SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2025, Vol. 68, Iss. 2, 122101:1–122101:15
https://doi.org/10.1007/s11432-023-3900-7

# Optimizing local search-based partial MaxSAT solving via initial assignment prediction

Chanjuan LIU[1], Guangyuan LIU[1], Chuan LUO[2*], Shaowei CAI[3], Zhendong LEI[3], Wenjie ZHANG[4], Yi CHU[3] & Guojing ZHANG[1]

[1]*School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China*
[2]*School of Software, Beihang University, Beijing 100191, China*
[3]*Institute of Software, Chinese Academy of Sciences, Beijing 100190, China*
[4]*School of Computer Science, Peking University, Beijing 100871, China*

Received 2 March 2023/Revised 31 August 2023/Accepted 17 November 2023/Published online 21 November 2024

**Abstract** Partial maximum satisfiability (PMS) is a significant generalization of Boolean satisfiability (SAT) and maximum satisfiability (MaxSAT), by introducing hard clauses and soft clauses. Compared with SAT and MaxSAT, the PMS problem has more real-world applications where both hard and soft constraints are involved. Local search is an effective incomplete method for solving PMS and is useful for important domains where good-quality solutions are desired within reasonable time. For local search PMS solvers, the approach for initial assignment generation is crucial because its effectiveness significantly affects practical performance. In this study, we propose a novel initial assignment prediction approach, called InitPMS. When predicting an assignment for PMS, InitPMS considers the specific structure of PMS instances, i.e., distinguishing hard and soft clauses. Our experiments on extensive PMS instances from MaxSAT evaluations (MSEs) 2020 and 2021 show that InitPMS significantly boosts the performance of five state-of-the-art local search PMS solvers, demonstrating its generality. In addition, our results indicate that incorporating InitPMS could improve the performance of one of the best incomplete PMS solvers in MaxSAT Evaluation 2021, indicating that InitPMS might help advance the state of the art in PMS solving.

**Keywords** MaxSAT, local search, initial assignment, incomplete method, assignment prediction

**Citation** Liu C J, Liu G Y, Luo C, et al. Optimizing local search-based partial MaxSAT solving via initial assignment prediction. Sci China Inf Sci, 2025, 68(2): 122101, https://doi.org/10.1007/s11432-023-3900-7

## 1 Introduction

Constraint satisfaction problem (CSP) is an important branch in the field of artificial intelligence and computer science [1], which can be used to formulate many problems of theoretical and practical interest in diverse areas, such as automatic planning [2] and temporal reasoning [3, 4]. The primary aim of CSP is to find an assignment to a set of variables while adhering to specified constraints. The general CSP is NP-complete [5]. CSP provides a generic framework for various problems, among which the Boolean satisfiability problem (SAT) is the first problem known to be NP-complete [6].

### 1.1 Background

Given a propositional formula conforming to the conjunctive normal form (CNF), the goal of SAT is to determine whether the formula has a satisfying set of truth assignments [7]. In practical engineering applications, it may not be able to find a complete assignment so that all clauses are satisfied. In this case, it is natural to look for a complete assignment to maximize the number of satisfied clauses, leading to the maximum satisfiability problem (MaxSAT). MaxSAT is the optimization version of SAT. Many real-life problems can be more accurately modeled by MaxSAT rather than SAT, such as timetabling problems [8] and the maximum clique problem [9].

An important generalization of MaxSAT is the partial maximum satisfiability problem (PMS), where clauses are separated into hard and soft clauses, and PMS aims to find an assignment that satisfies all hard clauses and maximizes the number of satisfied soft clauses [10]. Many practical constraint problems,

* Corresponding author (email: chuanluo@buaa.edu.cn)

such as network routing [11] and scheduling problems [12], contain both hard and soft constraints. When solving these problems, it is necessary to optimize the soft constraints as much as possible under the premise of ensuring that all the hard constraints are satisfied. Therefore, compared with SAT and MaxSAT, PMS provides a more natural and appropriate way to encode these problems. Consequently, developing efficient methods for solving PMS is of great significance from both theoretical and practical perspectives.

In theory, the PMS problem is NP-hard [13, 14]. The existing PMS solvers can be classified into two basic classes: SAT-based solvers and local search solvers. SAT-based solvers, which successively call high-performance SAT solvers (e.g., [15–18]), can prove the optimality of solutions if necessarily long runs are permitted. However, they may not return good-quality solutions for certain types of instances within acceptable time [14, 19]. Local search solvers are incomplete methods that often find high-quality solutions within reasonable time [14, 20–22].

Local search solvers maintain a complete assignment and iteratively modify the assignment (i.e., flipping the value of a variable). They keep track of the best solution found throughout the search and return it when reaching the termination condition, such as time limit (e.g., [14, 23–25]). In fact, local search solvers have achieved great success in the SAT community (e.g., [14, 23–37]). For solving PMS, local search solvers first generate an initial assignment and then repeatedly select and flip a variable until the termination criterion is met (e.g., a given time budget is exceeded). In the context of local search for PMS, most existing studies [14, 23, 24, 31] focus on improving the mechanism, such as dynamic weight adjustment mechanism [14, 25], for selecting the variable to be flipped. Recent studies have demonstrated that the mechanism for generating initial assignments substantially affects the performance of local search [38, 39]. Local search solvers are designed to generate initial assignments randomly. However, in our opinion, random initialization suffers from a severe issue: once an initial assignment is constructed in an unpromising search space, the solution output by the local search solver is possibly of inferior quality. Thus, an important direction is to design an effective initialization mechanism for boosting PMS solving.

### 1.2 Our main contributions

Inspired by the success of neural networks in diverse application fields [40], in this study, we propose a novel initial assignment prediction approach for optimizing the PMS solving, dubbed InitPMS, which aims to generate initial assignments for PMS instances through graph neural network (GNN). A specific structure of PMS instances is that hard and soft clauses are distinct. In view of such a specific structure, InitPMS is designed based on clause distinction; i.e., it distinguishes between hard and soft clauses. In particular, given a PMS instance, InitPMS first generates an assignment through GNN that distinguishes hard and soft clauses. Subsequently, InitPMS passes this assignment to a local search solver as the initial assignment and calls the local search solver to optimize the assignment. InitPMS is a general approach because it treats the local search solver as a black-box component and can be integrated with any local search solver.

We also design a multi-round mechanism for training InitPMS effectively. Specifically, InitPMS is pretrained on small SAT instances to warm up GNNs underlying InitPMS; then a number of SAT instances formed by the hard clauses of PMS instances are extracted to train InitPMS and improve its ability to handle hard clauses; finally, we train InitPMS on PMS instances from previous MaxSAT Evaluations (MSEs) to enhance its capability of dealing with soft clauses.

To show the generality of InitPMS, we conduct extensive experiments on PMS instances from MSEs 2020 and 2021. The results indicate that InitPMS can dramatically boost the performance of five state-of-the-art local search solvers, including CCEHC [14], Dist [23], SATLike3.0 [24], StableResolver [41], and BandMaxSAT [42]. As a supplementary argument, our results show that incorporating InitPMS could enhance one of the best hybrid PMS solvers in MSE 2021 (i.e., SATLike-c [43]), indicating that InitPMS can advance the state of the art in PMS solving.

### 1.3 Organization of our paper

The remainder of this paper is organized as follows. In Section 2, we introduce the preliminaries, where the definition of PMS and an overview of GCN are introduced. In Section 3, the relevant research on machine learning in the constraint satisfaction problem is presented. In Section 4, we present the InitPMS approach and introduce the basic components of InitPMS in detail. In Section 5, we describe the experimental design. In Section 6, we conduct extensive experiments to evaluate the performance of the InitPMS approach, and further analyze the experimental results. In Section 7, we summarize the work in this paper with a discussion on future work.

## 2 Preliminaries

In this section, we first introduce the common knowledge and definitions of the MaxSAT field, after which we introduce the background of graph convolutional neural network (GCN) and its variants.

### 2.1 Preliminaries of PMS

Given a set of $n$ Boolean variables $\{x_1, x_2, \ldots, x_n\}$, a literal is either a variable $x_i$ or its negation $\neg x_i$. A clause is a disjunction of literals, i.e., $c_i = l_{i,1} \vee l_{i,2} \vee \cdots \vee l_{i,k_i}$, where $l_{i,j}$ is a literal and $k_i$ denotes the length of clause $c_i$. A formula in conjunctive normal form (CNF), i.e., $F = c_1 \wedge c_2 \wedge \cdots \wedge c_m$, is a conjunction of clauses (SAT and MaxSAT instances are CNF formulas).

An assignment is a mapping that assigns a Boolean value (either 0 or 1) to each variable. Given an assignment $\alpha$, each clause is either satisfied or unsatisfied: a clause $c_i$ is satisfied if at least one literal in $c_i$ is true under $\alpha$; otherwise, $c_i$ is unsatisfied. A PMS instance comprises a CNF formula, where all clauses are categorized as either hard or soft.

