *time* values are shown in **bold** font.

6.1. Comparing ImSATLike to Other SLS Solvers

We compared *ImSATLike* with *SATLike*, as well as its 3.0 version on a computer equipped with an Intel(R) Core(TM) i5-10210U CPU @ 1.60 GHz 2.11 GHz with 8 GB

RAM, running the Windows 10 OS. First, we conducted experiments with 60 s as a cutoff, then we repeated such experiments with 300 s as a second cutoff (see Table 1).

Table 1. Comparative results of *ImSATLike* and *SATLike* with its 3.0 version.

ins_class	#ins	SATLike		SATLike 3.0		ImSATLike	
		#win	#time	#win	#time	#win	#time
60 s							
ms18_wt	172	101	23.2335	99	13.8662	**103**	17.4195
ms19_wt	282	154	23.4689	152	24.5351	**159**	19.5359
ms18_unwt	153	**77**	52.1586	73	50.2791	**77**	**49.0867**
ms19_unwt	288	158	14.0828	149	15.9554	**163**	16.2713
300 s							
ms18_wt	172	**110**	103.554	102	99.4858	106	76.8005
ms19_wt	282	**162**	119.761	160	91.7588	**162**	**79.0564**
ms18_unwt	153	77	88.226	73	115.1743	**86**	137.7936
ms19_unwt	288	166	68.926	152	77.1327	**168**	71.8434

From Table 1, we found the following. Within 60 s:

1. *ImSATLike* outperformed both *SATLike* and its 3.0 version in terms of #win on each of the four benchmark categories, with the exception of ms18_unwt, where *ImSATLike* and *SATLike* came to a draw;
2. In this benchmark category, *ImSATLike* generally located its solutions within a shorter time, compared to that spent by *SATLike*.

Within 300 s:

1. *ImSATLike* showed best performances in terms of #win in 3 categories, while *SATLike* did that in 2;
2. In ms19_wt, where both *ImSATLike* and *SATLike* achieved 162 in terms of #win, *ImSATLike* generally located its solutions within a much shorter time, compared to that spent by *SATLike*.

6.2. Individual Impacts of Our Strategies

To evaluate the individual impacts, we modified *SATLike* 3.0 and developed two *independent* variants. Then, we reperformed the experiments above with 60 s as the cutoff and compared these variants in terms of #win.

1. First, we replaced the initialization procedure in *SATLike* 3.0 with our VESI strategy and developed a solver named *SATLike_a1*.
2. Second, we replaced *SATLike* 3.0's diversification mode with our PSGP strategy and developed a second solver named *SATLike_a2*.

From Table 2, we found the following:

1. In none of the benchmark categories, *SATLike* 3.0 outperformed either *SATLike_a1* or *SATLike_a2* in terms of #win, which illustrates the robustness of our strategies.
2. In all of these categories, *SATLike_a2* significantly outperformed *SATLike* 3.0, which showed the power of our PSGP strategy.
3. In half of these categories, *SATLike_a1* was superior to *SATLike* 3.0, which presented the positive impacts of our VESI strategy.

Table 2. Individual impacts of the VSEI and PSGP strategies.

ins_class	#ins	SATLike_a1		SATLike 3.0		SATLike_a2	
		#win	#time	#win	#time	#win	#time
ms18_wt	172	99	14.603	99	13.8662	**103**	15.9523
ms19_wt	282	155	16.345	152	24.5351	**158**	26.3373
ms18_unwt	153	73	52.888	73	50.2791	**77**	52.7514
ms19_unwt	288	150	16.107	149	15.9554	**155**	16.3538

6.3. Comparing ImSATLike to SAT-Based Solvers

We compared *ImSATLike* with two SAT-based solvers, *Open-WBO-Inc* and *LinSBPS*, on a computer equipped with an Intel Core i5-10210U CPU @ 1.60 GHz × 8 with 8 GB RAM, running Ubuntu 18.04.5 LTS. These two competitors were the top two solvers in the incomplete track in MaxSAT Evaluation 2018, where the time limit was 60 s, and their codes were downloaded from the web pages of MaxSAT Evaluation (https://maxsat-evaluations.github.io/2018/, accessed on 8 November 2022). To be consistent with MaxSAT Evaluation 2018, we also set the cutoff here to be 60 s. From Table 3, we found that our solver performed somewhat close to the top SAT-based solvers on weighted partial MaxSAT instances, although it fell behind in general.

