Machine Learning Model for Selecting Assignments of Variables for SAT Problems

Jonathan Oliva
Department of Computer Science, University of York, United Kingdom

Peter Nightingale
Department of Computer Science, University of York, United Kingdom

Abstract

Machine learning (ML) methodologies have been evolving as they have been applied in different fields. One area of application is Boolean Satisfiability problems (SAT problems), which have importance in industry, such as machinery scheduling in assembly lines, or timing analysis in digital circuits reducing costs and optimizing resource production.

In this field, one of the trending explorations of ML application have been prediction of satisfiability, where a ML model generalizes whether a problem is satisfiable, that is, it has a solution. However, despite advances in this practice, the generalization of the correct assignment of variables that makes a problem satisfiable has been less explored in comparison except for UNSAT Core.

In this paper, we propose VAPS-GCN, a machine learning model to aid in the prediction of variable assignments for SAT problems. Adapting and improving a machine learning model for abstract argumentation to work on SAT problems, representing them as Boolean expressions. The process of the proposed model starts with converting a problem from a CNF format into a heterogeneous graph representation of SAT problem components. We propose a specialized network of GCN layers for aggregation between three distinct kinds of nodes. Finally, the model produces a binary generalization of the assignments of the variables from which the most probable are chosen to be part of the solution of the SAT problem. Moreover, the model is trained to solve some examples of combinatorial problems. Training the model first on available data of random problems and then continuing a method of fine-tuning for when provided data for a specific kind of combinatorial problem is insufficient.

The model produces a prediction of the assignment of each variable, combined with its confidence (based in a probability calculation). The output of the model may be useful in several ways, such as warm-starting solvers or deciding on some variables to assign prior to running a classical solver. In this paper, we evaluate the model on a MAXSAT problem. We fine-tune the model for the problem class, then use it to pre-assign some of the variables prior to applying a classical MaxSAT solver. We show substantial performance improvements with a very small loss of optimality.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming

Keywords and phrases Machine Learning, Artificial Intelligence, Graph Neural Network, Boolean Satisfiability Problems, Combinatorial Problems

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Peter Nightingale*: Partially supported by UK EPSRC grant EP/W001977/1

1 Introduction

The motivation of our model, Variable Assignment Prediction for Satisfiability using GCN (VAPS-GCN), is the limited machine learning methodologies that have been developed to be applied to generalize the assignments of variables present in an SAT problem, except for the variables that are part of the UNSAT core. There are several approaches to help with the process of solving SAT problems, for example, end-to-end machine learning models such as NeuroSAT[16], and DG-DAGRNN[2] where they focus on predicting the satisfiability for SAT problems. Information about satisfiability that can be used by traditional solvers to assist in


© Jonathan Oliva and Peter Nightingale;
licensed under Creative Commons License CC-BY 4.0

reducing time and costs. Approaches such as SATformer[17] to predict satisfiability and the UNSAT core composed of conflictive clauses and their literals. A core that causes a problem to be unsatisfiable. Our research is broadly concerned with applying machine learning in solving constraints satisfaction problems, by using Boolean satisfiability and conjunctive normal form to define the problems. Reducing the search space for the assignment of variables by pruning the most predictable values. In this and the next section, a brief introduction and an overall explanation are extended.

1.1 Constraint Satisfaction Problems

The Constraint Satisfaction Problem (CSP) consists of a problem composed of a finite set of variables, each of which is associated with a finite domain, and a set of constraints that restricts the values which variables can simultaneously take [18]. Problems which are not only present in Artificial Intelligence (AI) but also being applied in other parts of the industry, to manage processes and scheduling, for instance, determining the best series of activation which machines must follow in the assembly of a product. Minimizing costs associated with energy and time in the processing of fabrication.

1.2 Machine Learning Applied to SAT problems

Solving Boolean Satisfiability problems (SAT problems) raises some challenges to research. One of the most common difficulties is the amount variables presented in a problem, it increases costs associated with solving it, for instance, time. However, the capacity of machine learning(ML) models to learn patterns present in a SAT problem, it is factor that has brought attention of researches in the CSP field. Several strategies have been developed in turn to explore its advantages and strengths to counter the challenges in SAT problems [5]. Different problems have been addressed with machine learning models, for instance a ML model to give a prediction of the satisfiability of a problem or predicting the clauses and their literals that cause a problem to be unsatisfiable (i.e., predicting a UNSAT core) [6].