An assignment $\alpha$ is feasible if it satisfies all hard clauses; otherwise, $\alpha$ is infeasible. Given a PMS instance, $X$ denotes the set of all variables; $H$ and $S$ denote the sets of all hard and soft clauses, respectively.

The PMS problem entails finding an assignment that satisfies all hard clauses and maximizes the number of satisfied soft clauses.

### 2.2 Background of GCN

In recent years, different variants of graph neural networks have been developed. GCN are one of them. Until now, GCN is considered one of the basic graph neural network variants [44].

The general idea of GCN is to apply convolution over a graph. In GCN, convolution is the basic operation. It refers to multiplying the input weight matrices that are commonly known as filters or kernels. The filters act as a sliding window and learn features from neighboring nodes. Within the same layer, the same filter is used, a concept referred to as weight sharing. GCN consists of several convolutional and pooling layers for feature extraction, followed by the final fully-connected layers. The input of each convolutional layer of GCN is the adjacency matrix of the graph and the feature attributes of all nodes, and the output is the new feature embeddings of nodes. The insertion of an adjacency matrix in the forward pass equation of GCNs enables the model to learn the features of neighboring nodes, thus effectively performing a message-passing operation along the nodes within the graph.

With the in-depth research on GCN, various variants of GCN have emerged to solve problems in different scenarios, such as RGCN (relational graph convolutional network) and GGCN (gated graph convolutional network). RGCN is an extension of GCN in multi-relational edge graphs and GGCN uses the node itself as a hidden state while aggregating its neighbor nodes.

## 3 Related work

To the best of our knowledge, InitPMS is the first approach that incorporates the machine learning technique to predict initial assignment for local-search-based PMS solving. Here, we review the related work of solving PMS-related problems using machine learning.

Artificial neural networks have been applied in various problems [45–47] over the past few years and have been successfully utilized on numerous data structures, such as graphs [45], images [48], and sequences [49]. The field of SAT is no exception.

There have been some studies utilizing machine learning within the SAT community. Xu et al. [50] proposed SATzilla, which expresses the SAT problem through feature engineering and selects the most suitable solver for solving the problem via linear regression. Liang et al. [51] proposed a series of measurement methods, including the global learning rate, and used reinforcement learning and logistic regression to optimize the global learning rate, ultimately improving the performance of the solver. Yolcu et al. [52]

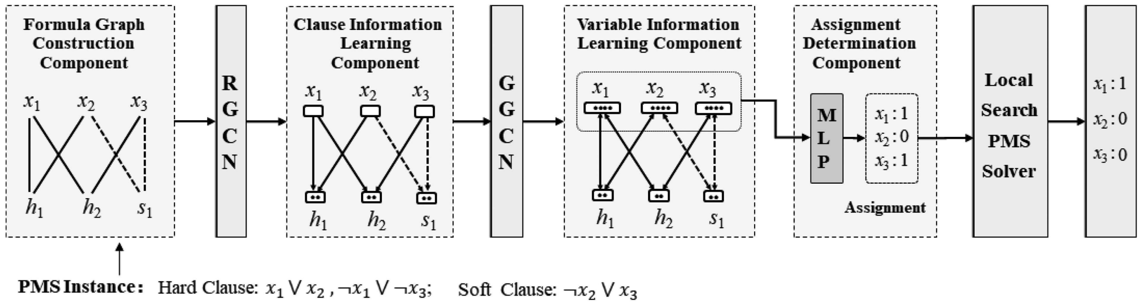


**Figure 1** Top-level design of InitPMS.

used reinforcement learning to simulate the variable selection process in a random local search solver, learning the optimal variable selection strategy. In summary, the existing studies towards SAT solving with machine learning technique mainly focus on four directions: (1) optimizing meta-heuristics [53]; (2) predicting the satisfiability of SAT instances [46]; (3) solving instances through end-to-end neural networks [54]; (4) enhancing conflict-driven-clause-learning (CDCL) solvers [55, 56].

Recently, to determine whether an SAT instance is satisfiable, Selsam et al. [46] proposed an end-to-end neural network model called NeuroSAT by using GNN. Later, Selsam et al. [57] created NeuroCore, which is an evolutionary version of NeuroSAT. NeuroCore guides CDCL solvers with unsatisfiable-core predictions which are computed at regular intervals within the neural networks on GPUs. CDCL solvers equipped with NeuroCore solve 6%–11% more instances than the original ones. Inspired by NeuroSAT, Ref. [39] brought the idea of deep learning into local search SAT solvers. By using GCN, a neural network is trained to guide the local search SAT solvers with initial assignments, and the result shows that the method can enhance local search SAT solvers by solving 27%–62% more instances.

In addition, there have been efforts focusing on utilizing machine learning to solve the MaxSAT problem (e.g., [54]). However, these methods are designed to treat all clauses equally. As discussed before, owing to the specific structure of the PMS problem (i.e., hard clauses and soft clauses are distinct), these approaches are not suitable for solving the PMS problem. In contrast to these methods, our InitPMS approach considers the specific structure of PMS during its initial assignment prediction procedure.

## 4 InitPMS approach

In this section, we propose the InitPMS approach, a clause distinction-based initial assignment prediction approach for optimizing PMS solving by incorporating GNNs.

The top-level design of InitPMS is shown in Figure 1. Taking a given PMS instance as input, InitPMS first generates an assignment (which is possibly infeasible). It then passes the assignment to a local search PMS solver as the initial assignment and calls the solver to output an optimized assignment. InitPMS consists of four components: (1) formula graph construction component, (2) clause information learning component, (3) variable information learning component, and (4) assignment determination component. All components are presented in the following subsections.

### 4.1 Formula graph construction component

The goal of this component is to capture the structure information of PMS instances. Since a PMS instance is expressed as a CNF formula, it is suitable to encode the PMS instance into a formula graph. However, existing formula graph transformation methods (e.g., [39]) do not distinguish hard clauses and soft clauses, which hinders the effective exploitation of the structure information of PMS instances.

To address this issue, we propose a new method of formula graph transformation. In particular, InitPMS encodes a PMS instance $F$ to a bipartite graph $G = (V, C, E)$, where $V$ is the vertex set (i.e., $V = X$); $C$ is the set of all clauses (i.e., $C = H \cup S$), recalling that $H$ is the set of all hard clauses, and $S$ is the set of all soft clauses; $E$ denotes the edge set that describes the relationship between variables and clauses. To ensure a clear distinction between hard and soft clauses, edge set $E$ is divided into two non-intersecting subsets $E_H$ and $E_S$, where $E_H$ is a hard-edge set that reflects the relationship between variables and hard clauses, and $E_S$ is a soft-edge set that represents the relationship between variables and soft clauses. Specifically, for each variable $x \in X$, if variable $x$ or its negation $\neg x$ appears in a hard clause $h \in H$, then $(x, h) \in E_H$. Similarly, if variable $x$ or its negation $\neg x$ appears in a soft clause $s \in S$, then $(x, s) \in E_S$.

Given an example PMS instance, where its hard clauses are $h_1 = x_1 \vee x_2$ and $h_2 = \neg x_1 \vee \neg x_3$, and its soft clause is $s_1 = \neg x_2 \vee x_3$, according to our new transformation method, we have $E_H = \{(x_1, h_1), (x_1, h_2), (x_2, h_1), (x_3, h_2)\}$ and $E_S = \{(x_2, s_1), (x_3, s_1)\}$. Additionally, from the generated formula graph we can extract the adjacency matrix regarding hard clauses, denoted by $A_H$, and the adjacency matrix regarding soft clauses, denoted by $A_S$.

### 4.2 Clause information learning component

The clause information learning component underlying InitPMS takes the constructed formula graph as input and focuses on extracting the feature vectors of both hard clauses and soft clauses. Since the input is a formula graph, it is advisable for InitPMS to employ GNN to learn clause information. Note that PMS instances involve both hard clauses and soft clauses, which distinguishes them from SAT instances where only one type of clause is involved. Conventional GNNs can acquire and process the structural features of the formula graph converted from an ordinary SAT instance, as only one type of edge exists between the variable node and the clause node. Compared with single-edge relationship graphs, the formula graph of a PMS instance is a multi-edge relationship graph, containing two types of edges (soft-clause edges and hard-clause edges) instead of one. Consequently, conventional GNNs may not effectively handle PMS instances because they treat soft-clause edges in the same manner as hard-clause edges, without recognizing the different types of edges and giving priority to hard clauses.

To address this issue, we propose to utilize the RGCN [58]. RGCN separates edges when dealing with multi-edge relationship graphs. In the PMS instance formula graph, only the edges of two different relations are included, namely soft clause edges and hard clause edges. RGCN generates its corresponding subgraph for the edges of these two different relationships, obtains the embedding of the subgraph, and finally gains the global embedding by weighted fusion of different subgraph embeddings. After the last step of weighted fusion, the nodes in the graph achieve the fusion embedding of different relationships instead of the embedding of a single relationship. Consequently, RGCN semantically distinguishes between soft clauses and hard clauses. Hence, RGCN can effectively capture structure information for graphs with edges that define multiple relations between variables and clauses.

