Karlsruher Institut für Technologie

Bachelor Thesis

Explainable Solver Selection for MaxSAT

Salomo Hummel

Date: 12. März 2025

Supervisors: Prof. Dr. Peter Sanders
Dr. Markus Iser

Institute of Theoretical Informatics, Algorithm Engineering
Department of Informatics
Karlsruhe Institute of Technology

Abstract

Maximum Boolean Satisfiability (MaxSAT) is a central NP-hard optimization problem with wide-ranging applications. In this work, we address the challenge of algorithm selection for MaxSAT by developing and comparing multiple prediction models designed to narrow the performance gap between the Single Best Solver (SBS) and the theoretical Virtual Best Solver (VBS). We propose grouping based models including Family Based Prediction (FP) and Clustering Based Prediction (CP), as well as their hierarchical extensions (FHP and CHP) that employ dedicated random forests for each group of instances. In addition, we examine the Direct Solver Predictor (DSP) model, which is not based on grouping. All models are evaluated on a dataset comprising 13,742 weighted CNF instances from the Global Benchmark Database, with performance measured through k-fold cross-validation and a range of metrics including PAR-2 score, prediction accuracy, gap closure, and Matthews Correlation Coefficient. Our experiments reveal that while grouping by families or clusters achieves high prediction accuracy when ample training data is available, only clustering significantly enhances the discriminative ability of the predictors in the face of unseen instance families. Notably, the Direct Solver Predictor model consistently achieves robust performance and closes the performance gap by up to 73%.

Acknowledgments

An dieser Stelle möchte ich mich herzlich bei allen Personen bedanken, die mich während der Bachelorarbeit unterstützt haben. Mein besonderer Dank gilt meinem Betreuer Markus Iser, nach dessen Meetings ich stets wieder Licht am Ende des Tunnels sah. Ebenso danke ich meinen Freunden, die mir immer zur Seite standen und ebenfalls für viel Motivation gesorgt haben.

Use of Generative AI

In dieser Arbeit wurde generative KI eingesetzt, um Code zu genrieren, die Rechtschreibung, Grammatik und den Stil zu korrigieren sowie die relevante Literatur besser zu verstehen (z. B. durch das Generieren von Zusammenfassungen und anschließenden Fragen). Dabei sind jedoch keine Informationen oder Ideen in die Arbeit eingeflossen, die nicht zuvor auf ihre Echtheit in der Originalquelle überprüft worden sind. Insbesondere wurde generative KI nicht für die Literaturrecherche und auch nicht zur Generierung von ganzen Abschnitten oder Kernaussagen sowie anderen thematischen Schwerpunkten eingesetzt. Es sind folgende Modelle verwendet worden: GitHub Copilot, ChatGPT mit GPT-4o, o1, o3 Mini und Gemini mit 2.0 Flash und 1.5 Pro.

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen, als die angegebenen Quellen und Hilfsmittel benutzt, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Karlsruhe, den 12.03. 2025

Contents

1 Introduction

1.1 Motivation

Maximum Boolean Satisfiability (MaxSAT) is a fundamental problem in computer science, with applications in various domains, including artificial intelligence, software verification, and circuit design. To be able to solve the MaxSAT problem, many MaxSAT solvers were developed, such as branch and bound solvers, SAT-based solvers, or integer linear programming (ILP) solvers. However, while all solvers have instances of the MaxSAT problem they excel at, each of them has instances with deplorable runtime behavior. This results in a significant performance gap between the virtual best solver (VBS) and the single best solver (SBS). Thus, we face the algorithm selection problem (Rice, 1976), where we want to choose the best solver for a new, previously unknown instance of the MaxSAT problem. We present and compare several approaches to MaxSAT solver selection, including family-based and clustering-based approaches. A key motivation for investigating family-based prediction models is their explainability. Analyzing solver performance at the family level might provide valuable insights into specific instance characteristics for solver development and solver configuration, thereby informing targeted improvements and optimizations.

Central questions are:

- How do different prediction models compare in terms of accurately selecting the best MaxSAT solver for a given instance?
- By how much can we close the gap between the single best solver and the virtual best solver?
- Do families or clusters help for building better prediction models?

1.2 Contribution

This thesis makes several contributions to the field of algorithm selection for Maximum Boolean Satisfiability (MaxSAT), addressing the performance gap between the Single Best Solver (SBS) and the Virtual Best Solver (VBS).

We introduce several prediction models for solver selections, namely: Family Based Prediction (FP), Clustering Based Prediction (CP), Family Based Hierarchical Prediction (FHP), Clustering Based Hierarchical Prediction (CHP) and Direct Solver Prediction (DSP). FP

and CP leverage instance-grouping, either from pre-assigned family labels or via unsupervised clustering, to map problem instances to the solver that exhibits the best performance on similar instances. FHP und CHP further refine FP and CP by using group-specific prediction models resulting in a hierarchical structure. DSP uses a single, robust random forest classifier to directly select the optimal solver based solely on instance features. Despite its simplicity, this method demonstrates comparable performance to FHP and CHP, especially in scenarios where new or unseen instance families appear. Using a dataset of 13,742 weighted conjunctive normal form (WCNF) instances from the Global Benchmark Database (GBD) we conduct extensive experiments using cross-validation and multiple performance metrics (e.g. gap closure, PAR-2 score, accuracy, and Matthews Correlation Coefficient) to validate our models. We demonstrate that although family-based approaches can yield high prediction accuracy when training data is abundant, they sometimes suffer when faced with novel families. This work quantifies how cluster-based approaches and direct solver prediction can adapt more robustly under such conditions, thereby narrowing the gap between SBS and VBS by up to 73%.

In summary, this thesis provides a comprehensive study of several proposed algorithm selection models for MaxSAT and bridges the gap between SBS and VBS.

1.3 Structure of Thesis

Chapter 2 presents related work in the field of algorithm selection and the closely related field of algorithm configuration for SAT and MaxSAT. We also present one work that extensively explores clustering of SAT instances that motivated the clustering-based approaches in our work.

Chapter 3 introduces the MaxSAT problem and the concepts of MaxSAT solvers and algorithm selection. We also summarize the fundamentals of clustering and classification algorithms, which are essential for understanding the subsequently proposed algorithm selection models.

Chapter 4 introduces five algorithm selection models. We describe each model in detail and motivate how it is intended to close the performance gap between the Single Best Solver (SBS) and the Virtual Best Solver (VBS).

Chapter 5 details our experimental setup including dataset selection and a description of the used evaluation metrics.

Chapter 6 presents the empirical results and provides a comparative analysis of the prediction models. By contrasting model performance against SBS and VBS baselines, this chapter informs how effectively our proposed techniques compare in solver selection accuracy and runtime efficiency.

Finally, Chapter 7 summarizes the main findings and outlines avenues for future research.

2 Related Work

Related work mainly falls into two categories. First, Algorithm Selection, where only the problem of selecting the best algorithm for a given instance is faced. The second is algorithm configuration, where the goal is to find the configurations of an algorithm that make up a good portfolio. These are often coupled with a subsequent selection of the best configuration for a given instance and thus being closely related to this work. In the third subsection, we have a look at Cluster Analysis for SAT instances which motivates the Clustering Based Predictor (cf. Section 4.3).

2.1 Algorithm Selection

An algorithm selector usually involves two phases. First is the offline training phase, and second is the online execution phase. The training phase is performed once, e.g., before a competition. A set of solvers and a set of features is determined. Then, a model is trained to accurately select the best performing solver based on the extracted features of the instances. In the execution phase, the algorithm selector calculates the features of a given problem instance, which are then input into the trained model to select the solver expected to perform best.

SATzilla uses a ridge regression method, called the empirical hardness model, for each solver in the portfolio to predict the runtime for a given instance. In the execution phase, a pre-determined set of solvers called pre-solvers is run on the given instance while the feature values are computed. If the instance was not easy and thus could not be solved by one of the pre-solvers, the empirical hardness models are used to predict the runtime for each solver in the portfolio. The solver with the lowest predicted runtime is then executed on the input instance. (Xu et al., 2008)

Matos is a MaxSAT algorithm portfolio inspired by SAT*zilla*. The prediction models used are based on ridge regression. For each solver in the portfolio, such a model is trained to predict the runtime of the solver for a given instance. In the execution phase, the solver with the lowest predicted runtime is chosen. The portfolio consists of three hand-selected solvers. (Matos et al., 2008)

CSHC selects solvers based on a cost-sensitive hierarchical clustering model. In the execution phase, the first 10% of available time a static schedule is executed. If no solution is found, the features of the instances are calculated with which the cluster of the given instance is determined. The best-performing solver on that cluster is selected to solve the instance. (Malitsky et al., 2013)

2.2 Algorithm Configuration

ISAC In the training phase, instances are clustered into c clusters using g-means on a normallized feature vector. Then the GGA tuning algorithm is used to find favorable parameters for each cluster, resulting in c algorithms. For the execution phase an instance is passed to ISAC which is assigned one of the c clusters of the training phase. The corresponding algorithm is used to solve the instance. (Kadioglu et al., 2010)

ISAC+ is an augmentation of ISAC where the training phase stays the same, however, the execution phase is replaced by the algorithm selector CSHC (Malitsky et al., 2013) choosing one of the c algorithms generated in the training phase. It is also tested on MaxSAT instances. (Ansótegui et al., 2014)

SNNAP is another augmetation of ISAC. For the training phase it adds the feature 'runtime' to the feature space. Also an additional model is trained which predicts the runtime for a given instance. In the execution phase this additional model is used to predict the runtime feature. Now the standard execution phase of ISAC is applied. (Collautti et al., 2013)

Reactive Dialectic Search Portfolios for MaxSAT (RDS) automatically adapts parameters of a metaheuristic known as Dialectic Search in real time via logistic-regression functions of eleven search statistics (e.g., elapsed time, restarts, improvements). By tuning the metaparameters of the Dialectic Search with ISAC++, this method achieves state-of-the-art performance on random and crafted instances. Combined with the industrial solver WPM3 in a portfolio, RDS took multiple gold medals in the 2016 MaxSAT Evaluation. (Ansótegui et al., 2017)

2.3 Cluster Analysis for SAT Instances

Heinen et al. (2022) explore how clustering can assist in algorithm selection for SAT problems by grouping instances with similar solver performance profiles. After extracting and preprocessing features (e.g., graph-based or gate-based statistics), they apply various clustering algorithms — most notably k-means and DBSCAN — to partition a broad collection

of SAT instances into families of "similar" problems. The core motivation is that instances clustered together might share a common solver that is both fast and stable. Key takeaways of the thesis include detailed experiments on hyperparameter tuning (like k for k-means) to obtain clusters that align well with solver efficiency, and comparisons of cluster-based solver assignments versus single best solver or virtual best solver baselines. The study reveals that many clusters do indeed encapsulate instances from the same SAT family, indicating that problem origin can correlate with solver preferences, yet also highlights exceptions where mixed-family clusters still share similar solver behavior.

3 Preliminaries

In this chapter we introduce the Maximum Boolean Satisfiability (MaxSAT) problem and the concept of MaxSAT solvers. We then explain the fundamentals of algorithm selection including the PAR-2 Score. Lastly, we introduce clustering and classification algorithms, namely constrained k-means and random forests.