Table 3. Comparative results of our algorithm and two SAT-based solvers.

ins_class	#ins	ImSATLike	LinSBPS	Open-WBO-Inc
ms18_wt	172	130	164	164
ms19_wt	282	185	269	266
ms18_unwt	153	78	135	134
ms19_unwt	288	174	270	260

6.4. Evaluations of a Hybrid Solver Incorporating ImSATLike

Combining solvers in different frameworks has proven to be a promising approach, which has been confirmed in recent MaxSAT Evaluations. Therefore, we combined our solver *ImSATLike* with a state-of-the-art SAT-based solver, *TT-Open-WBO-inc* [31], which was the champion in the incomplete track of MaxSAT Evaluation 2019. We call this hybrid solver *ImSATLike-TT*, and its work flow based on a MaxSAT formula is as follows:

1. A SAT solver $\mathcal{B}$ is called to find a feasible solution α_{init}.
2. $\mathcal{B}$ passes α_{init} to *ImSATLike* as its starting point.
3. *ImSATLike* bypasses its VSEI strategy and performs local search for k steps, where k was set to be 10^7.
4. *ImSATLike* passes its best-found solution to *TT-Open-WBO-inc*.
5. *TT-Open-WBO-inc* is run for the remaining time.

In most cases, *ImSATLike* was able to find high-quality solutions, but the time for it to find even better solutions will increase exponentially, so in this situation, *ImSATLike-TT* will turn to *TT-Open-WBO-inc* for better solutions.

For better comparisons, we also included *TT-Open-WBO-inc* and *SATLike*-ck as competitors. Note that *SATLike*-ck is the same as *ImSATLike-TT*, but employs *SATLike* as its embedded local search component. The experiments were conducted on a computer equipped with an Intel Core i5-10210U CPU @ 1.60 GHz × 8 with 8 GB RAM running Ubuntu 18.04.5 LTS, and the cutoff was set to 60 s. The experimental outcome can be found in Table 4, which shows the number of successfully solved instances for each SAT-based solver and each portfolio in each benchmark category.

Table 4. Comparative results of SAT-based solvers and two portfolios.

ins_class	#ins	ImSATLike	LinSBPS	Open-wbo	TT-Open-WBO-inc	SATLike-ck	ImSATLike-TT
ms18_wt	172	130	**164**	**164**	161	158	158
ms19_wt	282	185	**269**	266	265	261	263
ms18_unwt	153	78	135	134	135	**144**	**144**
ms19_unwt	288	174	270	260	263	**277**	**277**

In Table 4, we find the following:

1. On partial MaxSAT instances, *ImSATLike-TT* performed the same as *SATLike*-ck, and they were the top two solvers, which showed the superiority of portfolios over SAT-based solvers.
2. On weighted partial MaxSAT instances, *ImSATLike-TT* performed as well as *SATLike*-ck in ms18_wt, but outperformed it in ms19_wt, which showed the positive effects of our strategies.

7. Conclusions and the Future Work

In this paper, we presented a local search MaxSAT solver named *ImSATLike*, as well as a hybrid solver named *ImSATLike-TT*, which performed better than or the same as state-of-the-art competitors *SATLike* 3.0 and *SATLike*-ck, respectively, on (weighted) partial MaxSAT instances in recent MaxSAT Evaluations.

The main contributions include: (1) an initialization strategy to help generate a solution that is closer to high-quality feasible ones; (2) a diversification strategy to guide local search to a more promising area.

As for future works, it will be interesting to apply these strategies to solve other combinatorial problems such as the vertex cover and dominating set problems.