Different branches of ML approaches have been applied and developed to offer aid or solution to solving SAT problems. Examples such as QuerySAT[14] to determine a case as satisfiable by finding first the correct assignment of variables, then declaring a problem as unsatisfiable when not finding them. An example of aiding is Conflict-driven clause learning (CDCL) solvers such as NeuroCore [15] proposed for branching heuristics, an ML alternative to detect variable assignment that leads toward an unsatisfiable core in MiniSAT a classical SAT solver.

2 Background and Related Work

In this section, we present an outline of the work in the general area of machine learning for SAT solving. Mention some ML-based approaches to aid in SAT solving. Particularly those with a similar approach to predict assignments for variables. This section also includes some ML models for maximum satisfiability problems.

2.1 Preliminaries

Some concepts are to be considered in this paper. For starters, Conjunctive Normal Form (CNF), consists of a conjunction of clauses, where each clause is a disjunction of literals (a variable or its negation) [16].

Another concept is the Boolean Satisfiability problem (SAT problem) which aims to determine whether there exists a way of variable assignment of $TRUE$ and $FALSE$ values so that a given Boolean formula evaluates to $TRUE$. If it succeeds, then the formula is called satisfiable [6].

Other concepts or research relevant to this paper are works based on Argumentation Frameworks (AF) [4]. As a brief description, Argumentation frameworks are directed graphs where the direction of the edges represents an attack, a relationship of conflict against the nodes that are the arguments, each directed edge representing one argument attacking another. An argument to be accepted must be in a given AF, following a detailed semantic [4]. It is possible to represent as a constraint satisfaction problem the task regarding the determination of acceptable arguments.

2.2 Machine Learning Approaches to SAT

In recent papers ML-based for SAT problem, there is a work that tends to be mentioned regarding satisfiability prediction, NeuroSAT[16]. Model based on the message passing strategy to communicate and update nodes, which are clauses and literals of an SAT problem. A model trained in random problems, it predicts if a problem is satisfiable. It offers a partial strategy of obtaining the assignment of variables. However, the work is more focused on the prediction of satisfiability.

An interesting work is DG-DAGRNN[2], an ML approach to solve a kind of SAT problem, the Circuit-SAT. It focuses on discovering the solution to an SAT problem, once found, the model classifies a problem as satisfiable. This strategy avoids false positive generated by the models. However, the model design is focused on a particular kind of SAT problem, making it difficult to adapt to others. Using a similar strategy is QuerySAT [14], it was designed to predict the correct assignment for the variables, once it found a solution, it checks whether the assignments produced a satisfiable state in the problem.

An example of clauses prediction is SATformer [17], it is a model based on attention networks to solve SAT problems. However, its focus is on UNSAT problems. This model predicts the most probable clauses and variables to be part of the UNSAT core.

An example showing the potential of ML approaches is MapleSSV [12] from the Maple series. An SAT solver that combines a semi-classical SAT solver structure with an implementation of ML algorithms to perform optimization heuristics to solve SAT instances.

2.3 Other Works of Importance

A notable approach for this research is learning to prune for constraint programming (CP-LTP) [1]. Learning to Prune (LTP) is primarily for optimization problems. In LTP, a ML model is used to predict a set of variable assignments (for a subset of the variables) that are expected to be part of an optimal solution. The predicted assignments are then applied to the problem instance for later being solved with a conventional solver. The goal is to improve the runtime performance of the conventional solver with minimal or no impact on the optimality of the solution found. The CP-LTP approach translates a constraint model into a graph representation, extracting structure of the problem, creating features by variables, using an ML classifier to estimate probabilities. We also focus on using LTP methodology to predict parts of a large problem instance, looking to get an optimal value in less processing time, and with the less big gap against the real optimal value of the problem. A crucial difference is that the mentioned CP-LTP approach is a general framework that lets us utilize LTP methodology on problems expressed in a CP modeling language while our ML model does

not work directly with CP modeling but instead predicts SAT instances presented in CNF contained in DIMACS files.