3.1 Weighted Maximum Boolean Satisfiability

Maximum Boolean Satisfiability (MaxSAT) is a fundamental optimization problem in computer science, classified as an NP-hard problem. It is an extension of the Boolean Satisfiability Problem (SAT). In MaxSAT, given a Boolean formula in Conjunctive Normal Form (CNF), the objective is to determine an assignment of truth values to variables that maximizes the number of satisfied clauses or the sum of the weights of satisfied clauses in the weighted case.

Formally, let $X = \{x_1, x_2, ..., x_n\}$ be a set of Boolean variables. A literal l is either a variable x_i or its negation $\neg x_i$. A clause c is a disjunction of literals, represented as $c = (l_1 \vee l_2 \vee ... \vee l_k)$. A CNF formula F is a conjunction of clauses, expressed as $F = (c_1 \wedge c_2 \wedge ... \wedge c_m)$. In the Weighted MaxSAT problem, each clause c_i is associated with a positive weight w_i. The goal is to find a truth assignment $A : X \to \{0, 1\}$ that maximizes the total weight of satisfied clauses. Formally, the objective function is:

$$\text{maximize} \sum_{i=1}^{m} w_i \cdot A(c_i)$$

where $A(c_i) = 1$ if clause c_i is satisfied by assignment A, and $A(c_i) = 0$ otherwise. (Du et al., 1997)

3.2 MaxSAT Solvers

A MaxSAT solver aims to find a variable assignment for a given weighted MaxSAT instance that maximizes the sum of the weights of satisfied clauses. Due to the NP-hard nature of MaxSAT, solvers utilize diverse heuristics and algorithms to efficiently find optimal

or near-optimal solutions. Solver performance is highly dependent on instance characteristics, motivating the development of algorithm selection techniques.

There are three primary categories of MaxSAT solvers:

Branch and Bound Solvers: Branch and bound solvers systematically explore the solution space by branching on variables. They maintain upper and lower bounds on the solution, allowing them to prune unpromising branches that cannot improve upon the current best solution. While traditionally less competitive for MaxSAT, recent enhancements, such as incorporating clause learning (Marques-Silva et al., 2021), have improved their performance.

SAT-Based Solvers: SAT-based MaxSAT solvers capitalize on advancements in SAT solving. They transform the MaxSAT problem into a sequence of SAT instances, iteratively invoking a SAT solver for refinement. Most commonly, a Core-Guided approach is employed, which refines a lower bound by identifying and analyzing unsatisfiable cores—minimal sets of unsatisfiable clauses—to relax soft clauses on demand and thus being able to create constraints which involve a smaller set of relaxation variables.

SAT-based solvers have demonstrated strong effectiveness across diverse MaxSAT instances and are widely used in the field.

Integer Linear Programming (ILP) Solvers: MaxSAT can be formulated as an Integer Linear Programming (ILP) problem, where variables and clauses are represented as integer variables and linear constraints, respectively. The objective function maximizes the sum of the weights of satisfied clauses. While ILP solvers are a viable alternative, they generally do not match the performance of SAT-based approaches on general MaxSAT instances but can be beneficial for specific structures.

3.3 Algorithm Selection

The goal of algorithm selection is to improve the performance of solving a given problem by dynamically choosing the best solver from a set of solvers based on instance-specific characteristics. Instead of relying on a single solver, an algorithm selector employs machine learning techniques to predict which solver will perform best for a given instance. This approach is particularly useful for problems such as SAT and MaxSAT, where solver performance varies significantly across different problem instances.

Algorithm selection typically involves two distinct phases: an offline training phase and an online execution phase. In the offline training phase, performed once before deployment, the selector learns from historical data. Given a set of instances $I = \{i_1, i_2, ...\}$ of the Weighted MaxSAT problem, a set of solvers $S = \{s_1, s_2, ...\}$, and a performance metric

$m : S \times I \to \mathbb{R}$, the goal is to construct a prediction model p that maps any instance $i \in I$ to a solver $p(i) \in S$, optimizing performance according to m. To achieve this, relevant features $f(i) = (x_1, ..., x_n)^\top$ are extracted from each instance and used to train the predictive model. In the online execution phase, given a new problem instance, the selector computes its features and uses the trained model to predict and select the solver expected to perform best.

A hypothetical perfect selector termed the *Virtual Best Solver* (VBS) selects the best solver for each instance, serving as a reference point for optimal performance. The *Single Best Solver* (SBS) is the solver with the best overall performance across all instances. In practice, the goal is to bridge the gap between the SBS and the VBS. This gap represents the potential gains achievable through effective algorithm selection.

3.4 Clustering

Clustering is an unsupervised machine learning technique used to group similar data points. In algorithm selection, clustering can group instances of a problem (e.g. MaxSAT instances) that will likely be solved efficiently by the same algorithm (e.g. MaxSAT solver). In this thesis, we more specifically use the constrained k-means clustering algorithm, based on k-means, a widely used method known for its simplicity and efficiency. We will first explain the basic k-means algorithm, followed by the explanation of the constrained version.

k-means aims to partition a dataset into k clusters by minimizing the within-cluster sum of squares, also known as *inertia*. Inertia measures the internal coherence of clusters, with lower values indicating better clustering. Given a set of data points $\mathcal{X} = \{x_1, x_2, ...\}$, k-means seeks k cluster centers, $\mu_1, \mu_2, ..., \mu_k$, that minimize the following:

$$\sum_{x \in \mathcal{X}} \min_{j \in \{1,...,k\}} ||x - \mu_j||^2$$

where $||x_i - \mu_j||$ is the Euclidean distance between data point x_i and cluster center μ_j.

The k-means algorithm iteratively refines the cluster centers through the following steps:

(i) **Initialization:** k data points are randomly selected as initial cluster centers.

(ii) **Assignment:** Each data point x_i is assigned to the cluster whose center is closest to it.

(iii) **Update:** Cluster centers μ_j are recalculated as the mean of all data points in their respective clusters.

(iv) **Iteration:** Steps 2 and 3 are repeated until the cluster assignments stabilize or a maximum number of iterations is reached.

While k-means scales well to large datasets, it assumes clusters are convex and isotropic, which may not always hold. Additionally, inertia is not a normalized metric and can be

inflated in high-dimensional spaces due to the "curse of dimensionality". (scikit-learn, 2024b)

Now for the constrained k-means clustering, when applying k-means with the number of clusters $k \geq 20$ and the number of dimensions $n \geq 10$, which is the case in this thesis, we often observe empty or small clusters. We can add a constraint of minimum cluster size by reformulating the assignment step (ii) as a Minimum Cost Flow (MCF) linear network optimization problem. We will use the implementation by Levy-Kramer (2018) which is based on the constrained k-means clustering algorithm originally proposed by Bennett et al. (2000).

In this thesis, constrained k-means clustering is used in the Clustering Based Prediction (cf. Section 4.3) and Clustering Based Hierarchical Prediction (cf. Section 4.4) models to predict the best-performing solver for MaxSAT instances.

3.5 Classification

Classification is a supervised machine learning technique that assigns data points to predefined categories. In algorithm selection, classification can predict the best-performing algorithm for a given instance based on its features.

The Random Forest Classifier is the primary classification algorithm used in this work. Random forests are an ensemble learning method that constructs multiple decision trees during training. Predictions are made by aggregating the outputs of individual trees by averaging their probabilistic prediction.

Each individual tree is built from a bootstrap sample of the training set (i.e., a sample drawn with replacement). During the construction of each tree, the algorithm searches for the best split at each node by evaluating a random subset of features of size $\lfloor \sqrt{n} \rfloor$ from a total of n. These randomization techniques reduce the variance of the forest estimator and mitigate the tendency of individual decision trees to overfit.

Random forests generally offer high accuracy, robustness to noise, and the ability to handle high-dimensional data. They also provide implicit feature selection through the evaluation of feature importance during training. (scikit-learn, 2024a)

In this thesis, random forests are used in Family Based Prediction (cf. Section 4.2), Hierarchical Prediction (cf. Section 4.4) and Direct Solver Prediction (cf. Section 4.1).

4 Algorithm Selection Models

In this chapter, we present the design of multiple prediction models for algorithm selection in MaxSAT solving, which we will later evaluate in Section 6. Each model takes the feature vector of an instance as input and subsequently chooses exactly one solver from a given set of solvers.

Notation: Throughout this chapter, we denote by I the set of all instances in the training data, and by m the performance metric used by the models to optimize solver selection.

4.1 Direct Solver Prediction

We introduce the *Direct Solver Predictor* (DSP) model. DSP uses a random forest to directly predict one of the solvers from the solver set constructed in Section 6.1. The class label for an instance is the solver with the lowest runtime. If multiple solvers are the fastest, meaning they have the same runtime (e.g. when no solver can solve the instance), we choose the solver that solved the most instances on the training data.

This approach offers a valuable baseline for comparison due to its straightforward implementation. It also directly uses the way we selected our set of solvers, as a smaller set of class labels generally makes a random forest more accurate and robust.

The upper bound for performance improvement for the Solver Predictor is the VBS, which is the same as for the Hierarchical Predictors. The DSP is even simpler than the FP or CP because we use one random forest and do not require a mapping fts or cts. Still, the performance gain we can hope for is that of VBS as there is no relation to FBS. This is significantly more as depicted in Figure 6.1.

4.2 Family Based Prediction

We introduce the *Family Based Predictor* (FP) model. FP first predicts a family, then selects a solver based on that family. For family prediction, we train a random forest to predict the family of an instance. The class label for an instance is the family label as introduced in Section 5.1. During the training phase, we also construct the following mapping for a family $F \subseteq I$:

$$\mathrm{fts}(F) := \arg\min_{s \in S} m(s, F)$$

which means calculating the PAR-2 scores of all solvers on the subset of instances of the given family, then assigning the solver with the best PAR-2 score to that family. In the execution phase, given an instance, we use the random forest to predict a family F. The selected solver for that instance will then be $\mathrm{fts}(F)$.

This approach is motivated by the difference in PAR-2 score on families between solvers. In other words, one solver might be fast on instances of family A while a different solver is slow on family A but outperforms the other solvers on family B (cf. Section 6.2).

4.3 Clustering Based Prediction

We introduce the *Clustering Based Predictor* (CP) model. CP first groups instances into clusters based on their feature vectors $f = (x_1, \ldots, x_n)^\top$, and then selects a solver according to the assigned cluster. For clustering, we use constrained k-means with a minimum cluster size of n. Before clustering, we standardize the features using standard scaling, transforming each feature x_i into a standard score z_i:

$$z_i = \frac{x_i - \mu}{\sigma}$$

where $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the mean and $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$ is the standard deviation. (scikit-learn, 2024e)

The only hyperparameter we need to select is the number of clusters k (cf. Section 6.6). During training, we determine the best solver for each cluster $C \subseteq I$ by computing:

$$\mathrm{cts}(C) := \arg\min_{s \in S} \mathrm{par}_2(s, C)$$

This means we calculate the PAR-2 scores for all solvers on the instances within cluster C and assign the solver with the lowest PAR-2 score to that cluster. In the execution phase, we compute the features of a new instance, determine its cluster C, and select the solver $\mathrm{cts}(C)$ associated with that cluster.