To utilize RGCN, for each hard clause $h_i \in H$, we initialize a feature vector $\boldsymbol{hv}_i \in \mathbb{R}^d$ randomly as $h_i$'s representation, and similarly we initialize a feature vector $\boldsymbol{sv}_i \in \mathbb{R}^d$ randomly as $s_i$'s representation, where $d$ is the embedding size. These feature vectors of hard clauses and soft clauses are fed to RGCN to learn the clause information.

In each iteration of RGCN, the feature vector of each hard and soft clause is updated by embedding the overall information of all hard and soft clauses, respectively.

Specifically, at the $t$-th iteration of RGCN, the detailed updating procedures are listed as follows:

$$\boldsymbol{hv}_i^{(t)} = \mathrm{RELU}\Big(\boldsymbol{W}_1 \odot \boldsymbol{hv}_{\mathrm{avg}}^{(t-1)} + \boldsymbol{W}_2 \odot \boldsymbol{hv}_i^{(t-1)}\Big), \tag{1}$$

$$\boldsymbol{sv}_i^{(t)} = \mathrm{RELU}\Big(\boldsymbol{W}_3 \odot \boldsymbol{sv}_{\mathrm{avg}}^{(t-1)} + \boldsymbol{W}_4 \odot \boldsymbol{sv}_i^{(t-1)}\Big), \tag{2}$$

where $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{W}_3$, and $\boldsymbol{W}_4$ are network weights and will be determined through the training process; $\boldsymbol{hv}_i^{(t)}$ and $\boldsymbol{sv}_i^{(t)}$ denote the feature vectors of hard clause $h_i$ and soft clause $s_i$ at the $t$-th iteration, respectively; $\boldsymbol{hv}_{\mathrm{avg}}^{(t)}$ is the average of the vector representations over all hard clauses at the $t$-th iteration, i.e., $\boldsymbol{hv}_{\mathrm{avg}}^{(t)} = (\sum_{i=1}^{|H|} \boldsymbol{hv}_i^{(t)})/|H|$; $\boldsymbol{sv}_{\mathrm{avg}}^{(t)}$ is the average of the vector representations over all soft clauses at the $t$-th iteration, i.e., $\boldsymbol{sv}_{\mathrm{avg}}^{(t)} = (\sum_{i=1}^{|S|} \boldsymbol{sv}_i^{(t)})/|S|$; "RELU" stands for rectified linear unit activation function, applying to each element of the input vector.

Through RGCN, InitPMS can extract the representations of all hard clauses and all soft clauses.

### 4.3 Variable information learning component

This component aims to learn the variable information (i.e., computing the feature vectors of all variables) based on the representations of hard clauses and soft clauses.

We propose employing a GGCN [59], which is suitable for extracting variable information. To leverage the effectiveness of GGCN, for each variable $x_i \in X$, a feature vector $\boldsymbol{xv}_i \in \mathbb{R}^d$ is initialized as variable

$x_i$'s representation, which has the same embedding size as the representations of hard clauses and soft clauses (i.e., $\boldsymbol{hv}_i$ and $\boldsymbol{sv}_i$). The feature vectors of all variables are fed to GGCN to extract variable information.

In each iteration of GGCN, the feature vector of $x_i$ is updated by incorporating the information of those clauses where $x_i$ appears. Since adjacency matrices $A_H$ and $A_S$ record the structure information between variables and clauses, by considering $A_H$ and $A_S$, the clause information can be effectively incorporated into variable representations. Furthermore, the clause representations can be updated by considering the feature vectors of variables. That is, the representations of variables and clauses can be enhanced jointly. Particularly, at the $t$-th iteration of GGCN, the updating procedures of the representations of variables and clauses are given below:

$$\boldsymbol{hv}_i^{(t)} = \mathrm{LSTM}\Big(\boldsymbol{hv}_i^{(t-1)}, \boldsymbol{xv}_{\mathrm{avg}}^{(t-1)}\Big), \tag{3}$$

$$\boldsymbol{sv}_i^{(t)} = \mathrm{LSTM}\Big(\boldsymbol{sv}_i^{(t-1)}, \boldsymbol{xv}_{\mathrm{avg}}^{(t-1)}\Big), \tag{4}$$

$$\boldsymbol{xv}_i^{(t)} = \mathrm{LSTM}\Big(\boldsymbol{xv}_i^{(t-1)}, \boldsymbol{hv}_{\mathrm{avg}}^{(t-1)} \cdot \widetilde{A}_H + \boldsymbol{sv}_{\mathrm{avg}}^{(t-1)} \cdot \widetilde{A}_S\Big). \tag{5}$$

In the above equations, $\boldsymbol{xv}_i^{(t)}$ denotes the feature vector of variable $x_i$ at the $t$-th iteration; $\boldsymbol{xv}_{\mathrm{avg}}^{(t)}$ is the average of the vector representations over all variables at the $t$-th iteration, i.e., $\boldsymbol{xv}_{\mathrm{avg}}^{(t)} = (\sum_{i=1}^{|X|} \boldsymbol{xv}_i^{(t)})/|X|$; 'LSTM' represents a long short-term memory (LSTM) unit with layer normalization. We also conducted symmetrical normalization and MinHash techniques [60] on adjacency matrices (i.e., $A_H$ and $A_S$), resulting in $\widetilde{A}_H$ and $\widetilde{A}_S$. Thus, the sizes of $\widetilde{A}_H$ and $\widetilde{A}_S$ are compatible with the sizes of the representations of hard clauses, soft clauses, and variables.

After updating with GGCN, InitPMS learns the variable information, and the representations of all variables are effectively captured.

### 4.4 Assignment determination component

Based on the representation of each variable $x_i \in X$, the assignment determination component calculates probability $p_i$ for variable $x_i$ to be 1 via a multi-layer perceptron (MLP). Then, InitPMS uses probability $p_i$ to decide which Boolean value (either 0 or 1) should be assigned to $x_i$. After all variables are assigned, the initial assignment is determined. The objective of InitPMS is to minimize the binary cross-entropy loss, which is described as $\mathcal{L} = -(\sum_{i=1}^{|X|} [y_i \cdot \log(p_i) + (1-y_i) \cdot \log(1-p_i)])/|X|$, where $y_i$ is the ground truth of variable $x_i$.

Once the assignment is determined, InitPMS passes the generated assignment to a local search PMS solver as the initial assignment and calls the local search solver to further optimize the assignment. The whole process is shown in Algorithm 1, where for a solution $\alpha$, cost($\alpha$) represents the number of unsatisfied soft clauses under the premise of all hard clauses satisfied; if the hard clauses are not all satisfied, it is considered to be positive infinity. After receiving the initial assignment $N$ given by InitPMS, each local search solver continuously chooses a variable and flips it according to its heuristics (lines 9 and 10). The current best feasible solution $\alpha^*$ is updated whenever a better feasible solution is found (lines 6 and 7), and the final $\alpha^*$ and cost$^*$ are reported as the output upon reaching the termination condition of the underlying local search solver (lines 13 and 14). We note that InitPMS is a general approach and can cooperate with any local search PMS solver. In this study, we utilized InitPMS to improve five local search PMS solvers (see Subsection 5.2 for more details).

### 4.5 Multi-round training mechanism for InitPMS

InitPMS relies on GNNs including RGCN and GGC; thus, it is important to design an effective training strategy. A substantial number of training instances are desired. In the research field of PMS solving, PMS instances are taken from MSEs[1]. However, since the number of available PMS instances is insufficient, it is challenging to train InitPMS effectively by directly using PMS instances from MSEs as the only training set.

To address this issue, we propose an effective, multi-round mechanism for training InitPMS, which consists of three rounds and works as follows. In the first round, InitPMS is pre-trained on huge numbers of small SAT instances to warm up the RGCN and GGCN underlying InitPMS. In the second round,

1) https://maxsat-evaluations.github.io/.

**Algorithm 1** PMS solving of local search solver equipped with InitPMS.

**Input**: PMS instance $P$;
**Output**: A feasible solution $\alpha$ and its cost, or "no feasible solution found";

$N \leftarrow$ an initial assignment for $P$ predicted by InitPMS;
$\text{cost}^* := +\infty$; $\alpha^* := \emptyset$;
**while** End condition not reached **do**
    $\alpha \leftarrow N$;
    **while** Restart condition not reached **do**
        **if** $\alpha$ is feasible & $\text{cost}(\alpha) < \text{cost}^*$ **then**
            $\alpha^* := \alpha$; $\text{cost}^* := \text{cost}(\alpha)$;
        **end if**
        $x \leftarrow$ Select variables by some heuristics;
        $\alpha := \alpha$ with $x$ flipped;
    **end while**
**end while**
**if** $\alpha^*$ is feasible **then**
    **return** $(\text{cost}^*, \alpha^*)$;
**else**
    **return** "no feasible solution found";
**end if**

we extract a number of SAT instances formed by the hard clauses of PMS instances. These hard-clause SAT instances are then used to train InitPMS to improve its ability to handle hard clauses. In the third round, we train InitPMS on those PMS instances from previous MSEs to enhance their capability to deal with soft clauses.