Author Contributions: Conceptualization, Z.Z., Y.F. and J.Z.; methodology, Z.Z., Y.F., H.Y. and J.Z.; software, Z.Z. and Y.F.; validation, Z.Z. and J.Z.; formal analysis, Z.Z. and Y.F.; investigation, J.Z.; resources, X.W.; data curation, H.Y.; writing—original draft preparation, Z.Z., Y.F., X.W. and H.Y.; writing—review and editing, Z.Z., Y.F. and J.Z.; visualization, X.W.; supervision, J.Z.; project administration, J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China grants (61862051, 62241206, 61762019), the Science and Technology Plan Project of Guizhou Province grants (No. Qiankehe Foundation-ZK[2022] General 550 and [2019]1299), the Top-notch Talent Program of Guizhou province under Grant No. KY[2018]080, the Educational Department of Guizhou under Grant No. KY[2019]067, the Industrial Technology Foundation of Qiannan State of China grants (2019XK02ST, 2019XK01ST, 2020XK05ST), the Special Foundation for Talents in Qiannan Normal University for Nationalities in 2019, the Special project for high-level talents of Qiannan Normal University for Nationalities grants (Nos. qnsy202203, qnsy202204), the Project for Growing Youth Talents of the Educational Department of Guizhou (No. KY[2019]201), and the Program of Qiannan Normal University for Nationalities (Nos. QNSY2018JS013, QNSYRC201715, qnsy2018015).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The benchmarks used in this paper are available from https://maxsat-evaluations.github.io/2018/ and https://maxsat-evaluations.github.io/2019/ (accessed on 8 November 2022).