The motivation for using clusters instead of families (as in Section 4.2) is twofold: First, the large number of different families (cf. Section 5.1) makes it challenging to train an effective prediction model, as more classes typically reduce prediction accuracy. Second, many families are underrepresented or entirely absent in the training set (cf. Section 6.2), which negatively impacts model performance. To address these issues, we remove the dependency on predefined families by clustering instances based on their features. Heinen et al. (2022) demonstrated a strong association between clusters and families for SAT instances, a finding we confirm also holds for MaxSAT (cf. Section 6.2).

4.4 Hierarchical Prediction

We introduce the *Family Based Hierarchical Predictor* (FHP) model and the *Clustering Based Hierarchical Predictor* (CHP) model. Figure 4.1 illustrates the overall architecture of the hierarchical prediction models. Each instance is first assigned a label based on either its family or its cluster, following the procedures for Family Based Prediction and Clustering Based Prediction in Sections 4.2 and 4.3 respectively. For groups (families or clusters) that contain at least as many training instances as the number of features, a dedicated random forest model is trained specifically on this group to predict the most suitable solver.

For a group $G \subseteq I$ with fewer instances than this threshold, we employ a fallback strategy. We considered two alternative strategies: the Single Best Solver (SBS) strategy and the PAR-2 strategy. In the SBS strategy, we always select the solver with the best overall performance on the entire training set, formally defined as:

$$\arg\min_{s \in S} m(s, I)$$

In contrast, the PAR-2 strategy selects the solver with the best performance specifically on the group G:

$$\arg\min_{s \in S} m(s, G)$$

During the execution phase, an instance is first labeled using either the family-based or clustering-based procedure. If the corresponding group was sufficiently large during training, the associated random forest model is used to predict the solver. Otherwise, either the SBS or the PAR-2 strategy is used.

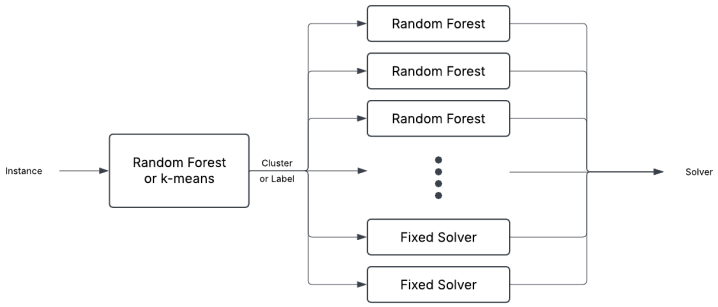


Figure 4.1: Hierarchical Prediction Model

5 Experimental Design

5.1 Dataset and Feature Selection

The dataset for this study consists of 13,742 weighted conjunctive normal form (WCNF) instances sourced from the Global Benchmark Database (GBD) (Iser and Jabs, 2024). We included all instances that appeared in any MaxSAT Evaluation (MSE) from 2019 to 2024. We did not include instances which were used in an MSE but not available in GBD. The remaining instances were randomly selected from GBD.

We split the dataset into three subsets. The *MSE24* dataset contains all 380 instances from the 2024 MaxSAT Evaluations that are available in GBD. The rest of the dataset was processed as follows:

Most instances in GBD are assigned a family label. It describes from which domain the problem originally stems. For instances labeled as 'unknown', we assigned the family based on the filename prefix up until the first occurrence of any delimiter $d \in \{, , \_, -, .\}$. Additionally, we introduced a new family, 'miscellaneous', by relabeling all families occurring five times or fewer in the dataset (excluding MSE24) as 'miscellaneous'. Thus all families have at least size six. We end up with 134 unique families of varying sizes. To create a train-test split, we applied stratified sampling based on family labels, resulting in the dataset *Train Data* with 10,687 instances and the dataset *Test Data* with 2,675 instances.

Feature extraction is not part of the presented work. Feature extraction was performed in advance by Iser and Jabs (2024) using gbds integrated feature extractor for 74 base features on weighted MaxSAT instances. We will henceforth call this set of features *base*.

5.2 K-Fold Cross-Validation

K-fold cross-validation splits the training set into k equally sized folds. One fold is held out as a validation subset, while the remaining $k - 1$ folds are used for training. The process is repeated k times, each time rotating which fold is used for validation. The final performance measure is the average performance across all k runs. (scikit-learn, 2024c)

By using k-fold cross-validation, we minimize test set information leakage while maintaining sufficient data for training and validation without excessive dataset fragmentation.

5.3 Evaluation Metrics

This section provides a detailed overview of the evaluation measures used in this study to assess the performance of the MaxSAT solver portfolios and prediction models.

Notation: Throughout this section we denote by $S = \{s_1, s_2, ...\}$ the chosen solver set and by $I = \{i_1, i_2, ...\}$ a set of instances.

PAR-2 Score The performance metric m for which all algorithm selection models are optimized in this work (cf. Section 3.3) is the penalized average runtime (PAR-2) score, which balances solver speed and robustness. The PAR-2 score penalizes unsolved instances by assigning them twice the timeout value T. In this work, we set the timeout value to $T = 3600$ seconds, following the standard used in the MaxSAT Evaluations (Jeremias Berg et al., 2024).

Formally, let $r : S \times I \to [0, T]$ be some measured solver runtimes. We define the penalized solver runtimes $r_2^T : S \times I \to [0, T) \cup \{2T\}$ as:

$$r_2^T(s, i) := \begin{cases} r(s, i), & r(s, i) < T \\ 2T, & \text{otherwise} \end{cases}$$

The PAR-2 score for a prediction model p is given by:

$$\text{par}_2(p, I) := \frac{1}{|I|} \sum_{i \in I} r_2^T(p(i), i)$$

For a fixed solver $s \in S$ we write $\text{par}_2(s, I)$ as a shorthand for $\text{par}_2(p, I)$ when $p(i) = s, \forall i \in I$. A lower PAR-2 score indicates better performance.

Virtual Best Solver (VBS) is the theoretical prediction model that chooses the solver with the lowest runtime for each instance. Formally, it is given by:

$$\text{VBS}(i) := \arg\min_{s \in S} r_2^T(s, i)$$

Single Best Solver (SBS) is the solver with the lowest PAR-2 score across all instances:

$$\text{SBS}(i) := \arg\min_{s \in S} \sum_{j \in I} r_2^T(s, j)$$

$\text{par}_2(\text{SBS}, I)$ gives an upper bound for the performance of our own approach as this is the current state of the art, whereas $\text{par}_2(\text{VBS}, I)$ gives a lower bound as it is the theoretical best we can hope to achieve.

Family Best Solver (FBS) is the theoretical prediction model which chooses the overall fastest solver on the family the instance is labeled with. Given the family label $l(i)$ of an instance i, we define the FBS as:

$$\text{FBS}(i) := \arg\min_{s \in S} \sum_{j \in I, l(j)=l(i)} r_2^T(s, j)$$

Accuracy of a prediction model p measures how often it could predict the correct label. Let $l(i)$ be the correct and $p(i)$ the predicted label by model p for instance $i \in I$. Accuracy is given by:

$$\text{accuracy}(p, I) := \frac{1}{|I|} \sum_{i \in I} \delta_{l(i)}^{p(i)}$$

where δ is the indicator function.

Matthews Correlation Coefficient (MCC) We also evaluate prediction models with MCC where applicable. As opposed to accuracy, this measure is resistant to imbalances in class labels, i.e. one solver being fastest for most instances. MCC has a range of $[-1, 1]$. The value is 0 or close to 0 if prediction is random, constant or has otherwise poor discriminative ability. A value of 1 indicates perfect prediction while -1 indicates perfect disagreement. Let K be the number of class labels. To define MCC formally, consider a $K \times K$ confusion matrix C and the following intermediate variables:

- $t_k = \sum_i^K C_{ik}$ the number of times class k truly occurred
- $p_k = \sum_i^K C_{ki}$ the number of times class k was predicted
- $c = \sum_k^K C_{kk}$ the total number of samples correctly predicted
- $s = \sum_i^K \sum_j^K C_{ij}$ the total number of samples

Now we can write the MCC as:

$$\text{MCC} := \frac{cs - \sum_k^K p_k t_k}{\sqrt{(s^2 - \sum_k^K p_k^2)(s^2 - \sum_k^K t_k^2)}}$$

(scikit-learn, 2024d)

Gap Closure is the percentage of achieved performance improvement in comparison to the SBS by a prediction model p relative to the theoretically possible performance improvement on a set of instances I, measured by PAR-2. Formally:

$$\frac{\text{par}_2(\text{SBS}, I) - \text{par}_2(p, I)}{\text{par}_2(\text{SBS}, I) - \text{par}_2(\text{VBS}, I)}$$

5.4 Evaluation environment and libraries

Implementation can be found under `https://gitlab.kit.edu/ugqur/ba`.

All experiments and evaluations were performed on a machine with an 11th Gen Intel Core i5-1135G7 @ 2.40 GHz × 8 and 16 GiB RAM. Measuring runtimes of CASHW-MAXSAT_CorePlus, WMaxCDCL, and CGSS2 on all selected instances was done on a machine running Linux 5.14.0 with an Intel Xeon Gold 6138 CPU @ 2.00GHz, 4 sockets, 20 cores, 2 threads and 772638 MB RAM while running at most 10 jobs in parallel.

All experiments and evaluations are written in *python* (3.11.11) using the *notebook* (7.2.2) environment. For many math operations we use *numpy* (2.1.3). For data management we use *gbd-tools* (4.9.7) by Iser and Jabs (2024). For handling data in memory we use *pandas* (2.2.3). For generating plots we use *matplotlib* (3.9.2), *seaborn* (0.13.2) and *graphviz* (0.20.3). For creating, training and evaluating prediction models we use *scikit-learn* (1.6.1) by Pedregosa et al. (2011).

6 Experimental Evaluation

In this chapter, we first choose a good set of solvers. We then evaluate our dataset to draw first conclusions. After giving an overview of the experimental results, we extensively evaluate and compare all of the models as previously introduced in Chapter 4.

6.1 Choosing a set of solvers

This work builds its solver portfolio on submissions from the 2023 MaxSAT Evaluation (MSE23) (Matti Järvisalo et al., 2024). While many solvers competed, it is impractical to include them all for two main reasons. First, evaluating every solver on every training instance would require excessive time. Second, classification models tend to perform better when the number of classes (solvers) is small. Past research confirms that adding more solvers yields limited performance gains once a certain portfolio size is reached (see Bach et al. (2022)).

To identify a set of three complementing solvers that form an effective virtual best solver, we employ the beam-search procedure proposed by Bach et al. (2022) on instances of the 2023 MaxSAT Evaluations which were available on the Global Benchmark Database. We calculated the mean runtime of the VBS for a set size of 1, 2, 3, and 4 to be 915.13, 782.73, 752.27, and 727.55 respectively. We chose to select the set of size three which consists of the following solvers:

- **CASHWMAXSAT_CorePlus (CWM_CP):** a SAT-based MaxSAT solver that employs a core-guided approach (Pan et al., 2023)
- **CGSS2:** a SAT-based MaxSAT solver that employs a core-guided approach (Ihalainen et al., 2021), (Ihalainen, 2022)
- **WMaxCDCL_S6_HS12 (WMaxCDCL):** a branch-and-bound solver enhanced with clause learning (Coll et al., 2023). This is the SBS on all of our datasets.