#### 4.5.1 *First round of training*

As recommended by [39], we use a random generator to construct small SAT instances comprising 10 to 20 variables, and the clause-to-variable ratio ranging from 2 to 6. Following this setup, we first construct 5 million small SAT instances. Then, for each generated SAT instance, we attempt to solve it using a state-of-the-art SAT solver named CaDiCaL [61] with a cutoff time of 3000 CPU seconds. If an instance proves to be satisfiable by CaDiCaL, then the instance is added into our first-round training set, and, for each variable $x$ in this instance, $x$'s ground truth is labeled with $x$'s Boolean value in CaDiCaL's reported solution. Through the above procedure, we obtain the first-round training set containing approximately 3.6 million instances.

#### 4.5.2 *Second round of training*

To solve PMS instances, in the second round, it is vital to leverage the structure information of PMS instances. As described before, local search is an incomplete method. To avoid overfitting, it is reasonable to collect the PMS instances from MSEs' complete tracks rather than incomplete tracks. Furthermore, in this study, we use PMS instances from MSEs 2020 and 2021 as the testing set to evaluate the effectiveness of InitPMS (see Subsection 5.1). To this end, we collect 2219 PMS instances from the complete tracks of MSEs 2015–2019[2)].

Since it is recognized that hard clauses in PMS instances should be emphasized [14, 23], in the second round, it is advisable to improve the ability of InitPMS to process hard clauses. Hence, for each PMS instance inst, if inst does not contain any hard clause, then it is discarded from this round; otherwise, we extract an SAT instance formed by all hard clauses in inst, and the resulting SAT instance is called hard-clause SAT instance. Next, we obtain the second-round training set including 1992 hard-clause SAT instances.

#### 4.5.3 *Third round of training*

The task of the third round is to enhance the capability of InitPMS to process soft clauses. Therefore, in this round, we use the 2219 PMS instances from the complete tracks of MSEs 2015–2019. For each PMS instance inst, we attempt to solve it through a PMS solver called SATLike-c with a cutoff time of 3000 CPU seconds. If SATLike-c cannot find a feasible solution for inst, then inst is discarded; otherwise, inst is retained, and, for each variable $x$ in inst, $x$'s ground truth is labeled with $x$'s Boolean value in

2) Some instances from MSEs 2015–2019 are the same as those from MSEs 2020 and 2021. In this study, we eliminate the overlapped instances from the training set. Thus, the training and testing sets are strictly disjoint.

SATLike-c's reported, feasible solution. Finally, we obtain the third-round training set consisting of 2183 PMS instances.

## 5 Experiments design

In this section, we conduct experiments on extensive PMS instances to evaluate the general effectiveness of InitPMS.

### 5.1 Testing benchmarks

InitPMS is evaluated on those PMS instances collected from both complete and incomplete tracks of MSEs 2020 and 2021. Since InitPMS is a GNN-based approach, a small fraction of PMS instances cannot be handled by InitPMS on our workstation owing to the insufficiency of video memory[3)]. Hence, we discard those instances accordingly, after which 513 and 257 PMS instances remain in MSE 2020's complete track[4)] (20-Comp) and incomplete track[5)] (20-Incomp), respectively, while 541 and 155 PMS instances remain in MSE 2021's complete track[6)] (21-Comp) and incomplete track[7)] (21-Incomp), respectively.

### 5.2 State-of-the-art PMS solvers

Since this study aims to improve local search-based PMS solving, it is necessary to evaluate the effectiveness of InitPMS on various local search PMS solvers. In our experiments, InitPMS was tested on five state-of-the-art local search solvers, namely CCEHC [14], Dist [23], SATLike3.0 [24], StableResolver [41], and BandMaxSAT [42], which were competitive solvers of incomplete tracks in previous MSEs. We compared the performance of each of the five local search solvers when initialized by InitPMS and by random assignments. In addition, we also recorded performances of these local search solvers when initialized by feasible solutions generated by an SAT solver called Kissat [62], which is a condensed and improved reimplementation of CaDiCaL [61]. We refer to this initialization method of local search solvers as InitSAT.

Furthermore, to confirm InitPMS's benefits of distinguishing hard and soft clauses, we evaluated the performance of these five local search solvers with an alternative version of InitPMS called InitPMS-alt, which treats hard clauses and soft clauses equally.

Moreover, we adopted one of the best hybrid PMS solvers and its variants in MSE 2021 (i.e., SATLike-c and SATLike-ck). By hybridizing high-performance SAT-based and local search solvers, the hybrid solver represents the current state of the art in PMS solving. We conducted experiments to integrate the solver with InitPMS to show that InitPMS can help advance the state of the art in PMS solving.

### 5.3 Experimental setup

All experiments in this study were conducted on a workstation equipped with Intel I9-10900X CPU, 32 GB memory, and NVIDIA GeForce RTX 3090 GPU with 24 GB video memory, running the operating system of Ubuntu 20.04.

Each solver performed one run on each instance, with the cutoff time of 300 CPU seconds, following the rules of the incomplete track of MSEs.

In our experiments, for each PMS solver ps, we used "pure" (or ps) to indicate the pure version (the solver ps itself), and "+SAT" (or ps-SAT), "+InitPMS " (or ps-InitPMS) and "+alt" (or ps-alt) to indicate solver ps equipped with InitSAT, InitPMS, and InitPMS-alt, respectively.

In our experiments, we used two metrics to justify the performance of PMS solvers. For each PMS solver ps on each testing benchmark tb, the first metric is the number of winning instances, which is calculated in the following way: for each instance, if the cost of the solution found by solver ps is the lowest among that of all competing solvers in the table, the number of winning instances of ps is increased by one; the second metric is the average competition score over benchmark tb, which is adopted by the

3) The video memory usage problem remains a key challenge in the GNN research field and is beyond this paper's scope. For large PMS instances that exceed video memory, if an InitPMS-based solver attempts to solve them, rather than employing InitPMS, it resorts to its original initialization method, which is equivalent to the original solver.

4) http://www.cs.toronto.edu/maxsat-lib/maxsat-instances/downloads/ms-evals/ms20_complete_unwt.zip.

5) http://www.cs.toronto.edu/maxsat-lib/maxsat-instances/downloads/ms-evals/ms20_incomplete_unwt.zip.

6) https://www.cs.helsinki.fi/group/coreo/mse2021/mse2021_benchmarks/mse21_complete_unwt.zip.

7) https://www.cs.helsinki.fi/group/coreo/mse2021/mse2021_benchmarks/mse21_incomplete_wt.zip.

**Table 1** Results for local search PMS solvers on the number of winning instances[a)].

| Solvers | 20-Comp (513) | | | | 20-Incomp (257) | | | | 21-Comp (541) | | | | 21-Incomp (155) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS |
| CCEHC | 227 | 234 | 235 | **257** | 73 | 99 | 81 | **104** | 240 | 276 | 243 | **284** | 53 | 61 | 55 | **57** |
| Dist | 218 | 263 | 229 | **309** | 75 | 109 | 84 | **117** | 258 | 295 | 276 | **324** | 55 | 62 | 57 | **64** |
| SATLike3.0 | 292 | 331 | 298 | **338** | 130 | 159 | 135 | **168** | 378 | 415 | 379 | **442** | 67 | 73 | 68 | **76** |
| StableResolver | 299 | 368 | 301 | **374** | 121 | 160 | 126 | **173** | 328 | 356 | 330 | **414** | 51 | 68 | 54 | **74** |
| BandMaxSAT | 344 | 358 | 348 | **363** | 178 | 186 | 180 | **188** | 422 | 437 | 424 | **448** | 104 | 109 | 105 | **112** |

a) The best results for each benchmark are in bold.