An example in a similar area is partial maximum satisfiability problems. InitPMS [10] is proposed to work with local search solvers by aiding in the initialization of variables as an alternative to random initialization. An ML approach to predict assignments. Nevertheless, these strategies can generate error values for the assignments; therefore, it needs a way to filter the less probable predictions.

Our model was based on the approach of AFGCN [11], a ML model that depicts an end-to-end solver for resolution of AF for the field of Abstract Argumentation (AA). ML model based on the application of several layers of GCN, of the work of Kipf and Welling [7]. Nevertheless, it is important to mention some key differences. AFGCN was designed to work on a homogeneous graph, which is a graph with only one kind of node while VAPS-GCN model was designed to work on a heterogeneous graph, which is multi type of nodes. Therefore, adapting different layers of the model to work on multi node type of data. Additionally, an increase in the number of edges between nodes, and distinct kinds of edges represent the different relationships between different type of nodes. Including a clear separation of node features in the input data.

3 Predicting Assignments with a Machine Learning Model

The main part of the machine learning model approach is AFGCN[11] as a methodology to predict assignments. However, AFGCN was based on a homogeneous graph while for SAT problems was used a heterogeneous graph. Therefore, several adaptations and changes were used. The architecture and redesign of the process is presented in Figure. 2

The work starts by using CSP instances converted to SAT in CNF. They are introduced into the framework to be processed by the model. As SAT instances, they are converted into graph representations following a heterogeneous graph scheme called LCG*, a remarkable scheme described in the data generator of G4SATBench [9]. Utilizing clauses and literals to create three kinds of nodes. That is, clauses, literals, and the negation of literals, with relation between them represented as edges, as showed in Figure. 1. Focusing on avoiding loss of information regarding the relationship between literals and their negation.

For graph construction, ML model layers and information preprocessing, the Deep Graph Library (DGL)[19] was used. DGL is a library dedicated to extending and grouping in a

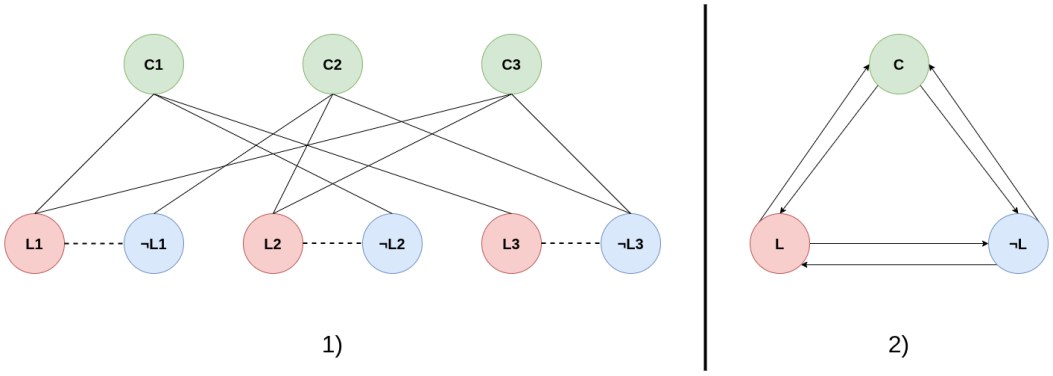

Figure 1 To the left the LCG* scheme using as example a SAT instance; C1: (L1, -L2, L3), C2: (-L1, L2, -L3), C3: (L1, L2, -L3). To the right the graph construction using DGL library.

framework several tools and modules to help in the implementation and designing of graph neural network models.

For each node in the graph several metrics from the area of graph network analysis, for instance, eigenvector and closeness centrality, are used to categorize the nodes and the relationships that they exhibit in the connections between nodes in the graph scheme generated from a respective SAT problem. Separating the nodes regarding three types: clauses, literals, and negation of literals. Adding some extra categorization to differentiate the node regarding the type.