Figure 6.1 plots the number of solved instances over time for the three selected solvers and for the VBS and FBS. We already see CGSS2 outperforming the other solvers at the beginning, but it is later overtaken by first WMaxCDCL, then CHASWMAXSAT_CorePlus. We can see VBS capitalizing on the different strengths of the solvers, e.g., when first following the shape of CGSS2, then later following a similar gradient as WMaxCDCL.

family	WMaxCDCL	CWM_CP	CGSS2	count	Q
dalculus	14.28	146.73	0.16	15	901.76
auctions	0.69	0.71	480.94	15	693.89
preference	307.01	600.42	9.62	8	62.40
rna-alignment	29.99	21.57	0.66	15	45.78
metro	514.10	34.38	192.08	15	14.96
haplotyping-pedigrees	129.50	10.54	11.27	15	12.29
qcp	2.50	2.31	0.21	9	11.77
mpe	500.92	1190.19	122.95	15	9.68
drmx-cryptogen	550.31	1606.67	169.20	15	9.50
judgment	767.40	4355.30	7200.00	15	9.38
frb	228.65	628.12	1058.33	15	4.63
warehouses	0.50	0.55	1.81	8	3.61
relational-inference	1074.33	2596.75	2964.94	6	2.76
planning	44.90	43.71	17.20	6	2.61
MaxSATQueriesin...	1049.96	1099.07	503.21	15	2.18
causal-discovery	701.55	323.81	346.66	15	2.17
setcover	3656.88	3676.56	7200.00	6	1.97
protein_ins	45.57	44.78	27.90	12	1.63
tcp	1587.41	1010.84	1077.37	8	1.57
timetabling	2086.44	1437.61	2163.36	7	1.50
CSG	10.58	8.65	12.60	10	1.46
switchingactivitymaximization	5767.64	6389.77	7200.00	7	1.25
lisbon-wedding	4438.21	5041.53	5322.51	15	1.20
af	1403.42	1381.27	1358.64	6	1.03
pseudoBoolean	480.50	480.44	480.08	15	1.00

Table 6.1: PAR-2 score on MSE24 instances of our solver set per family. Only displaying families with at least six instances. ‘count’ is the number of instances labeled with that family and ‘Q’ is the number of times the fastest solver is faster than the slowest solver. Sorted descending by ‘Q’.

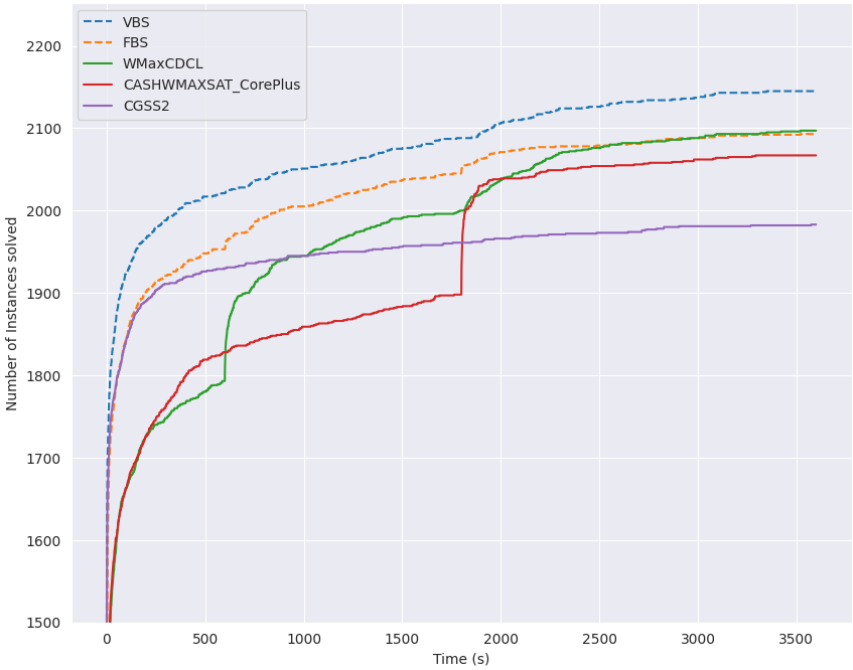


Figure 6.1: VBS, FBS and the chosen solver set on Test Data

6.2 Data Evaluation

Table 6.1 lists all families of size six or bigger of the MSE24 dataset and their mean PAR-2 score of each solver from the solver set. 'count' is the number of instances in the family and 'Q' is the PAR-2 score of the slowest solver divided by the PAR-2 score of the fastest solver which gives the number of times the fastest solver is faster than the slowest. There are some families for which one solver performs much better than the others. For example, looking at family 'judgment' we see WMaxCDCL is a lot faster than the other solvers. However, for the family 'mpe' CGSS2 is the fastest solver by a big margin. This means always selecting the fastest solver based on the family label could lead to a significant performance improvement over the SBS. To get a grasp of how much of an improvement we can expect, we have a look at the FBS in Figure 6.1. Here we see the number of solved instances over time for each solver from our solver set and for the resulting FBS and VBS on Test Data. WMaxCDCL is the SBS. Although the FBS does perform better than the SBS overall with a PAR-2 score of 1653.54 compared to 1718.57, it is not close to the VBS with a PAR-2 score of 1516.46 and even solves fewer instances than SBS within the timeout period. This already lowers the expectations for Family Based Prediction.

Figure 6.2 is a heatmap of the appearance of family labels in MaxSAT Evaluations (MSE) for the years 2019-2024. Only families with a total of at least six instances across the years are displayed. A value of one means the family was the most common in that year, a value of zero means the family did not appear in that year. As we can see, family appearance

changes a lot from year to year. This means if we get a new set of instances (e.g. an upcoming MSE 2025), it is likely to contain families for which we only have sparse training data. To simulate this data sparsity and investigate the stability of Family Based Prediction, we also compare all our models on the MSE24 dataset.

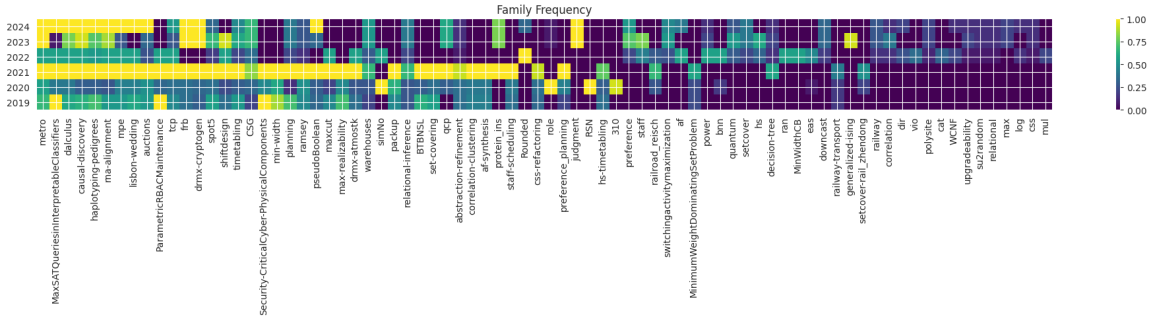


Figure 6.2: Families and their frequency of appearance in MSE by year relative to the most common family of that year. Only displaying families with a total of at least six instances across the years.

6.3 Overview

This section will provide an overview of the experimental results, which we will then detail in the following sections.

We evaluated all models on the training data using 10-fold cross-validation. This results in ten values of each metric, one for each of the folds. Figure 6.3, 6.4, and 6.5 visualize these values for accuracy, MCC, and PAR-2 Score respectively as box plots for all models and for VBS, SBS and FBS. Mean accuracy ranges from 0.2973 to 1.0, mean MCC ranges from -0.0514 to 1.0 where VBS achieves 1.0 in both metrics as expected. CHP, FHP and DSP perform well on accuracy and MCC, while FBS, CP and FP perform about the same as SBS, which we are trying to beat. PAR-2 Score wise, VBS naturally performs best, after which follows SBS, FHP and CHP. SBS achieved the lowest PAR-2 score with CP and FP being slightly better.

Figure 6.6 shows the achieved gap closure for all models on Test Data and MSE24. Direct Solver Predictor and CHP perform best. Compared to CHP, Direct Solver Predictor is 0.0426 worse on Test Data and 0.1048 better on MSE24. FHP also performs similarly to CHP and Direct Solver Predictor on Test Data. None of FP, CP, or FHP showed any performance improvements on MSE24.