**Table 2** Results for local search PMS solvers on competition score[a)].

| Solvers | 20-Comp | | | | 20-Incomp | | | | 21-Comp | | | | 21-Incomp | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS |
| CCEHC | 0.6609 | 0.7024 | 0.6780 | **0.7383** | 0.6320 | 0.7148 | 0.6626 | **0.7461** | 0.6989 | 0.7532 | 0.7076 | **0.7604** | 0.6499 | 0.7092 | 0.6542 | **0.7331** |
| Dist | 0.6912 | 0.7294 | 0.7068 | **0.7909** | 0.6558 | 0.7237 | 0.6742 | **0.7649** | 0.7692 | 0.8059 | 0.7718 | **0.8368** | 0.6797 | 0.7116 | 0.6942 | **0.7575** |
| SATLike3.0 | 0.7600 | 0.7958 | 0.7676 | **0.8411** | 0.6705 | 0.7690 | 0.6864 | **0.7970** | 0.8076 | 0.8404 | 0.8227 | **0.8832** | 0.6061 | 0.6606 | 0.6126 | **0.7163** |
| StableResolver | 0.7217 | 0.8032 | 0.7304 | **0.8201** | 0.6063 | 0.7541 | 0.6289 | **0.7884** | 0.7501 | 0.8364 | 0.7656 | **0.8834** | 0.6530 | 0.7096 | 0.6619 | **0.7491** |
| BandMaxSAT | 0.8434 | 0.8515 | 0.8493 | **0.8646** | 0.7998 | 0.8124 | 0.8013 | **0.8196** | 0.8649 | 0.8829 | 0.8718 | **0.8914** | 0.8042 | 0.8102 | 0.8054 | **0.8136** |

a) The best results for each benchmark are in bold.

incomplete tracks of recent MSEs. Following the rules[8)] of MSE 2021's incomplete track, for a single instance, the competition score is computed by the sum of the ratios between the best solution found by a given solver and the best solution found by all solvers. Hence, the value of the average competition score over a benchmark ranges between 0 and 1. For each of these two metrics, if a PMS solver achieves a larger metric value, it indicates superior performance on that benchmark.

## 6 Experimental results

In this section, we report and discuss the experimental results to demonstrate the effectiveness of InitPMS.

We first study the performance improvement of the five state-of-the-art local search PMS solvers (i.e., CCEHC, Dist, SATLike3.0, StableResolver, and BandMaxSAT) brought by InitPMS. Tables 1 and 2 demonstrate the comparative results of these five solvers with and without InitPMS on the metrics of the number of winning instances and competition score, respectively[9)].

As can be observed from Table 1, integrating InitPMS significantly boosts the number of winning instances for all local search PMS solvers on all four testing benchmarks. For example, on the 20-Incomp benchmark, the number of winning instances is increased by 31, 42, 38, 52, and 10, respectively.

Similarly, from Table 2, on all four testing benchmarks, integrating InitPMS significantly improves the competition scores of all local search PMS solvers. For example, on the 21-Comp benchmark, the competition score of CCEHC, Dist, SATLike3.0, StableResolver, and BandMaxSAT is increased by 6.15%, 6.76%, 7.56%, 13.33%, and 2.65%, respectively.

The performances of the five state-of-the-art local search PMS solvers when initialized by InitSAT and InitPMS are also shown in Tables 1 and 2. According to these two tables, InitPMS outperforms InitSAT with respect to both the number of winning instances and competition score.

In summary, by incorporating InitPMS, all these five local search solvers gain different degrees of improvements on extensive testing instances with respect to different evaluation metrics for PMS. The experimental results indicate the generality of InitPMS and demonstrate that InitPMS can significantly enhance the performance of various local search PMS solvers.

### 6.1 Incorporating InitPMS into hybrid PMS solver

As described in Subsection 5.2, a hybrid PMS solver namely SATLike-c is one of the best incomplete PMS solvers in MSE 2021 and represents the current state of the art in PMS solving.

We conducted experiments to incorporate InitPMS into the hybrid PMS solver on all four testing benchmarks and compared the results of the hybrid solver with and without InitPMS on the metrics of the number of winning instances and competition score, respectively[10)].

8) https://maxsat-evaluations.github.io/2021/rules.html.

9) In Table 1, the number in brackets denotes the number of PMS instances in each benchmark.

10) For SATLike-c, the local search component uses SAT-based initialization methods, which means that it already has the "+SAT" option. For "+alt" option of SATLike-c, the performance has been tested and is worse than the original version of the solver as expected. Therefore, the "+alt" option of SATLike-c is not included.

**Table 3** Results for hybrid PMS solvers on the number of winning instances a).

| Benchmark | SATLike-c | SATLike-cInitPMS |
|---|---|---|
| 20-Comp | 464 | **473** |
| 20-Incomp | 175 | **185** |
| 21-Comp | 483 | **496** |
| 21-Incomp | 117 | **121** |

a) The best results for each benchmark are in bold.

**Table 4** Results for hybrid solvers on competition score a).

| Benchmark | SATLike-c | SATLike-cInitPMS |
|---|---|---|
| 20-Comp | 0.8919 | **0.9136** |
| 20-Incomp | 0.8627 | **0.8832** |
| 21-Comp | 0.9037 | **0.9229** |
| 21-Incomp | 0.8528 | **0.8742** |

a) The best results for each benchmark are in bold.

As can be observed from Table 3, on all four testing benchmarks, the hybrid PMS solver incorporating InitPMS achieves better performance than that without InitPMS in terms of the number of the winning instances. For example, on the 21-Comp benchmark, 13 more winning instances are obtained by SATLike-c with InitPMS than SATLike-c without InitPMS. The comparison results on competition score are shown in Table 4, where on all four testing benchmarks, the hybrid PMS solver equipped with InitPMS also achieves improvement with respect to competition score. These positive results indicate that InitPMS might help advance the state of the art in PMS solving.

### 6.2 Effectiveness of clause distinction

As discussed before, InitPMS distinguishes hard clauses and soft clauses when generating assignments for PMS instances. To evaluate the effectiveness of the clause distinction mechanism, we directly compare InitPMS against InitPMS-alt, recalling that InitPMS-alt is an alternative version of InitPMS that treats hard clauses and soft clauses equally.

Tables 1 and 2 also summarize the performance of five PMS solvers equipped with InitPMS-alt and InitPMS. It can be observed that on all four testing benchmarks, all five PMS solvers equipped with InitPMS perform better than those with InitPMS-alt in terms of all evaluation metrics, confirming the effectiveness of clause distinction.

### 6.3 Efficiency of PMS solving with InitPMS

To get a glimpse of the role of InitPMS in improving the efficiency of local search solvers in PMS solving, we report in Figures 2 and 3 the number of winning instances found by these solvers during the above solving process. In each subfigure of Figures 2 and 3, the $X$-axis represents the solving time in seconds, and the $Y$-axis represents the number of winning instances found by different solvers on the whole dataset including the four benchmarks at the current time. A solver is expected to obtain optimal solutions in a shorter time. Therefore, the larger the value of the $Y$-axis, the higher the efficiency of the solver.

According to Figures 2 and 3, in the beginning InitSAT dominates the number of winning instances. Later and finally, InitPMS performs the best. This observation is expected because InitSAT and InitPMS follow distinct approaches. InitSAT greedily generates a feasible solution with the aid of an SAT solver first, and then passes the initial feasible solution to the local search solvers for improvement. In contrast, InitPMS directly learns the optimality of the solutions for PMS instances via the designed GNN and generates an assignment prediction that is then taken as a starting point of the local search PMS solver. These results further demonstrate the effectiveness of InitPMS in solving PMS instances.

### 6.4 Infeasible solutions in InitPMS predicted solutions

As mentioned earlier, InitPMS ultimately focuses on learning at the global level; therefore, some of the predicted solutions generated by InitPMS are not directly feasible solutions. This phenomenon naturally raises the question of whether InitPMS contributes to enhancing the local search solvers even when it computes an infeasible solution. To answer this question, we implemented experiments investigating the performance of InitPMS on those non-feasible predicted solutions. The experimental results are shown