Each instance in the dataset is composed of the extracted input features described in this subsection and the LCG* graph created with DGL for each problem instance as described in the previous subsection. The extracted features are composed of three groups, a dictionary where a keyword is a node type, and the data or value is the list of features extracted for each node in the node type. However, labels or solutions for problem instances are calculated then stored as hash to train the model. Furthermore, there are cases when multiple labels are created by problem. Methodology used because SAT problems can have more than one solution. This methodology is applied in random problems and in the resource-constrained project scheduling problem.

The VAPS-GCN model processes the data. The input features are submitted to an embedding layer to capture the behavior exhibit while the graphs created are to be used in the GCN layers presented in the model.

An important aspect to explain is the GCN layers and their algorithm used in this model. Originally, GCN layers are designed to perform on homogeneous graphs. It outputs vectors representing the relevance and behavior that exist between the nodes and their neighborhood. Following the next definition of their propagation rule:

$$f(H^{(l)}, A) = \sigma(\widehat{D}^{-1/2}\widehat{A}\widehat{D}^{-1/2}H^{(l)}W^{(l)})$$

Where A is the adjacency matrix, $\widehat{A}$ is the result of adding self-loops represented as an identity matrix to the adjacency matrix. $H^{(l)}$ are input features of layer l, and $W^{(l)}$ is the weight matrix of layer l. The $\widehat{D}^{-1/2}$ are a symmetric normalization of the rows and columns of the adjacency matrix. σ is an activation function applied to the layer [11][7].

GCN layers are notable for capturing relevant information from nodes when dealing with homogeneous graphs. However, the VAPS-GCN model is oriented to manage the information from the nodes of heterogeneous graphs. Using an innovative design to capture heterogeneous behavior of the data, the model is composed of four modules to compute convolutions between heterogeneous graphs, known as hetero convolutions. A module for hetero convolution works by managing the node-edge-node relationship, a full description conformed by source node to destination node on a specific edge type. Description used as a key to separate the submodules or ML layers applied to the specific node-edge-node relationship. In this case, using GCN layers as submodules as presented in Figure 2.

A hetero convolution layer operates by reading the features of the source nodes, updating a state to be later sent to the destination nodes. In addition, if there are multiple edges with the same destination node, they are aggregated together with a method specified by the user in its declaration. In the case of this project, it is the sum method. Following this process, each hetero convolution layer generates a new state for the respective node features.

After the four blocks composed by a hetero convolution layer, a ReLU function layer, and a dropout layer, the model architecture continues toward a linear function. Then it passes through a process to give it a form from where it is possible to calculate the loss and the selection of assignments. Using a Sigmoid output layer to generate probability of Boolean true or false for the respective literals, their negations, and clauses of the SAT problem.