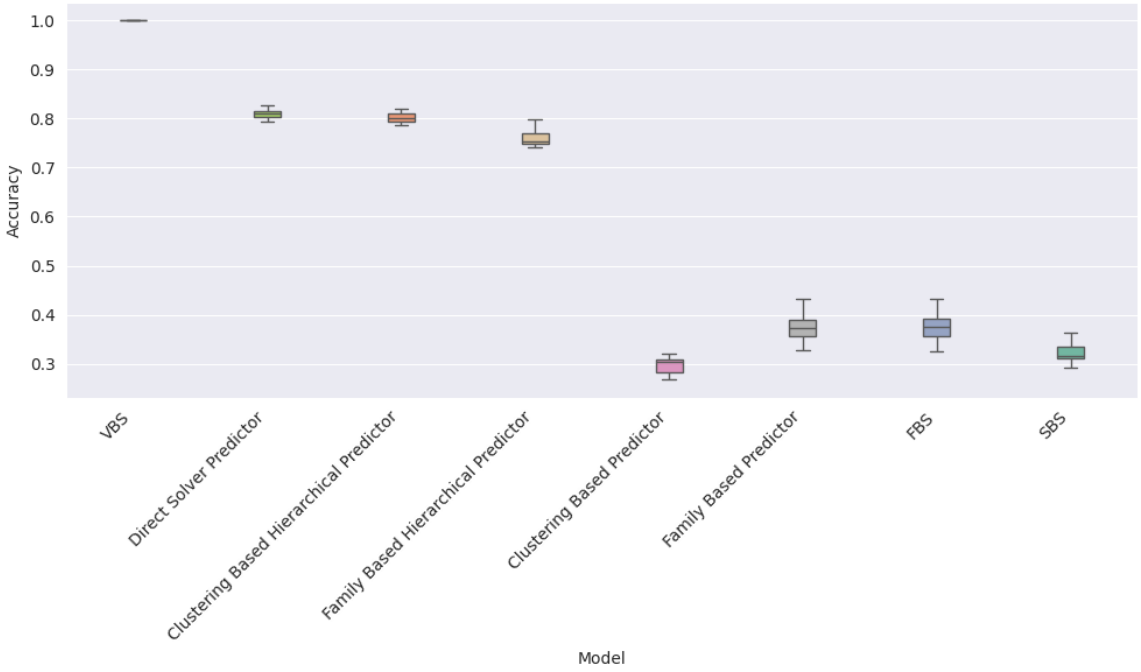


Figure 6.3: Accuracy of all models on Train Data with 10-fold cross-validation

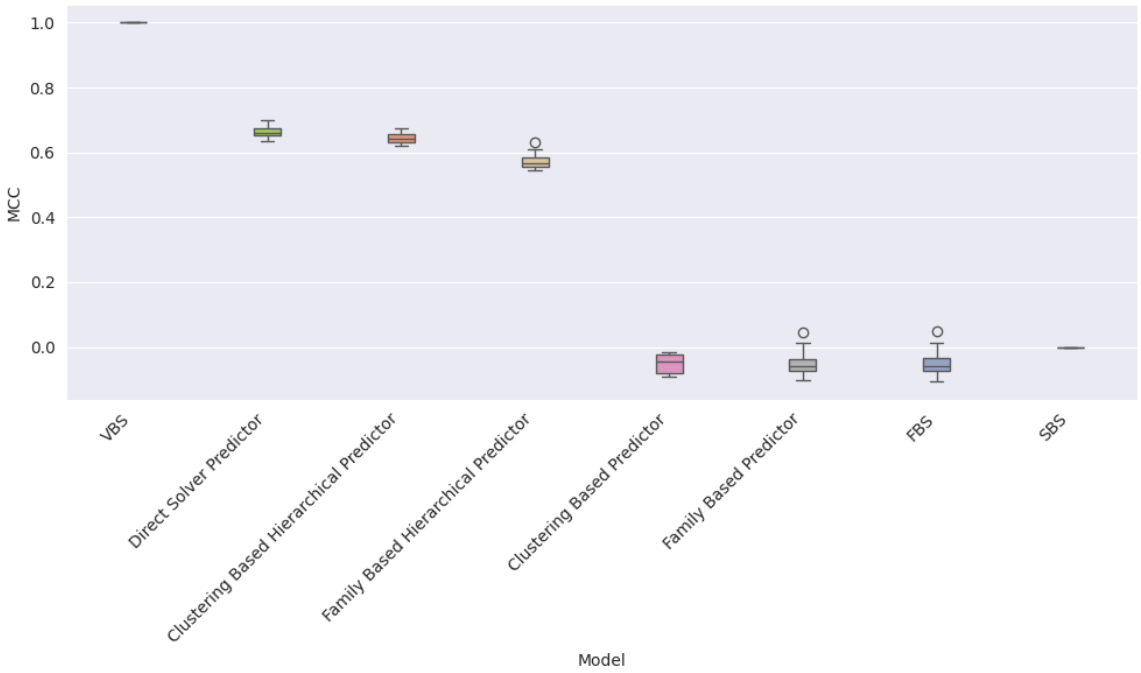


Figure 6.4: MCC of all models on Train Data with 10-fold cross-validation

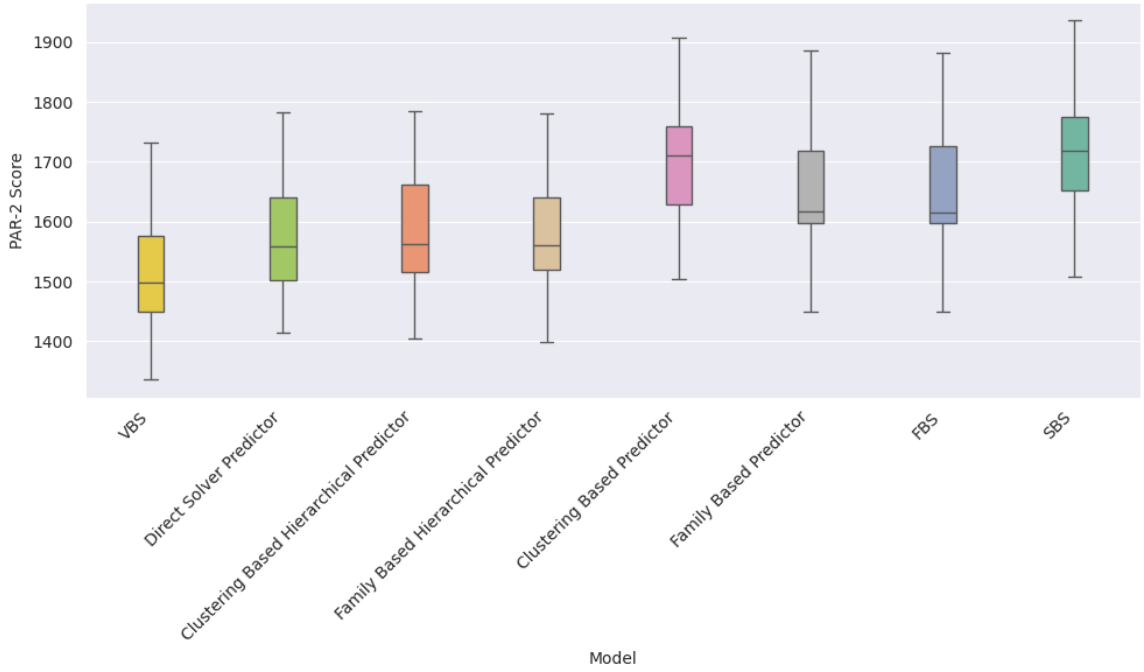


Figure 6.5: PAR-2 Score of all models on Train Data with 10-fold cross-validation

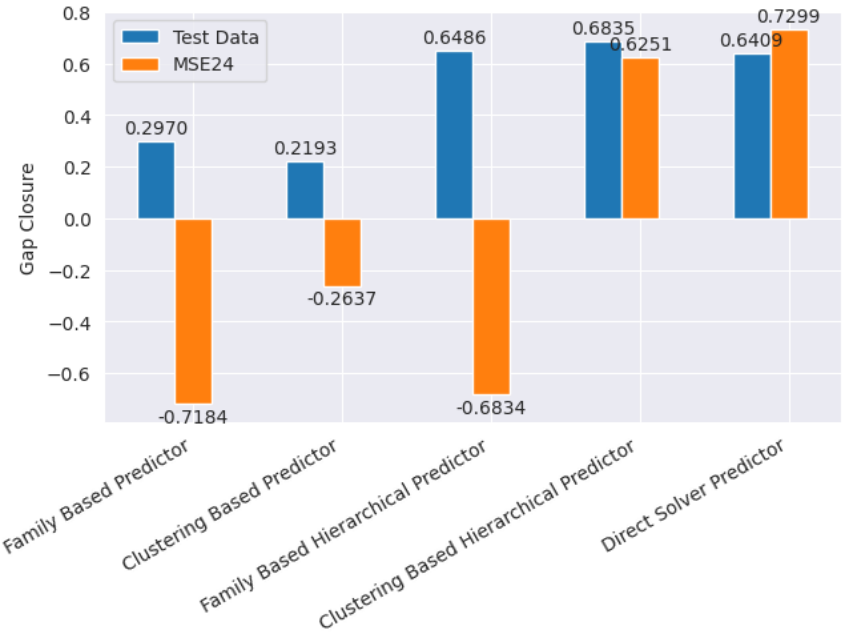


Figure 6.6: Gap Closure of all models on Test Data and MSE24

6.4 Direct Solver Predictor

In this section, we evaluate the Direct Solver Predictor (DSP) as introduced in 4.1.

The DSP achieves an average accuracy of 81.10% and average MCC of 66.24% on Train Data with 10-fold cross-validation as visualized in Figure 6.3 and Figure 6.4. The PAR-2 score across 10 folds depicted in Figure 6.5 is 1568.39 on average.

When trained on the whole of Train Data, DSP achieves a gap closure of 64.09% on Test Data and 72.99% on MSE24.

DSP outperforms all other models on every metric except gap closure on Test Data where FHP and CHP do marginally better.

6.5 Family Based Preciction

In this Section, we evaluate Family Based Prediction (FP) as introduced in Section 4.2.

We first analyze the predictive qualities for families (before applying the mapping fts). Figure 6.7 shows a boxplot of the accuracy and the MCC of FP and DSP with 10-fold cross-validation on Train Data. On average across the ten folds, the random forest correctly predicted the family with an accuracy of 94.80%. The mean MCC across the ten folds is 93.68%. The random forest of DSP directly predicting one of the three solvers only achieves an average accuracy of 81.10% and an average MCC of 66.24%. This means the prediction of families can be done much more precisely, even for the large number of 134 families. The goal is to translate this accuracy to solver prediction. Applying the mapping fts we get the accuracy depicted in the previous Figure 6.3 of 37.47%. This is notably worse, which means we are not able to translate family accuracy to solver accuracy with a simple mapping. Notably, it does beat the FBS which has an accuracy of 37.38%. This can happen if the random forest does not predict the correct family F_{true} but instead predicts a wrong family F_{pred}, for which however the solver $\mathrm{fts}(F_{pred})$ is the fastest but $\mathrm{fts}(F_{true})$ is not, for this specific instance. Having a look at the PAR-2 score in Figure 6.5 again, FP, with a score of 1644.74, also does better than FBS by 0.75. It also could have been the case that accuracy is higher for FP but PAR-2 score is not, if the solvers predicted incorrectly by FP were slower than the fastest solver by too much.

Now the Family Based Predictor is trained on the whole of Train Data. In Figure 6.8 we see the number of solved instances over time by VBS, FBS and FP on Test Data. The Family Based Predictor performs virtually the same as the FBS, meaning we gained all possible performance improvement from this method. As seen in Figure 6.6 FP did improve over SBS with a gap closure of 29.70%.

Evaluating on the MSE24 dataset, the accuracy of predicting the correct family drops to 66.05%. This is expected as there appear families for which we have few or even no training samples (cf. Section 6.2). On MSE24 the Family Based Predictor achieves a PAR-2 score

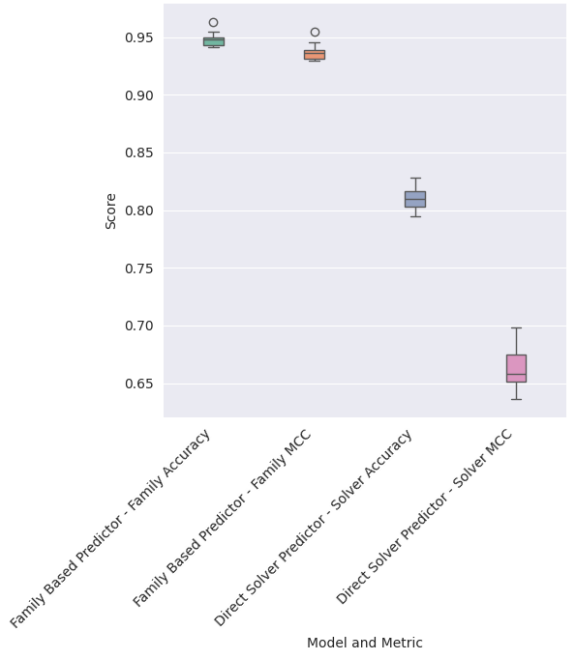


Figure 6.7: Comparison of accuracy and MCC between FP and DSP

of 1340.91 which is 8.69% worse than the SBS with a PAR-2 score of 1282.83. In Figure 6.9 we see the comparison on the MSE24 data where the number of instances solved over time by SBS, Family Prediction and Clustering Prediction is shown. We see Family Based Prediction performing best for the first 500 seconds. Then SBS takes the lead, solving the most instances and achieving the best PAR-2 score.

6.6 Clustering Based Prediction

In this section we evaluate Clustering Based Prediction as introduced in Section 4.3.

We first need to choose the number of clusters k. In Figure 6.10a we see the average PAR2 score over ten folds on Train Data for $k \in [15, 44]$. We find $k = 34$ to achieve the lowest PAR2 score which is the same value Heinen et al. (2022) found to be the best cluster size for SAT instances. We set the number of clusters $k = 34$ from now on.