Acknowledgments: We would like to thank the anonymous Referees for their helpful comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Purdom, P.W. Solving Satisfiability with Less Searching. *IEEE Trans. Pattern Anal. Mach. Intell.* **1984**, *PAMI-6*, 510–513. [CrossRef]
2. Smyth, K.; Hoos, H.H.; Stützle, T. Iterated Robust Tabu Search for MAX-SAT. In Proceedings of the Advances in Artificial Intelligence, 16th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2003, Halifax, NS, Canada, 11–13 June 2003; Xiang, Y., Chaib-draa, B., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2671, pp. 129–144. [CrossRef]
3. Cai, S.; Luo, C.; Thornton, J.; Su, K. Tailoring Local Search for Partial MaxSAT. In Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014; Brodley, C.E., Stone, P., Eds.; AAAI Press: Washington, DC, USA, 2014; pp. 2623–2629.
4. Jiang, Y.; Kautz, H.A.; Selman, B. Solving Problems with Hard and Soft Constraints Using a Stochastic Algorithm for MAX-SAT. In Proceedings of the 1st International Joint Workshop on Artificial Intelligence and Operations Research, Timberline, OR, USA, 6–10 June 1995.
5. Thornton, J.; Sattar, A. Dynamic Constraint Weighting for Over-Constrained Problems. In Proceedings of the PRICAI'98, Topics in Artificial Intelligence, 5th Pacific Rim International Conference on Artificial Intelligence, Singapore, 22–27 November 1998; Lee, H., Motoda, H., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1531, pp. 377–388. [CrossRef]
6. Cha, B.; Iwama, K.; Kambayashi, Y.; Miyazaki, S. 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, Providence, RI, USA, 27–31 July 1997; Kuipers, B., Webber, B.L., Eds.; AAAI Press: Washington, DC, USA; The MIT Press: Cambridge, MA, USA, 1997; pp. 263–268.
7. Fu, Z.; Malik, S. On Solving the Partial MAX-SAT Problem. In Proceedings of the Theory and Applications of Satisfiability Testing—SAT 2006, Seattle, WA, USA, 12–15 August 2006; Biere, A., Gomes, C.P., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 252–265.
8. Allouche, D.; Traoré, S.; André, I.; de Givry, S.; Katsirelos, G.; Barbe, S.; Schiex, T. Computational Protein Design as a Cost Function Network Optimization Problem. In Proceedings of the Principles and Practice of Constraint Programming—18th International Conference, CP 2012, Québec City, QC, Canada, 8–12 October 2012; Milano, M., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7514, pp. 840–849. [CrossRef]
9. Allouche, D.; André, I.; Barbe, S.; Davies, J.; de Givry, S.; Katsirelos, G.; O'Sullivan, B.; Prestwich, S.D.; Schiex, T.; Traoré, S. Computational protein design as an optimization problem. *Artif. Intell.* **2014**, *212*, 59–79. [CrossRef]
10. Naji-Azimi, Z.; Toth, P.; Galli, L. An electromagnetism metaheuristic for the unicost set covering problem. *Eur. J. Oper. Res.* **2010**, *205*, 290–300. [CrossRef]
11. Liao, X.; Koshimura, M.; Fujita, H.; Hasegawa, R. Solving the Coalition Structure Generation Problem with MaxSAT. In Proceedings of the 2012 IEEE 24th International Conference on Tools with Artificial Intelligence, Athens, Greece, 7–9 November 2012; Volume 1, pp. 910–915. [CrossRef]
12. Lai, Y.; Xu, Y.; Mai, D.; Fan, Y.; Yang, F. Optimized Large-Scale Road Sensing Through Crowdsourced Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 3878–3889. [CrossRef]
13. Chieu, H.L.; Lee, W.S. Relaxed Survey Propagation for The Weighted Maximum Satisfiability Problem. *J. Artif. Intell. Res.* **2009**, *36*, 229–266. [CrossRef]
14. Lin, H.; Su, K. Exploiting Inference Rules to Compute Lower Bounds for MAX-SAT Solving. In Proceedings of the IJCAI 2007, the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, 6–12 January 2007; pp. 2334–2339.
15. Heras, F.; Larrosa, J.; Oliveras, A. MiniMaxSAT: An Efficient Weighted Max-SAT solver. *J. Artif. Intell. Res.* **2008**, *31*, 1–32. [CrossRef]
16. Lin, H.; Su, K.; Li, C.M. Within-problem Learning for Efficient Lower Bound Computation in Max-SAT Solving. In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, IL, USA, 13–17 July 2008; Fox, D., Gomes, C.P., Eds.; AAAI Press: Washington, DC, USA, 2008; pp. 351–356.
17. Li, C.M.; Manyà, F.; Mohamedou, N.O.; Planes, J. Exploiting Cycle Structures in Max-SAT. In Proceedings of the Theory and Applications of Satisfiability Testing—SAT 2009, 12th International Conference, Swansea, UK, 30 June–3 July 2009; Kullmann, O., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5584, pp. 467–480. [CrossRef]
18. Davis, M.; Putnam, H. A Computing Procedure for Quantification Theory. *J. ACM* **1960**, *7*, 201–215. [CrossRef]
19. Davis, M.; Logemann, G.; Loveland, D. A Machine Program for Theorem-Proving. *Commun. ACM* **1962**, *5*, 394–397. [CrossRef]
20. Fu, Z. Extending the Power of Boolean Satisfiability Solvers: Techniques and Applications. Ph.D. Thesis, Princeton University, Princeton, NJ, USA, 2007.
21. Ansótegui, C.; Bonet, M.L.; Levy, J. Solving (Weighted) Partial MaxSAT through Satisfiability Testing. In Proceedings of the Theory and Applications of Satisfiability Testing—SAT 2009, 12th International Conference, Swansea, UK, 30 June–3 July 2009; Kullmann, O., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5584, pp. 427–440. [CrossRef]
22. Ansótegui, C.; Bonet, M.L.; Levy, J. On Solving MaxSAT Through SAT. In Proceedings of the POS-10, Pragmatics of SAT, Edinburgh, UK, 10 July 2010; Berre, D.L., Ed.; EPiC Series in Computing; EasyChair: Stockport, UK, 2010; Volume 8, pp. 41–48. [CrossRef]