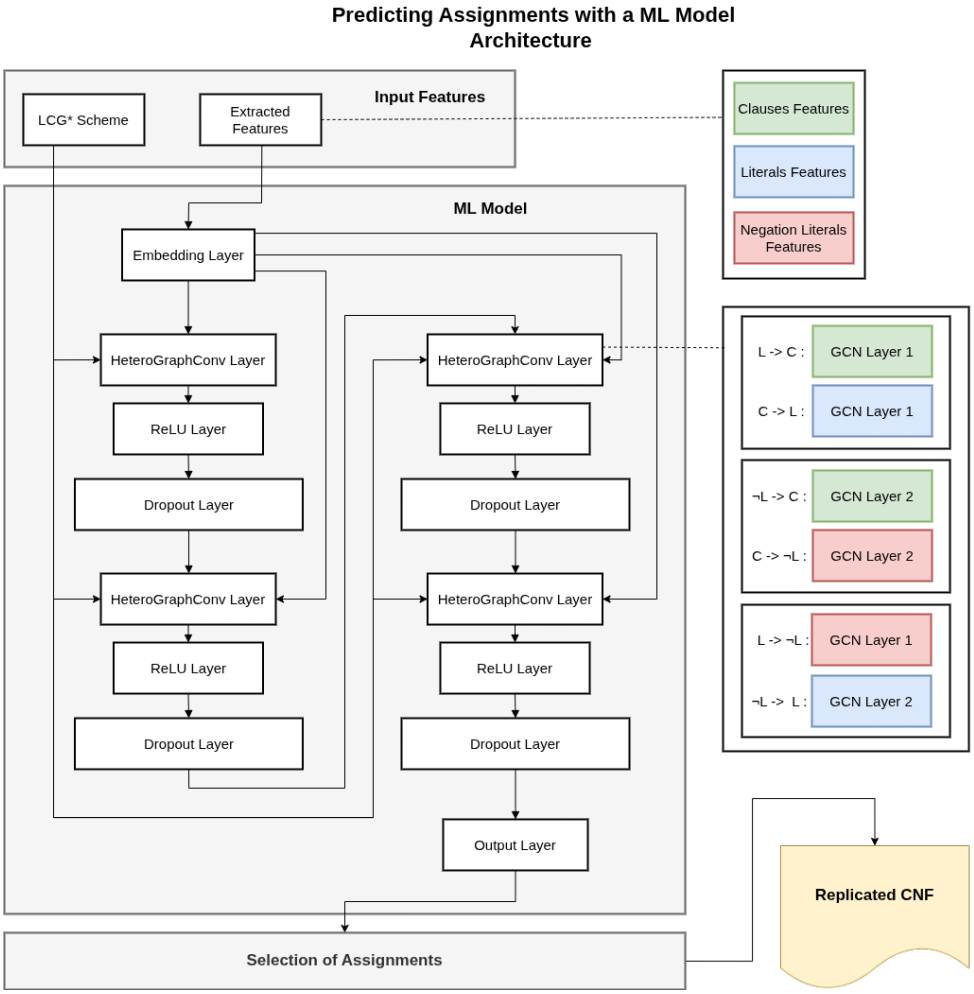

Figure 2 Machine learning design model and features. L represent the literals, C the clauses, and ¬L the negation of literals.

Selecting those predictions where a literal classified as binary one, while its negation should be binary zero, or the inverse. The objective is to find a separation in probabilities between the predictions, where they must aim toward opposite bounds between them such as binary one and zero. The higher the separation, the more confident the prediction is.

Finally, a threshold is used to separate the more confident assignments. Generating a replication of the SAT problem as a new SAT instance. Adding some unary clauses to assign literals corresponding to the confident predictions.

4 Experimental Evaluations

In this section, the problems to be used to apply the model are described, together with the results of using the model already trained to prune them. Furthermore, the chosen metric to be used to measure the model's overall performance is the Matthews Correlation Coefficient (MCC). A balanced metric for binary data that works well when overseeing unbalanced datasets.

The VAPS-GCN model was trained using a fine-tuning approach, in which the model initially learns with general and more available data. Then in a later time, this pretrained model continues its learning training with objective but scarcer data, a smaller dataset with a specific kind of SAT problem. Being VAPS-GCN model to be pretrained first in random problems. Then the model is trained in objective data, in this research, it will be the resource constraint project scheduling problem instances. Both cases of training using Adam as optimizer, and binary cross-entropy as loss function.

In addition, a balancing was performed over the labels of the dataset. The loss method was balanced with the Boolean $TRUE$ and $FALSE$ assignments of the real solutions present in the list of probable assignments for a problem, considering that they are in CNF. It was done with the intention of avoiding fitting problems caused by having a binary dataset with more $TRUE$ than $FALSE$ values, or the other way around.

An important mention is the training methodology. Using predicted assignments and a list of the real labels, correct solutions, to train the model. Here, each solution is a vector composed of ordered assignments toward a satisfiable solution. Each element of the list is a probable assignment of the literals for an SAT problem in CNF. Then, the most similar solution vector is used to compute the loss. The number of solutions in the label list for both problems in this research is twenty.

The purpose is to let the model detect a prediction based on its understanding of the structure of the problem. Once a path has been detected, train the model to improve its prediction. Learning to identify the general structure of an SAT problem and directing the model toward solutions found. This also works as an approach when handling SAT problems with more than one solution. This approach used with the two kinds of problems mentioned in the next subsections.