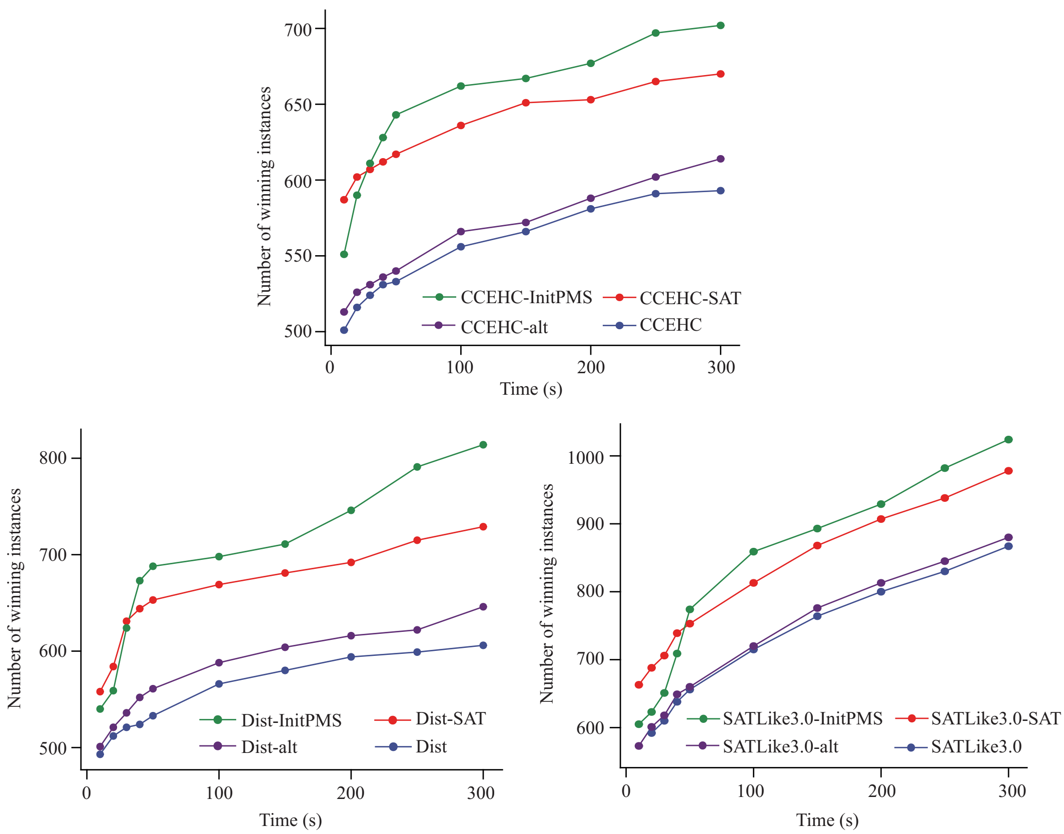


**Figure 2** Results for CCEHC, Dist, SATLike3.0 on changes of the number of winning instances.

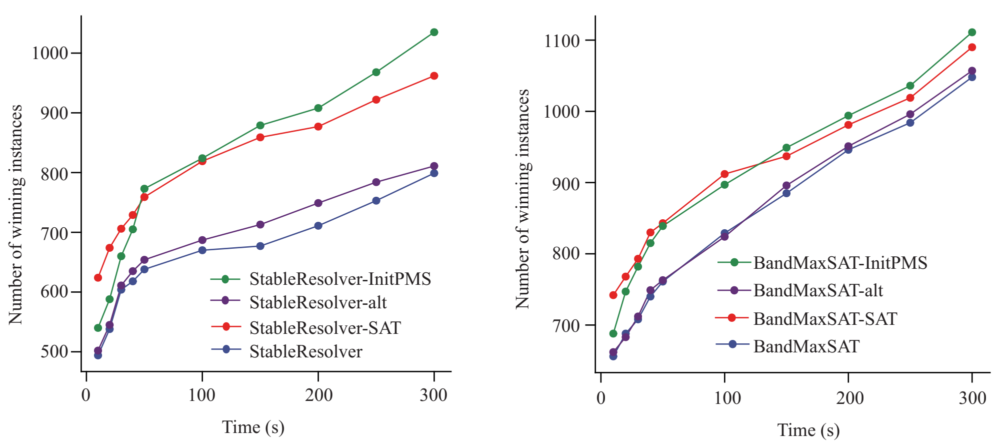


**Figure 3** Results for StableResolver, BandMaxSAT on changes of the number of winning instances.

in Table 5. The numbers to the right of the benchmark in Table 5 represent the number of infeasible solutions in the predicted solutions generated by InitPMS. The values in Table 5 are the competition scores of each solver on those infeasible solutions. The results show that InitPMS still performs the best in all scenarios. As an example, for SATLike3.0 on 21-Incomp, the average score over all benchmarks for pure, "+SAT", "+alt", "+InitPMS " is 0.5986, 0.6231, 0.6097, 0.6546, respectively. These results demonstrate the effectiveness of InitPMS, even when it generates an infeasible solution.

### 6.5 Enhancing local search solvers with longer cutoff time

The results indicate that the longer the local search algorithm runs, the better its performance tends to be. In this subsection, we would like to explore the following question: how does InitPMS perform when the local search algorithm runs longer than 300 s? In this experiment, the cutoff time of the solvers is

**Table 5** Results of infeasible predicted solution on competition score[a)].

| Solvers | 20-Comp (277) | | | | 20-Incomp (150) | | | | 21-Comp (302) | | | | 21-Incomp (103) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS |
| CCEHC | 0.6397 | 0.6680 | 0.6436 | **0.6761** | 0.6103 | 0.6392 | 0.6172 | **0.648** | 0.6556 | 0.6773 | 0.6596 | **0.6889** | 0.6179 | 0.6205 | 0.6136 | **0.6240** |
| Dist | 0.6627 | 0.7094 | 0.6681 | **0.7164** | 0.6535 | 0.6838 | 0.6611 | **0.6903** | 0.7120 | 0.7474 | 0.7192 | **0.7655** | 0.6347 | 0.6694 | 0.6447 | **0.6839** |
| SATLike3.0 | 0.7267 | 0.7473 | 0.7298 | **0.7689** | 0.6630 | 0.7053 | 0.6641 | **0.7248** | 0.7588 | 0.7876 | 0.7644 | **0.8078** | 0.5986 | 0.6231 | 0.6097 | **0.6546** |
| StableResolver | 0.6993 | 0.7297 | 0.7039 | **0.7546** | 0.6095 | 0.6481 | 0.6149 | **0.6751** | 0.7141 | 0.7583 | 0.7231 | **0.7754** | 0.6271 | 0.6520 | 0.6377 | **0.6715** |
| BandMaxSAT | 0.7995 | 0.8067 | 0.7941 | **0.8032** | 0.7715 | 0.7851 | 0.7787 | **0.7869** | 0.8177 | 0.8296 | 0.8168 | **0.8363** | 0.7604 | 0.7775 | 0.7694 | **0.7784** |

a) The best results for each benchmark are in bold.

**Table 6** Results for local search PMS solvers with 1000 s on the number of winning instances[a)].

| Solvers | 20-Comp (513) | | | | 20-Incomp (257) | | | | 21-Comp (541) | | | | 21-Incomp (155) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS |
| CCEHC | 236 | 247 | 251 | **258** | 71 | 82 | 76 | **84** | 253 | 290 | 263 | **305** | 59 | 63 | 61 | **66** |
| Dist | 236 | 276 | 246 | **318** | 78 | 81 | 78 | **86** | 273 | 300 | 279 | **329** | 58 | 65 | 60 | **72** |
| SATLike3.0 | 314 | 352 | 325 | **370** | 108 | 112 | 109 | **114** | 325 | 362 | 333 | **391** | 59 | 62 | 60 | **63** |
| StableResolver | 385 | 393 | 393 | **402** | 158 | 159 | 158 | **159** | 424 | 429 | 427 | **443** | 87 | 89 | 88 | **89** |
| BandMaxSAT | 355 | 378 | 367 | **389** | 127 | 132 | 129 | **134** | 364 | 385 | 373 | **397** | 69 | 73 | 69 | **77** |

a) The best results for each benchmark are in bold.

**Table 7** Results for local search PMS solvers with 1000 s on competition score[a)].

| Solvers | 20-Comp | | | | 20-Incomp | | | | 21-Comp | | | | 21-Incomp | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS |
| CCEHC | 0.6719 | 0.7257 | 0.7176 | **0.7614** | 0.6945 | 0.7745 | 0.7071 | **0.7894** | 0.6979 | 0.7571 | 0.7203 | **0.7631** | 0.6880 | 0.7260 | 0.6996 | **0.7641** |
| Dist | 0.7279 | 0.7464 | 0.7370 | **0.7993** | 0.7015 | 0.7296 | 0.7127 | **0.7405** | 0.7634 | 0.8002 | 0.7786 | **0.8469** | 0.7122 | 0.7471 | 0.7231 | **0.7814** |
| SATLike3.0 | 0.7899 | 0.8054 | 0.7910 | **0.8495** | 0.7169 | 0.7384 | 0.7183 | **0.7531** | 0.8061 | 0.8485 | 0.8344 | **0.8678** | 0.6485 | 0.6741 | 0.6563 | **0.6923** |
| StableResolver | 0.8815 | 0.9145 | 0.8968 | **0.9336** | 0.8987 | 0.9006 | 0.8997 | **0.9148** | 0.8433 | 0.8874 | 0.8669 | **0.9267** | 0.8537 | 0.8684 | 0.8664 | **0.8775** |
| BandMaxSAT | 0.8564 | 0.8724 | 0.8578 | **0.8870** | 0.7688 | 0.7714 | 0.7703 | **0.7866** | 0.8157 | 0.8481 | 0.8661 | **0.8824** | 0.7272 | 0.7479 | 0.7394 | **0.7600** |

a) The best results for each benchmark are in bold.