23. Ansótegui, C.; Bonet, M.L.; Levy, J. A New Algorithm for Weighted Partial MaxSAT. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, GA, USA, 11–15 July 2010; Fox, M., Poole, D., Eds.; AAAI Press: Washington, DC, USA, 2010.
24. Ansótegui, C.; Bonet, M.L.; Gabàs, J.; Levy, J. Improving WPM2 for (Weighted) Partial MaxSAT. In Proceedings of the International Conference on Principles and Practice of Constraint Programming, Uppsala, Sweden, 16–20 September 2013; Schulte, C., Ed.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 117–132.
25. Ansótegui, C.; Bonet, M.L.; Levy, J. SAT-based MaxSAT algorithms. *Artif. Intell.* **2013**, *196*, 77–105. [CrossRef]
26. Narodytska, N.; Bacchus, F. *Maximum Satisfiability Using Core-Guided MAXSAT Resolution*; AAAI Press: Washington, DC, USA, 2014; pp. 2717–2723.
27. Ansótegui, C.; Gabàs, J. WPM3: An (in)complete algorithm for weighted partial MaxSAT. *Artif. Intell.* **2017**, *250*, 37–57. [CrossRef]
28. Marques Silva, J.; Sakallah, K. GRASP—A new search algorithm for satisfiability. In Proceedings of the International Conference on Computer Aided Design, San Jose, CA, USA, 10–14 November 1996; pp. 220–227. [CrossRef]
29. Marques-Silva, J.; Sakallah, K. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. Comput.* **1999**, *48*, 506–521. [CrossRef]
30. Martins, R.; Manquinho, V.M.; Lynce, I. Open-WBO: A Modular MaxSAT Solver. In Proceedings of the Theory and Applications of Satisfiability Testing—SAT 2014—17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, 14–17 July 2014; Sinz, C., Egly, U., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8561, pp. 438–445. [CrossRef]
31. Nadel, A. Anytime Weighted MaxSAT with Improved Polarity Selection and Bit-Vector Optimization. In Proceedings of the 2019 Formal Methods in Computer Aided Design, FMCAD 2019, San Jose, CA, USA, 22–25 October 2019; Barrett, C.W., Yang, J., Eds.; IEEE: Piscataway, NJ, USA, 2019; pp. 193–202. [CrossRef]
32. Hoos, H.; Sttzle, T. *Stochastic Local Search: Foundations & Applications*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2004.
33. Selman, B.; Levesque, H.J.; Mitchell, D.G. A New Method for Solving Hard Satisfiability Problems. In Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, USA, 12–16 July 1992; Swartout, W.R., Ed.; AAAI Press: Washington, DC, USA; The MIT Press: Cambridge, MA, USA, 1992; pp. 440–446.
34. Selman, B.; Kautz, H.A.; Cohen, B. Noise Strategies for Improving Local Search. In Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, 31 July–4 August 1994; Hayes-Roth, B., Korf, R.E., Eds.; AAAI Press: Washington, DC, USA; The MIT Press: Cambridge, MA, USA, 1994; Volume 1, pp. 337–343.
35. Cai, S.; Luo, C.; Lin, J.; Su, K. New local search methods for partial MaxSAT. *Artif. Intell.* **2016**, *240*, 1–18. [CrossRef]
36. Luo, C.; Cai, S.; Su, K.; Huang, W. CCEHC: An efficient local search algorithm for weighted partial maximum satisfiability. *Artif. Intell.* **2017**, *243*, 26–44. [CrossRef]
37. Cai, S.; Lei, Z. Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability. *Artif. Intell.* **2020**, *287*, 103354. [CrossRef]
38. Lei, Z.; Cai, S. Solving (Weighted) Partial MaxSAT by Dynamic Local Search for SAT. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, 13–19 July 2018; pp. 1346–1352. [CrossRef]
39. Zhang, Z.; Xu, D.; Zhou, J. An algorithm for solving satisfiability problem based on the structural information of formulas. *Front. Comput. Sci.* **2021**, *15*, 156405. [CrossRef]
40. Zhang, Z.; Xu, D.; Zhou, J. A Structural Entropy Measurement Principle of Propositional Formulas in Conjunctive Normal Form. *Entropy* **2021**, *23*, 303. [CrossRef] [PubMed]
41. Li, A.; Pan, Y. Structural Information and Dynamical Complexity of Networks. *IEEE Trans. Inf. Theory* **2016**, *62*, 3290–3339. [CrossRef]
42. Balint, A.; Schöning, U. Choosing Probability Distributions for Stochastic Local Search and the Role of Make versus Break. In Proceedings of the Theory and Applications of Satisfiability Testing—SAT 2012—15th International Conference, Trento, Italy, 17–20 June 2012; Cimatti, A., Sebastiani, R., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7317, pp. 16–29. [CrossRef]
43. Cai, S.; Luo, C.; Zhang, H. From Decimation to Local Search and Back: A New Approach to MaxSAT. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, 19–25 August 2017; pp. 571–577. [CrossRef]