The resulting clusters of the Clustering Based Predictor, trained on Train Data, and their size can be seen in Figure 6.11. The bars are labeled with the size relative to the total number of instances in Train Data, the largest family the cluster consists of, and how much of the cluster that family makes up. We observe a highly variable cluster size. There are a few large clusters, with sizes exceeding 600, and many small clusters with the minimum size. There are many clusters with very dominant Families. 14 out of the 34 clusters have a family making up more than 80% of the cluster. Figure 6.12 presents a heatmap

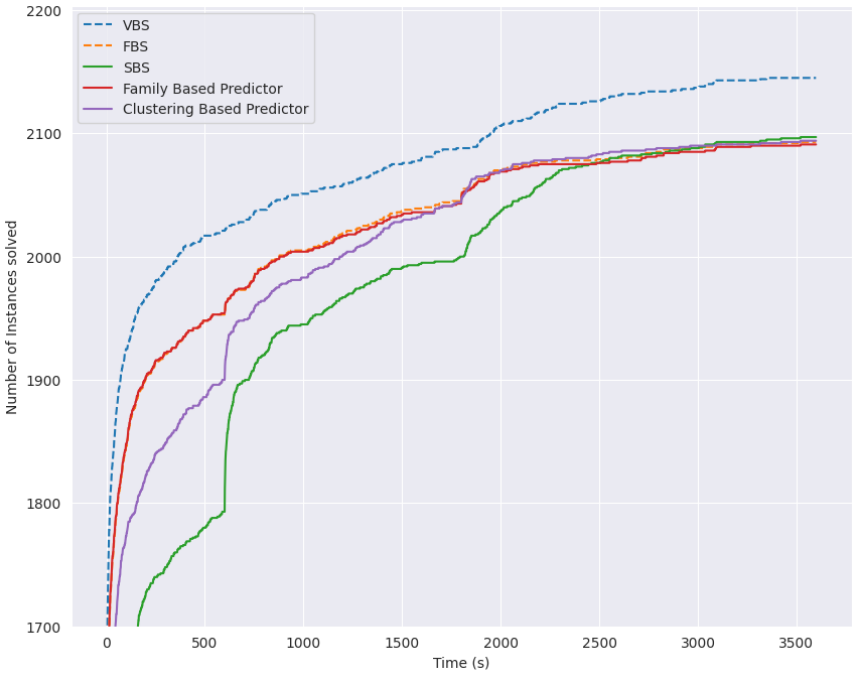


Figure 6.8: Number of instances solved over time by FBS, Family Based Prediction, Clustering Based Prediction and SBS on Test Data

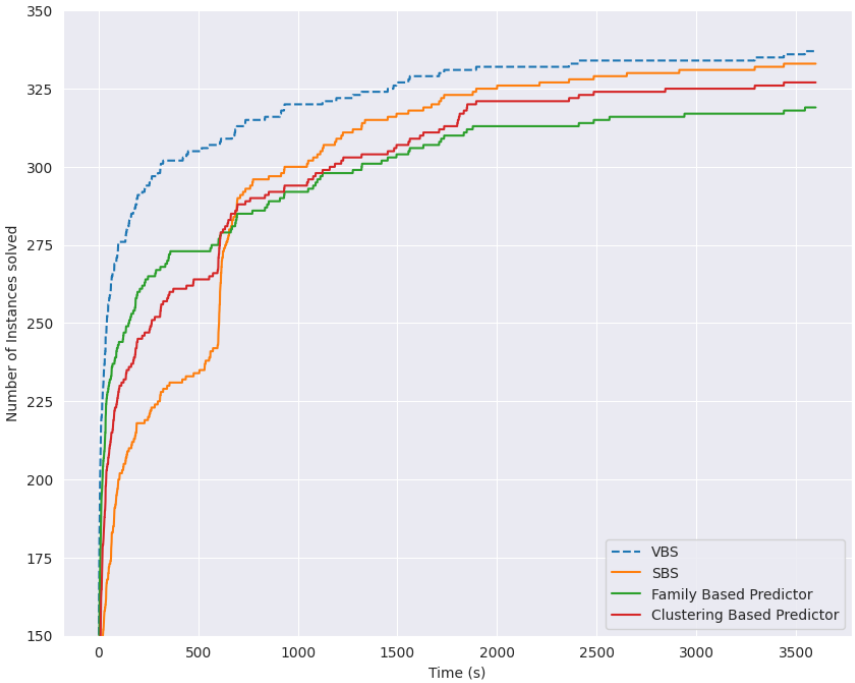
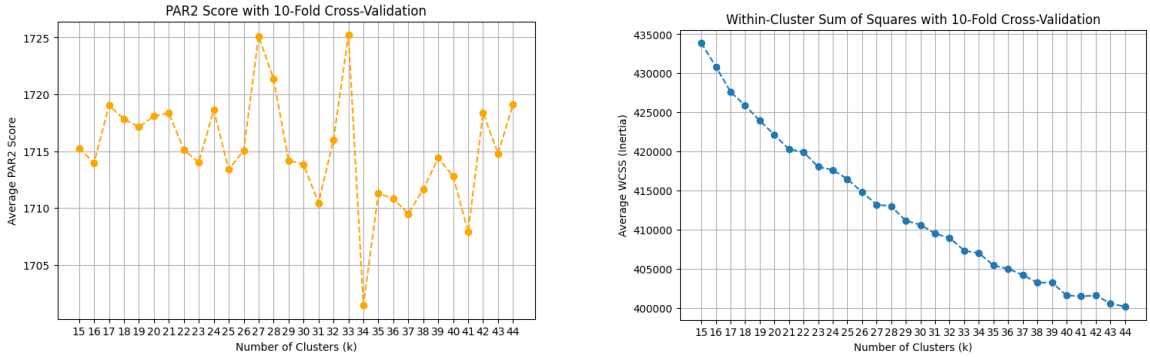


Figure 6.9: Number of instances solved over time by Family Based Prediction, Clustering Based Prediction and SBS on MSE24 data

illustrating how families are distributed over the clusters, where each row sums up to 1. The sparsity of the heatmap indicates that most family-cluster combinations have minimal or no representation. This suggests that each cluster is predominantly composed of a small number of families, while many families appear in only a few clusters or even a single one. This strong association between clusters and families aligns with the findings of Heinen et al. (2022), confirming that similar clustering behavior extends to MaxSAT instances.



(a) Average PAR2 Score with 10-Fold Cross-Validation on Train Data

(b) Average WCSS with 10-Fold Cross-Validation on Train Data

Figure 6.10

6.7 Comparing Families and Clusters

In this section, we compare Family Based Prediction and Clustering Based Prediction.

Looking at Figure 6.8 again, we find the Clustering Based Predictor performing almost consistently worse than the Family Based Predictor but better than SBS, only achieving a PAR-2 score of 1674.24 which is 15.69 worse than Family Based Prediction but still beating SBS by 44.33.

However, on the MSE24 data in Figure 6.9 we see Clustering Based Predictor to be performing better than the Family Based Predictor, achieving a PAR-2 score of 1239.39, which is 101.84 less than the Family Based Predictor. Interestingly, this is still worse than the SBS even though it was better than SBS on Test Data. As the only intended difference between Test Data and MSE24 is the kind of families appearing, and clustering is not dependent on the family labels, this further indicates association between clusters and families. If clusters were completely independent of families, we would expect behavior similar to DSP, where we could even see gap closure change in the opposite direction (cf. Figure 6.6.

Let us examine Figure 6.4 again. FBS has a MCC of -0.05 meaning it has no predictive power. Family Based Prediction is expected to perform about as well as FBS, which holds here with a MCC of -0.045. The Clustering Based Predictor also has a MCC close to

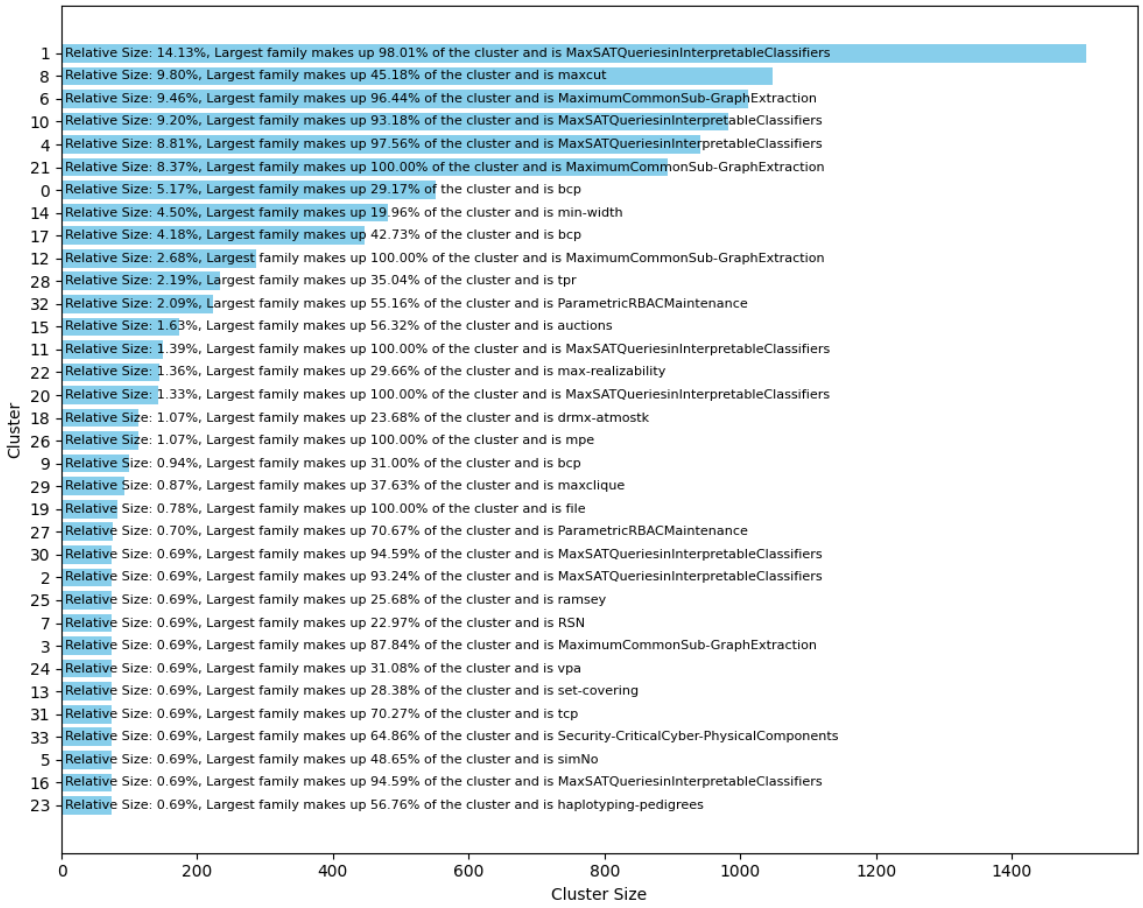


Figure 6.11: Size of clusters and the largest family they consist of on Train Data