4.1 Random Problems

The initial data used to train the model were random problems generated by the SR generator present in the G4SATBench framework [9]. The generated random problems do not have a defined internal structure compared to real-world problems. However, they can be used to guide the model in learning the dynamics of a SAT problem in CNF. Their design can be a challenge to resolve to ML models. So, they are usually used to assert model performance.

Generating a dataset, then separating it into 50000 and 10000 satisfiable problems for the training and validation sets, respectively. Each problem has between 40 and 100 variables. After training the model over 250 epochs, and with a batch of 300, the validation was measured, producing a value of 46.10% for the mean MCC of the best epoch obtained in the model, and an accuracy generated on the predictions of the same set got a value of 73.05%.

4.1.1 Resource Constrained Project Scheduling Problem

Resource-constrained project scheduling problems (RCPSP) are combinatorial optimization problems where the objective is to find an optimal organization of activities that can use the resources associated with their operation in a small interval of time and cost-efficient. Activities with a known processing time or duration and a number and kind of resources to operate. Activities can follow a sequence, a precedence relation between pair of activities to operate. Furthermore, there are distinct types of resources; their availability responds to the demands of the activities for their operation. The problem looks for a schedule of the activities with the shortest duration that respects resource availability and the precedence relations between activities.[3].

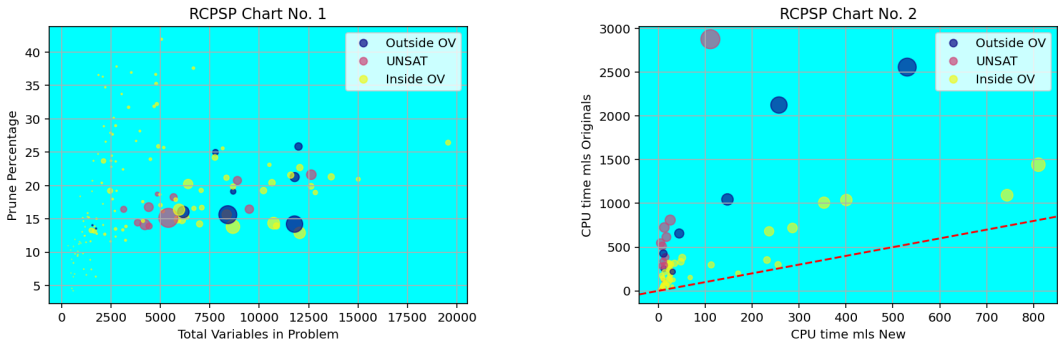

Figure 3 To the left a chart reflecting the level of pruning. To the right a chart reflecting the speed increase in comparison with the original. The size of the circles indicates the hardness, that is, time used by the MAXSAT solver to find the solution for the instance without the VAPS-GCN model. Both charts display distinct types of new solution found after using the ML model: Outside Optimal Value (OV), UNSAT, and Inside Optimal Value (OV)

The RCPSP instances used in this research paper are satisfiable and originally obtained from PSPLIB[8]. After Savile Row[13] a constraint programming modeling assistant was used to generate DIMACS files, the WMaxCDCL2024 MAXSAT solver was applied with twenty different random seeds to add some level of randomness to the labels of the training and validation sets. Using only the hard clauses in the construction of the graphs used in the VAPS-GCN model.

The results of the experiment are presented in Figure 3. Taking into consideration a hard instance as the time that a MAXSAT solver needs to be used to find a solution without the application of the VAPS-GCN model. There is a clear indication of improving speed in solving MAXSAT problems, in most cases a decrease around 50% in solving time (CPU time), including some of the most difficult instances when applying our ML model. In the pruning area, despite changes in the number of variables, we found that most of the pruning of hard instances oscillate between 12% and 25%. Moreover, when measuring the optimal makespan between the original and the new instances, an average percentage gap of 0.30% was found, with a 0.0% as median.

However, the decrease in CPU time to solve an instance, and solutions generated by the solver when using our VAPS-GCN model tend to be outside of the optimal values generated by the solver, or become UNSAT when handling hard instances. Moreover, only the less hard instances (less than around 500 milliseconds original CPU time) tend to be inside the original optimal value. Even if their solving time decreased around 50%.