**Table 8** Results for local search PMS solvers on the number of winning instances for pure MaxSAT[a)].

| Solvers | 20-Comp (55) | | | | 20-Incomp (39) | | | | 21-Comp (55) | | | | 21-Incomp (31) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS |
| CCEHC | 32 | 31 | 32 | **34** | 28 | 28 | 27 | **29** | 34 | 34 | 34 | **34** | 30 | 29 | 30 | **31** |
| Dist | 34 | 34 | 32 | **33** | 25 | 25 | 25 | **28** | 36 | 36 | 35 | **36** | 25 | 25 | 25 | **25** |
| SATLike3.0 | 39 | 39 | 39 | **40** | 26 | 26 | 27 | **28** | 38 | 37 | 38 | **38** | 22 | 22 | 22 | **22** |
| StableResolver | 28 | 29 | 28 | **30** | 23 | 23 | 24 | **25** | 35 | 35 | 36 | **37** | 23 | 23 | 23 | **24** |
| BandMaxSAT | 47 | 47 | 47 | **48** | 31 | 31 | 31 | **32** | 47 | 47 | 47 | **47** | 26 | 26 | 26 | **27** |

a) The best results for each benchmark are in bold.

**Table 9** Results for local search PMS solvers on competition score for pure MaxSAT[a)].

| Solvers | 20-Comp | | | | 20-Incomp | | | | 21-Comp | | | | 21-Incomp | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS | pure | +SAT | +alt | +InitPMS |
| CCEHC | 0.5845 | 0.5733 | 0.5694 | **0.5884** | 0.7441 | 0.7390 | 0.7370 | **0.7560** | 0.6234 | 0.6159 | 0.6199 | **0.6277** | 0.9942 | 0.9937 | 0.9938 | **0.9954** |
| Dist | 0.7315 | 0.7337 | 0.7308 | **0.7442** | 0.8226 | 0.8278 | 0.8324 | **0.8425** | 0.7831 | 0.7885 | 0.7803 | **0.7891** | 0.9743 | 0.9743 | 0.9740 | **0.9754** |
| SATLike3.0 | 0.9331 | 0.9405 | 0.9447 | **0.9369** | 0.9689 | 0.9495 | 0.9511 | **0.9858** | 0.9218 | 0.9119 | 0.9211 | **0.9394** | 0.9694 | 0.9702 | 0.9694 | **0.9713** |
| StableResolver | 0.6954 | 0.7035 | 0.7053 | **0.6995** | 0.8304 | 0.8274 | 0.8255 | **0.8327** | 0.7966 | 0.7825 | 0.7727 | **0.8037** | 0.9770 | 0.9770 | 0.9763 | **0.9777** |
| BandMaxSAT | 0.9779 | 0.9704 | 0.9604 | **0.9909** | 0.9676 | 0.9607 | 0.9684 | **0.9784** | 0.9696 | 0.9689 | 0.9612 | **0.9801** | 0.9930 | 0.9928 | 0.9929 | **0.9931** |

a) The best results for each benchmark are in bold.

increased to 1000 s, and the resulting performances of different solvers are shown in Tables 6 and 7. It can be observed from Tables 6 and 7 that with a significant increase in time, the performances of the original versions of all the solvers have been improved. Notably, the InitPMS approach remains effective and helps to improve the performance of all the local search solvers with a longer cutoff time.

### 6.6 Performance on pure MaxSAT

Since PMS is a variant of MaxSAT, one might be curious about how InitPMS performs on pure MaxSAT instances. Tables 8 and 9 show the performance of each solver on pure MaxSAT instances in four datasets. The numbers to the right of the benchmark represent the number of pure MaxSAT instances. As can be observed from Tables 8 and 9, InitPMS improves the performances of these local solvers on pure MaxSAT instances. However, because the proportion of pure MaxSAT instances is relatively small in the benchmarks, the degree of improvement on pure MaxSAT instances may not be as significant as that on PMS instances.

### 6.7 Non-independent combination of GNNs and local search

Since the focus of this study is the initial assignment prediction for PMS with GNN, the proposed initialization module InitPMS is independent of the local search solvers. It is noted that neural networks can be combined with local search solvers in a non-independent way. Here, we explore the combination

**Table 10** Results for SATLike3.0 with probability on the number of winning instances[a)].

| Benchmark | SATLike3.0 | SATLike3.0+InitPMS | SATLike3.0+prob |
|---|---|---|---|
| 20-Comp | 371 | 406 | **411** |
| 20-Incomp | 165 | 189 | **193** |
| 21-Comp | 392 | 454 | **459** |
| 21-Incomp | 79 | 122 | **123** |

a) The best results for each benchmark are in bold.

**Table 11** Results for SATLike3.0 with probability on competition score[a)].

| Benchmark | SATLike3.0 | SATLike3.0+InitPMS | SATLike3.0+prob |
|---|---|---|---|
| 20-Comp | 0.7915 | 0.9228 | **0.9336** |
| 20-Incomp | 0.7386 | 0.9328 | **0.9458** |
| 21-Comp | 0.8134 | 0.9620 | **0.9711** |
| 21-Incomp | 0.6134 | 0.9261 | **0.9309** |

a) The best results for each benchmark are in bold.

of these two techniques as follows: the neural network outputs the probability of each variable being assigned a value of 0 or 1. This probability not only serves as the initial assignment of the local search solvers but also guides the local search process. Specifically, when a local search solver falls into a local optimum, it restarts based on the above probability information. Experimental results on one of the local search solvers, viz., SATLike3.0, are shown in Tables 10 and 11. In Tables 10 and 11, we use "SATLike3.0+prob" to indicate solver SATLike3.0 equipped with the above probability. These results suggest that compared with the InitPMS, this non-independent form of combining neural networks and local search achieves better performance.

## 7 Conclusion and future work

In this study, we propose a novel and effective initial assignment prediction approach dubbed InitPMS for PMS instances. Because PMS instances contain both hard clauses and soft clauses, InitPMS incorporates this specific structure of PMS instances when predicting initial assignments by distinguishing between hard clauses and soft clauses. Extensive experiments on numerous PMS instances collected from MSEs 2020 and 2021 demonstrate that InitPMS can significantly boost the performance of five state-of-the-art local search PMS solvers, confirming the generality of InitPMS. Furthermore, our experimental results show that incorporating InitPMS could enhance the best incomplete PMS solver in MSE 2021, indicating that InitPMS might help advance the state of the art in PMS solving.

This study opens a research direction of using graph neural networks to boost local search for PMS. For future work, we plan to extend our approach to the weighted version of PMS by taking the weights into account, which requires non-trivial work on adjusting the network. We would also like to explore further the non-independent combination of neural networks and local search.

**Acknowledgements** This work was supported in part by National Key Research and Development Program of China (Grant No. 2022YFB4502003), National Natural Science Foundation of China (Grant Nos. 62172072, 62202025, 62122078, 62302492), Natural Science Foundation of Liaoning Province of China (Grant No. 2021-MS-114), CCF-Huawei Populus Grove Fund (Grant No. CCF-HuaweiSY202311), and Frontier Cross Fund Project of Beihang University.

**References**

1 Huang P, Yin M H. An upper (lower) bound for Max (Min) CSP. Sci China Inf Sci, 2014, 57: 072109
2 Huang H, Yang S L, Li X Q, et al. An embedded hamiltonian graph-guided heuristic algorithm for two-echelon vehicle routing problem. IEEE Trans Cybern, 2022, 52: 5695–5707
3 Mouhoub M, Charpillet F, Haton J P. Experimental analysis of numeric and symbolic constraint satisfaction techniques for temporal reasoning. Constraints, 1998, 3: 151–164
4 Liu C J, Zhu E Q, Zhang Y K, et al. Characterization, verification and generation of strategies in games with resource constraints. Automatica, 2022, 140: 110254
5 Mackworth A K, Freuder E C. The complexity of constraint satisfaction revisited. Artif Intell, 1993, 59: 57–62
6 Cook S A. The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, Shaker Heights, 1971. 151–158
7 Li Y, Liu G, Lou X, et al. Intra-individual comparison of different gadolinium-based contrast agents in the quantitative evaluation of C6 glioma with dynamic contrast-enhanced magnetic resonance imaging. Sci China Life Sci, 2017, 60: 11–15
8 Demirović E, Musliu N. MaxSAT-based large neighborhood search for high school timetabling. Comput Operations Res, 2017, 78: 172–180
9 Fang Z W, Li C M, Xu K. An exact algorithm based on MaxSAT reasoning for the maximum weight clique problem. J Artif Intell Res, 2016, 55: 799–833