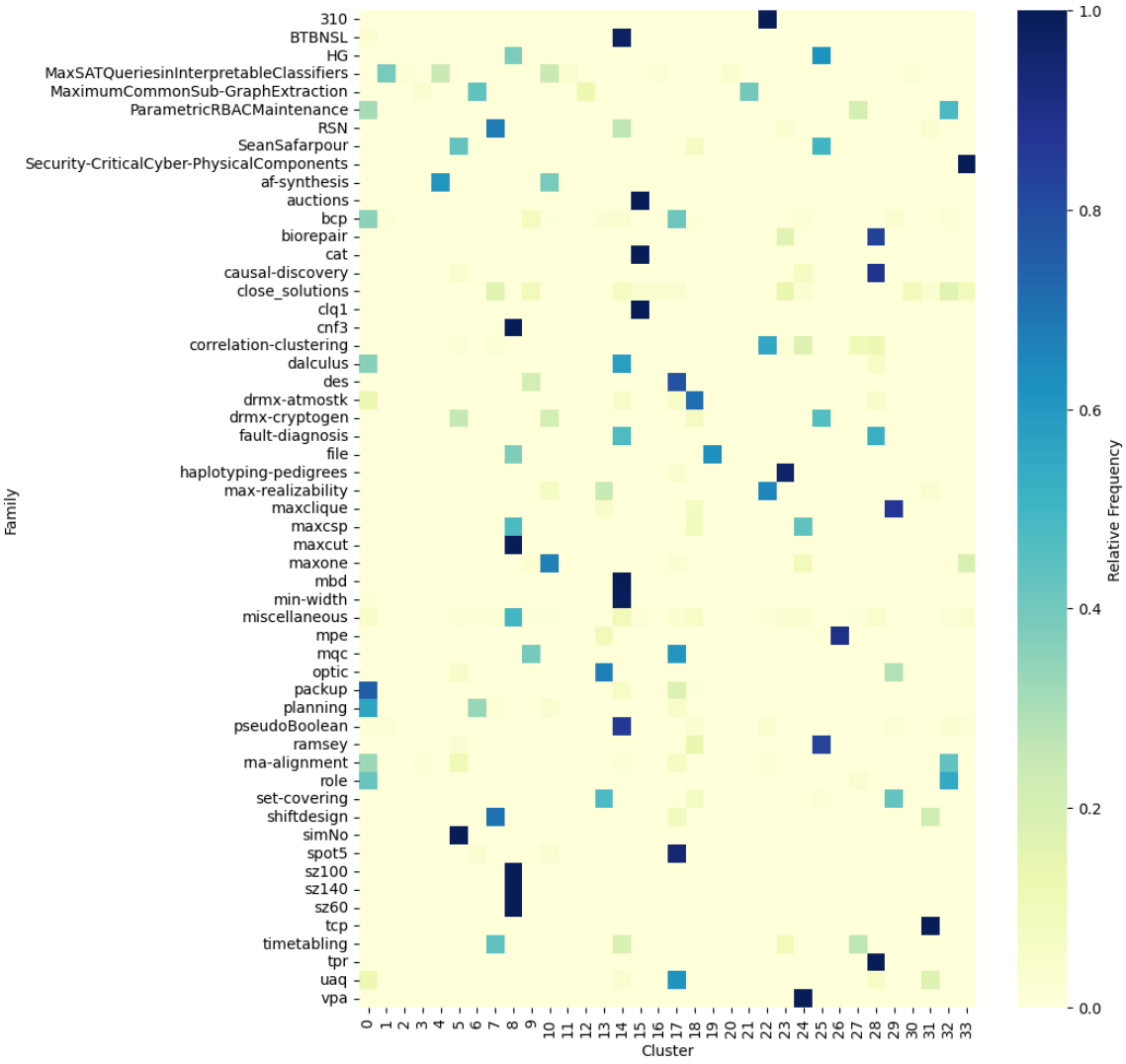


Figure 6.12: Heatmap of the relative frequency of a family in a cluster on Train Data. Omitting families of size 20 or less.

zero with -0.054. We conclude that families and clusters do not have an impact on the discriminative ability of the models to predict the fastest solver.

6.8 Hierarchical Prediction

In this section, we evaluate and compare Family Based Hierarchical Prediction (FHP) and Clustering Based Hierarchical Prediction (CHP) as introduced in Section 4.4.

In Table 6.2 we first compare the PAR-2 strategy with the SBS strategy for both the Clustering and Family Based Hierarchical Predictor. The strategies perform almost identically, with PAR-2 being marginally better in all three scores for both the clustering and family approach. Thus, henceforth Clustering and Family Based Hierarchical Predictor will refer to the variant using the PAR-2 strategy.

Predictor	Accuracy	MCC	PAR-2
Clustering Based Hierarchical Predictor PAR-2	80.37%	0.6494	1569.90
Clustering Based Hierarchical Predictor SBS	80.34%	0.6489	1569.21
Family Based Hierarchical Predictor PAR-2	75.95%	0.5739	1569.25
Family Based Hierarchical Predictor SBS	71.77%	0.5170	1593.85

Table 6.2: Mean accuracy, MCC and PAR-2 score of Clustering and Family Based Hierarchical Predictor with PAR-2 and SBS strategy of 10-fold cross-validation on Train Data

Now that we have decided on the PAR-2 strategy, let us look at the resulting structures of the Hierarchical Predictors. The full structure can be found in the appendix in Table A.1 for CHP and Table A.2 for FHP. CHP trains a random forest for all clusters, as we added the constraint of a minimum cluster size. As there are a lot more families than clusters, many families are small and do not have as many instances as there are features, thus not being able to train a random forest, which is a drawback to FHP. FHP only trains a random forest on 2104 instances out of the 2675 available in Train Data.

Figure 6.13 shows the number of solved instances over time on Test Data for VBS, SBS, CP, CHP, FHP and DSP. We see CHP outperforming FHP until about the 1800s mark. Then FHP takes the lead achieving a PAR-2 score on the dataset Test Data of 1587.47 while CHP achieves 1580.42. FHP solves 2123 instances within the timeout of 3600s, which is 2 more than CHP and 6 more than DSP. Both Hierarchical Predictors perform better than Family Based Prediction and Clustering Based Prediction. This means that introducing the group-specific random forests improved performance. The gap closure in Figure 6.6 shows FHP and CHP are only marginally better than the much simpler DSP. Most notably though is the difference in performance on MSE24 where FHP achieves $> 60\%$ gap closure on Test Data but -60% on MSE24 thus not only being worse than SBS but even being worse than CP. Figure 6.14 shows the number of solved instances over time on MSE24 for VBS, SBS, CHP, FHP and DSP. FHP initially performs a bit better than SBS, which is due to

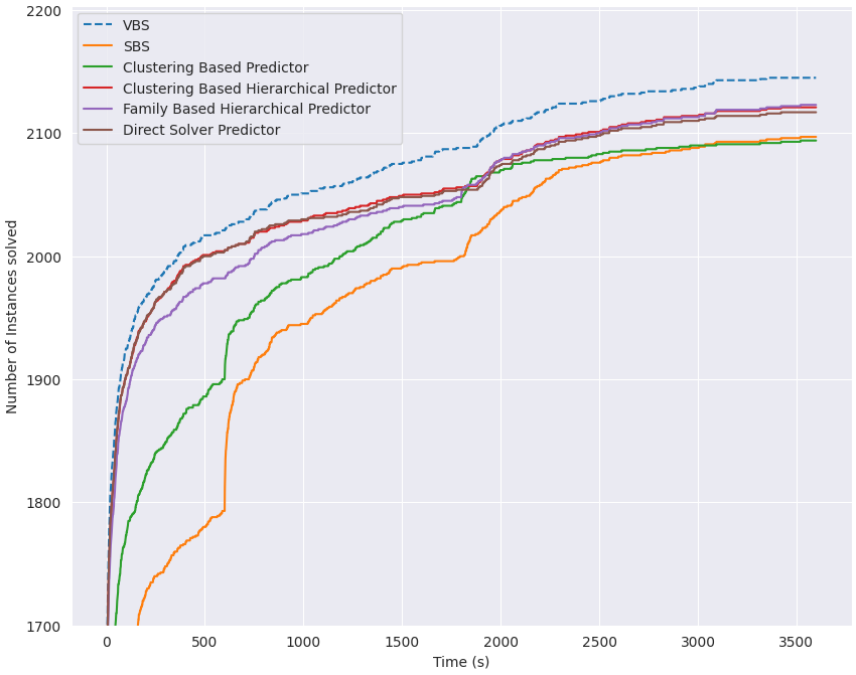


Figure 6.13: Number of instances solved over time by FHP, CHP and DSP on Test Data with Family Based Predictor, VBS, and FBS for comparison

one of the other solvers performing better than SBS at the start (cf. Section 6.1). CHP, however, performs almost as well as the DSP, achieving a similar gap closure on Test Data and MSE24. This means we are able to create a robust solver selector by replacing families with clusters, though it only performs about as well as the much simpler DSP.

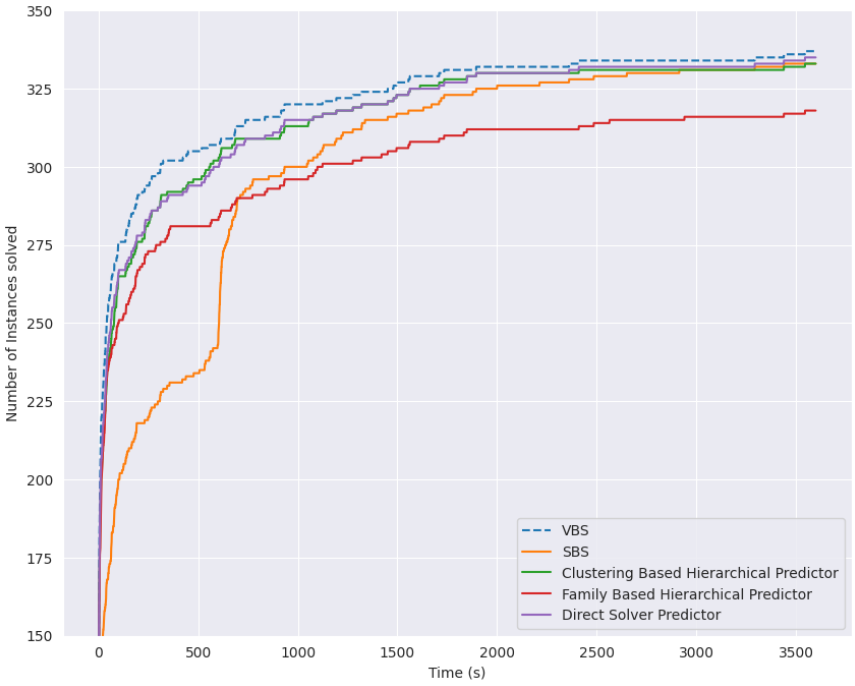


Figure 6.14: Number of instances solved over time by the Hierarchical Predictors and the Solver Predictor on MSE24

7 Conclusion and Future Work

This work investigated multiple algorithm selection models for MaxSAT, aiming to bridge the performance gap between the Single Best Solver (SBS) and the theoretical Virtual Best Solver (VBS). We introduced a suite of models—including Direct Solver Prediction (DSP), Family Based Prediction (FP), Clustering Based Prediction (CP), and their hierarchical counterparts—to assess whether grouping strategies based on family labels or unsupervised clusters could offer explainable and effective solver selection.

Our experiments on a large weighted MaxSAT dataset revealed several key findings. First, if families are well represented in the training data, Family Based Hierarchical Prediction (FHP) achieves the best score by a very small margin over Clustering Based Hierarchical Prediction (CHP) and Direct Solver Prediction (DSP). DSP demonstrated robust performance that consistently closed the performance gap by up to 73%, matching or slightly exceeding the more complex hierarchical approaches especially on a dataset with previously unseen families. Clustering Based Prediction (CP) and Family Based Prediction (FP) were outperformed by FHP, CHP and DSP. Although FP achieves high accuracy in identifying instance families, its mapping to solver selection suffers especially when facing novel or underrepresented families. While clustering approaches provided a more robust alternative to families, their overall performance was comparable to that of the direct approach.