5 Conclusions

LTP offers a very promising approach to reduce CPU time in solving combinatorial problems such as the RCPSP. The application of ML model methodologies generated a solution faster than using the MAXSAT solver uniquely. It found solutions for most of the problems. However, it still needs to improve on recognizing assignments of variables that are part of the optimal solution for the hardest instances. However, it is a fact that solutions were found for most of the problem instances and the small optimal makespan gap offers an opportunity to improve. Improving methodologies could include using some strategies for unit propagation. Also, working with critical information like the variables in the constraint programming model. Including soft clauses in the processing of the VAPS-GCN model.

References

1. Deepak Ajwani, Peter Nightingale, and Felix Ulrich-Oltean. Generalizing learning-to-prune for constraint programming. In *Proceedings of the 23rd workshop on Constraint Modelling and Reformulation (ModRef 2024)*, 2024. URL: `https://modref.github.io/ModRef2024.html`.
2. Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Learning to solve circuit-sat: An unsupervised differentiable approach. In *International Conference on Learning Representations*, 2018.
3. Christian Artigues, Sophie Demassey, and Emmanuel Neron. *Resource-constrained project scheduling: models, algorithms, extensions and applications*. John Wiley & Sons, 2013.
4. Günther Charwat, Wolfgang Dvořák, Sarah A Gaggl, Johannes P Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation–a survey. *Artificial intelligence*, 220:28–63, 2015.
5. Mohammad Hossein Fazel Zarandi, Ali Akbar Sadat Asl, Shahabeddin Sotudian, and Oscar Castillo. A state of the art review of intelligent scheduling. *Artificial Intelligence Review*, 53(1):501–593, 2020.
6. Wenxuan Guo, Hui-Ling Zhen, Xijun Li, Wanqian Luo, Mingxuan Yuan, Yaohui Jin, and Junchi Yan. Machine learning methods in solving the boolean satisfiability problem. *Machine Intelligence Research*, 20(5):640–655, 2023.
7. Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
8. Rainer Kolisch and Arno Sprecher. Psplib-a project scheduling problem library: Or software-orsep operations research software exchange program. *European journal of operational research*, 96(1):205–216, 1997.
9. Zhaoyu Li, Jinpei Guo, and Xujie Si. G4satbench: Benchmarking and advancing sat solving with graph neural networks. *arXiv preprint arXiv:2309.16941*, 2023.
10. Chanjuan Liu, Guangyuan Liu, Chuan Luo, Shaowei Cai, Zhendong Lei, Wenjie Zhang, Yi Chu, and Guojing Zhang. Optimizing local search-based partial maxsat solving via initial assignment prediction. *Science China Information Sciences*, 68(2):1–15, 2025.
11. Lars Malmqvist. *Approximate solutions to abstract argumentation problems using graph neural networks*. PhD thesis, University of York, 2022.
12. Saeed Nejati, Md Solimul Chowdhury, and Vijay Ganesh. Maplessv sat solver for sat competition 2021. *SAT COMPETITION 2021*, page 35, 2021.
13. Peter Nightingale. Savile row manual. *arXiv preprint arXiv:2201.03472*, 2021.
14. Emils Ozolins, Karlis Freivalds, Andis Draguns, Eliza Gaile, Ronalds Zakovskis, and Sergejs Kozlovics. Goal-aware neural sat solver. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.
15. Daniel Selsam and Nikolaj Bjørner. Guiding high-performance sat solvers with unsat-core predictions. In *Theory and Applications of Satisfiability Testing–SAT 2019: 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9–12, 2019, Proceedings 22*, pages 336–353. Springer, 2019.
16. Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L Dill. Learning a sat solver from single-bit supervision. *arXiv preprint arXiv:1802.03685*, 2018.
17. Zhengyuan Shi, Min Li, Yi Liu, Sadaf Khan, Junhua Huang, Hui-Ling Zhen, Mingxuan Yuan, and Qiang Xu. Satformer: transformer-based unsat core learning. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–4. IEEE, 2023.
18. E. Tsang. *Foundations of Constraint Satisfaction*. Computation in cognitive science. Academic Press, 1993. URL: `https://books.google.co.uk/books?id=TnxQAAAAMAAJ`.
19. Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.