10 Li C M, Manyà F. MaxSAT, hard and soft constraints. In: Proceedings of the Handbook of Satisfiability. Amsterdam: IOS Press, 2009. 613–631
11 Feng Y, Bastani O, Martins R, et al. Automated synthesis of semantic malware signatures using maximum satisfiability. In: Proceedings of the Network and Distributed System Security Symposium, 2017
12 Demirović E, Musliu N, Winter F. Modeling and solving staff scheduling with partial weighted MaxSAT. Ann Oper Res, 2019, 275: 79–99
13 Cohen D A, Cooper M C, Jeavons P. A complete characterization of complexity for Boolean constraint optimization problems. In: Proceedings of the International Conference on Principles and Practice of Constraint Programming, 2004. 212–226
14 Luo C, Cai S W, Su K L, et al. CCEHC: an efficient local search algorithm for weighted partial maximum satisfiability. Artif Intell, 2017, 243: 26–44
15 Ansótegui C, Bonet M L, Levy J. SAT-based MaxSAT algorithms. Artif Intell, 2013, 196: 77–105
16 Ignatiev A, Morgado A, Marques-Silva J. RC2: an efficient MaxSAT solver. J Satisfiability Boolean Modeling Comput, 2019, 11: 53–64
17 Demirovic E, Stuckey P J. Techniques inspired by local search for incomplete MaxSAT and the linear algorithm: varying resolution and solution-guided search. In: Proceedings of the International Conference on Principles and Practice of Constraint Programming, 2019. 177–194
18 Piotrów M. UWrMaxSat: efficient solver for MaxSAT and pseudo-boolean problems. In: Proceedings of the 32nd International Conference on Tools with Artificial Intelligence, 2020. 132–136
19 Chieu H L, Lee W S. Relaxed survey propagation for the weighted maximum satisfiability problem. J Artif Intell Res, 2009, 36: 229–266
20 Wang Y Y, Ouyang D T, Zhang L M, et al. A novel local search for unicost set covering problem using hyperedge configuration checking and weight diversity. Sci China Inf Sci, 2017, 60: 062103
21 Mavrovouniotis M, Muller F M, Yang S. Ant colony optimization with local search for dynamic traveling salesman problems. IEEE Trans Cybern, 2017, 47: 1743–1756
22 Luo C, Xing W Q, Cai S W, et al. NuSC: an effective local search algorithm for solving the set covering problem. IEEE Trans Cybern, 2024, 54: 1403–1416
23 Cai S W, Luo C, Lin J K, et al. New local search methods for partial MaxSAT. Artif Intell, 2016, 240: 1–18
24 Cai S W, Lei Z D. Old techniques in new ways: clause weighting, unit propagation and hybridization for maximum satisfiability. Artif Intell, 2020, 287: 103354
25 Luo C, Cai S W, Su K L, et al. Clause states based configuration checking in local search for satisfiability. IEEE Trans Cybern, 2015, 45: 1028–1041
26 Luo C, Su K L, Cai S W. Improving local search for random 3-SAT using quantitative configuration checking. In: Proceedings of the 20th European Conference on Artificial Intelligence, 2012. 570–575
27 Luo C, Cai S W, Wu W, et al. Focused random walk with configuration checking and break minimum for satisfiability. In: Proceedings of the International Conference on Principles and Practice of Constraint Programming, 2013. 481–496
28 Luo C, Cai S W, Wu W, et al. Double configuration checking in stochastic local search for satisfiability. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence, 2014. 2703–2709
29 Luo C, Su K L, Cai S W. More efficient two-mode stochastic local search for random 3-satisfiability. Appl Intell, 2014, 41: 665–680
30 Cai S, Luo C, Su K. CCAnr: a configuration checking based local search solver for non-random satisfiability. In: Proceedings of the International Conference on Theory and Applications of Satisfiability Testing, 2015. 1–8
31 Luo C, Cai S W, Wu W, et al. CCLS: an efficient local search algorithm for weighted maximum satisfiability. IEEE Trans Comput, 2015, 64: 1830–1843
32 Chu Y, Luo C, Huang W X, et al. Hard neighboring variables based configuration checking in stochastic local search for weighted partial maximum satisfiability. In: Proceedings of the 29th International Conference on Tools with Artificial Intelligence, 2017. 139–146
33 Chu Y, Luo C, Cai S W, et al. Empirical investigation of stochastic local search for maximum satisfiability. Front Comput Sci, 2019, 13: 86–98
34 Luo C, Hoos H H, Cai S W. PbO-CCSAT: boosting local search for satisfiability using programming by optimisation. In: Proceedings of the International Conference on Parallel Problem Solving from Nature, 2020. 373–389
35 Lei Z D, Cai S W, Luo C, et al. Efficient local search for pseudo boolean optimization. In: Proceedings of the International Conference on Theory and Applications of Satisfiability Testing, 2021. 332–348
36 Cai S W, Luo C, Zhang X D, et al. Improving local search for structured SAT formulas via unit propagation based construct and cut initialization. In: Proceedings of the International Conference on Principles and Practice of Constraint Programming, 2021. 1–10
37 Chu Y, Cai S W, Luo C. NuWLS: improving local search for (weighted) partial maxsat by new weighting techniques. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2023. 3915–3923
38 Cai S W, Luo C, Zhang H C. From decimation to local search and back: a new approach to MaxSAT. In: Proceedings of the International Joint Conference on Artificial Intelligence, 2017. 571–577
39 Zhang W J, Sun Z Y, Zhu Q H, et al. NLocalSAT: boosting local search with solution prediction. In: Proceedings of the International Joint Conference on Artificial Intelligence, 2020. 1177–1183
40 LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521: 436–444
41 Reisch J, Großmann P, Kliewer N. Stable resolving-a randomized local search heuristic for MaxSAT. In: Proceedings of German Conference on Artificial Intelligence (Künstliche Intelligenz). Berlin: Springer, 2020. 163–175
42 Zheng J Z, He K, Zhou J R, et al. BandMaxSAT: a local search MaxSAT solver with multi-armed bandit. In: Proceedings of the 31st International Joint Conference on Artificial Intelligence, 2022. 1901–1907
43 Lei Z D, Cai S W, Geng F, et al. Satlike-c: solver description. MaxSAT Evaluation 2021, 2019. https://maxsat-evaluations.github.io/2021/
44 Chen J X, Lin G Q, Chen J X, et al. Towards efficient allocation of graph convolutional networks on hybrid computation-in-memory architecture. Sci China Inf Sci, 2021, 64: 160409
45 Edwards M, Xie X H. Graph convolutional neural network. In: Proceedings of the British Machine Vision Conference, 2016
46 Selsam D, Lamm M, Bünz B, et al. Learning a SAT solver from single-bit supervision. In: Proceedings of the International Conference on Learning Representations, 2019
47 Li K W, Zhang T, Wang R, et al. Deep reinforcement learning for combinatorial optimization: covering salesman problems. IEEE Trans Cybern, 2022, 52: 13142–13155
48 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Proceedings of the International Conference on Learning Representations, 2015
49 Mikolov T, Karafiát M, Burget L, et al. Recurrent neural network based language model. In: Proceedings of the Interspeech, 2010. 1045–1048
50 Xu L, Hutter F, Hoos H H, et al. SATzilla: portfolio-based algorithm selection for SAT. J Artif Intell Res, 2008, 32: 565–606
51 Liang J H, Ganesh V, Poupart P, et al. Learning rate based branching heuristic for SAT solvers. In: Proceedings of the

International Conference on Theory and Applications of Satisfiability Testing, 2016. 123–140
52 Yolcu E, Póczos B. Learning local search heuristics for Boolean satisfiability. In: Proceedings of the International Conference on Neural Information Processing Systems, 2019. 7990–8001
53 Sadeg S, Hamdad L, Chettab H, et al. Feature selection based bee swarm meta-heuristic approach for combinatorial optimisation problems: a case-study on MaxSAT. Memetic Comp, 2020, 12: 283–298
54 Shi F, Lee C H, Bashar M K, et al. Transformer-based machine learning for fast SAT solvers and logic synthesis. 2021. ArXiv:2107.07116
55 Li Z W, Chen Q F, Koltun V. Combinatorial optimization with graph convolutional networks and guided tree search. In: Proceedings of the International Conference on Neural Information Processing Systems, 2018. 537–546
56 Liang J H, Oh C, Mathew M, et al. Machine learning-based restart policy for CDCL SAT solvers. In: Proceedings of the International Conference on Theory and Applications of Satisfiability Testing, 2018. 94–110
57 Selsam D, Bjørner N. Guiding high-performance sat solvers with unsat-core predictions. In: Proceedings of the International Conference on Theory and Applications of Satisfiability Testing. Berlin: Springer, 2019. 336–353
58 Schlichtkrull M S, Kipf T N, Bloem P, et al. Modeling relational data with graph convolutional networks. In: Proceedings of the European Semantic Web Conference, 2018. 593–607
59 Li Y J, Tarlow D, Brockschmidt M, et al. Gated graph sequence neural networks. In: Proceedings of the International Conference on Learning Representations, 2016
60 Wu W, Li B, Chen L, et al. Efficient attributed network embedding via recursive randomized hashing. In: Proceedings of the International Joint Conference on Artificial Intelligence, 2018. 2861–2867
61 Biere A. CaDiCal, Lingeling, Plingeling, Treengeling and YalSAT entering the SAT Competition 2018. In: Proceedings of the SAT Competition, 2017. 14
62 Biere A, Fazekas K, Fleury M, et al. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In: Proceedings of the SAT Competition, 2020. 51–53