We also confirmed observations from prior work on the association between families and clusters for SAT instances to hold for MaxSAT as well.

Future Work While this study has demonstrated the effectiveness of solver selection models for MaxSAT, multiple directions for future research remain. First, exploring more advanced feature selection may further improve model accuracy and generalization to unseen problem instances. Additionally, deep learning-based approaches, such as graph neural networks, could be investigated to capture structural properties of MaxSAT instances more effectively. Further study of MaxSAT families is also interesting, for example, by comparing them to other groupings to examine any unique predictive power. Instead of using family labels from filename prefixes as done in this study, a more thorough labeling by a domain expert could further improve accuracy and stability for family-based models. Such findings could inspire advancements in solver configuration as they can lead to insights into specific strengths and weaknesses of individual solvers. Finally, applying similar algorithm selection techniques to other combinatorial optimization problems, such as Constraint Satisfaction Problems or Mixed Integer Programming, could provide insights into the generalizability of the proposed methods.

A Structure of Hierarchical Predictors

Cluster	Train Size	Step 2	Test Size	Accuracy	Gap Closure
1	1510	Random Forest	369	80.49%	98.40%
8	1047	Random Forest	266	83.46%	31.64%
6	1011	Random Forest	260	91.54%	93.01%
10	983	Random Forest	242	36.78%	-135.93%
4	942	Random Forest	234	79.49%	70.72%
21	894	Random Forest	218	92.66%	-124.11%
0	552	Random Forest	142	80.28%	71.28%
14	481	Random Forest	133	84.21%	97.78%
17	447	Random Forest	103	81.55%	14.46%
12	286	Random Forest	81	100.00%	100.00%
28	234	Random Forest	54	85.19%	86.79%
32	223	Random Forest	48	93.75%	83.48%
15	174	Random Forest	45	88.89%	21.09%
11	149	Random Forest	40	100.00%	-
22	145	Random Forest	48	72.92%	67.18%
20	142	Random Forest	39	94.87%	77.88%
18	114	Random Forest	49	63.27%	-84.51%
26	114	Random Forest	29	100.00%	100.00%
9	100	Random Forest	31	74.19%	99.68%
29	93	Random Forest	30	70.00%	82.48%
19	83	Random Forest	21	47.62%	7.32%
27	75	Random Forest	21	95.24%	-602.45%
2	74	Random Forest	25	96.00%	0.00%
3	74	Random Forest	12	91.67%	77.66%
5	74	Random Forest	9	77.78%	87.47%
7	74	Random Forest	8	75.00%	95.92%
13	74	Random Forest	8	75.00%	-126.31%
16	74	Random Forest	14	100.00%	100.00%
23	74	Random Forest	22	68.18%	21.45%
24	74	Random Forest	3	0.00%	-44284.96%
25	74	Random Forest	6	100.00%	100.00%
30	74	Random Forest	26	100.00%	-
31	74	Random Forest	16	62.50%	99.37%
33	74	Random Forest	23	95.65%	97.27%

Table A.1: Structure of the Clustering Based Hierarchical Predictor. With cluster id, size of the cluster on Train Data and on Test Data, whether a random forest was used for step two, the accuracy of the random forest in step two on Test Data, and the achieved gap closure on Test Data. Sorted by the size of the cluster on Train Data.

Table A.2: Structure of the Family Based Hierarchical Predictor. With the accuracy of random forest on Test Data in step two where applicable. Sorted by the size of the family on Train Data.

Family	Train Size	Step 2	Test Size	Accuracy	Gap Closure
MaxSATQueriesinInterpretableCl...	3815	Random Forest	954	73.06%	86.03%
MaximumCommonSub-GraphExtraction	2226	Random Forest	557	93.18%	46.99%
maxcut	473	Random Forest	128	89.06%	3.73%
bcp	454	Random Forest	116	81.90%	96.83%
miscellaneous	371	Random Forest	81	81.48%	56.03%
ParametricRBACMaintenance	255	Random Forest	65	90.77%	54.48%
packup	146	Random Forest	39	92.31%	94.56%
file	133	Random Forest	35	51.43%	7.25%
mpe	114	Random Forest	29	100.00%	100.00%
auctions	98	Random Forest	30	90.00%	86.44%
min-width	97	Random Forest	27	85.19%	84.22%
mbd	86	Random Forest	22	100.00%	100.00%
tpr	82	Random Forest	21	90.48%	99.77%
pseudoBoolean	73	WMaxCDCL	17	-	0.00%
cnf3	70	WMaxCDCL	17	-	0.00%
causal-discovery	69	CASHWMAXSAT_CorePlus	18	-	-24.17%
correlation-clustering	67	WMaxCDCL	19	-	0.00%
mqc	66	CGSS2	17	-	100.00%
max-realizability	57	CGSS2	12	-	-97.43%
planning	57	CGSS2	14	-	79.67%
rna-alignment	55	CGSS2	14	-	100.00%
haplotyping-pedigrees	53	CGSS2	13	-	99.18%
tcp	52	CGSS2	13	-	77.33%
dalculus	50	CGSS2	12	-	99.95%
Security-CriticalCyber-Physica...	48	WMaxCDCL	14	-	0.00%

Continued on next page

Table A.2 – Continued

Family	Train Size	Step 2	Test Size	Accuracy	Gap Closure
set-covering	44	WMaxCDCL	13	-	0.00%
BTBNSL	42	WMaxCDCL	12	-	0.00%
maxone	42	CGSS2	11	-	99.95%
timetabling	41	CASHWMAXSAT_CorePlus	9	-	62.31%
des	38	CASHWMAXSAT_CorePlus	9	-	57.15%
drmx-atmostk	38	CGSS2	9	-	100.00%
maxclique	38	CASHWMAXSAT_CorePlus	9	-	47.79%
simNo	36	WMaxCDCL	9	-	0.00%
spot5	36	CASHWMAXSAT_CorePlus	9	-	-150.35%
HG	34	CASHWMAXSAT_CorePlus	8	-	-7336.51%
RSN	34	WMaxCDCL	10	-	0.00%
sz100	34	WMaxCDCL	6	-	0.00%
uaq	34	CASHWMAXSAT_CorePlus	8	-	24.59%
role	33	CGSS2	9	-	-0.05%
clq1	32	CASHWMAXSAT_CorePlus	8	-	99.44%
af-synthesis	31	WMaxCDCL	10	-	0.00%
sz140	31	WMaxCDCL	7	-	0.00%
close_solutions	30	WMaxCDCL	6	-	0.00%
cat	29	CASHWMAXSAT_CorePlus	2	-	-1.71%
ramsey	29	CGSS2	7	-	100.00%
SeanSafarpour	28	WMaxCDCL	8	-	0.00%
drmx-cryptogen	28	CGSS2	7	-	100.00%
biorepair	24	CASHWMAXSAT_CorePlus	7	-	70.94%
maxcsp	23	CGSS2	6	-	100.00%
shiftdesign	23	WMaxCDCL	6	-	0.00%
sz60	23	CASHWMAXSAT_CorePlus	5	-	66.72%
vpa	23	CASHWMAXSAT_CorePlus	6	-	34.07%

Continued on next page

Table A.2 – Continued

Family	Train Size	Step 2	Test Size	Accuracy	Gap Closure
310	22	CGSS2	7	-	99.89%
fault-diagnosis	21	CASHWMAXSAT_CorePlus	8	-	10.88%
optic	21	CASHWMAXSAT_CorePlus	5	-	11.13%
frb	20	WMaxCDCL	6	-	0.00%
preference_planning	20	CGSS2	5	-	94.67%
qcp	19	CGSS2	5	-	99.79%
Rounded	18	WMaxCDCL	2	-	-
implicit_hitting_set_pathological	18	WMaxCDCL	5	-	0.00%
aes-key-recovery	16	WMaxCDCL	4	-	0.00%
generalized-ising	16	WMaxCDCL	4	-	-
gen-hyper-tw	15	WMaxCDCL	5	-	-
kbtree	15	WMaxCDCL	4	-	0.00%
reversi	15	CGSS2	4	-	71.39%
vcsp21	14	CGSS2	4	-	100.00%
MinWidthCB	13	WMaxCDCL	0	-	-
ran	13	CASHWMAXSAT_CorePlus	1	-	100.00%
atcoss	12	WMaxCDCL	3	-	0.00%
hs-timetabling	12	WMaxCDCL	1	-	-
lisbon-wedding	12	WMaxCDCL	3	-	0.00%
metro	12	CGSS2	3	-	93.26%
extension-enforcement	11	CASHWMAXSAT_CorePlus	3	-	-
power	11	CGSS2	5	-	80.43%
eas	10	CGSS2	2	-	100.00%
routing	10	CGSS2	2	-	100.00%
staff	10	CGSS2	2	-	100.00%
staff-scheduling	10	CGSS2	2	-	-
xai-mindset2	10	WMaxCDCL	3	-	0.00%

Continued on next page

Table A.2 – Continued

Family	Train Size	Step 2	Test Size	Accuracy	Gap Closure
20	9	WMaxCDCL	2	-	0.00%
ConsistentQueryAnswering	9	WMaxCDCL	1	-	0.00%
css-refactoring	9	CGSS2	2	-	84.04%
mul	9	CASHWMAXSAT_CorePlus	2	-	-19.32%
railway-transport	9	CGSS2	2	-	100.00%
s2v140c1300	9	CASHWMAXSAT_CorePlus	2	-	-
af	8	CASHWMAXSAT_CorePlus	0	-	-
vcsp37	8	CGSS2	2	-	100.00%
vcsp42	8	CGSS2	2	-	100.00%
vio	8	WMaxCDCL	1	-	-
abstraction-refinement	7	CASHWMAXSAT_CorePlus	2	-	-
hs	7	CGSS2	3	-	100.00%
min-fill	7	WMaxCDCL	2	-	0.00%
optimizing	7	WMaxCDCL	2	-	-
railroad_reisch	7	WMaxCDCL	2	-	-
s2v100c1400	7	CASHWMAXSAT_CorePlus	0	-	-
vcsp50	7	CGSS2	2	-	100.00%
close	6	CGSS2	0	-	-
decision-tree	6	WMaxCDCL	1	-	-
logic-synthesis	6	CASHWMAXSAT_CorePlus	1	-	-
min1s2v160c800	6	CASHWMAXSAT_CorePlus	2	-	-121.30%
min1s2v160c960	6	WMaxCDCL	2	-	0.00%
ram	6	CGSS2	1	-	100.00%
s2v100c1600	6	CASHWMAXSAT_CorePlus	2	-	93.52%
s2v120c1400	6	CASHWMAXSAT_CorePlus	1	-	100.00%
s2v120c1600	6	CASHWMAXSAT_CorePlus	2	-	100.00%
s2v140c1400	6	CASHWMAXSAT_CorePlus	1	-	100.00%

Continued on next page

Table A.2 – Continued

Family	Train Size	Step 2	Test Size	Accuracy	Gap Closure
s3v110c1000	6	WMaxCDCL	1	-	-
s3v70c1100	6	WMaxCDCL	1	-	-
s3v70c1200	6	WMaxCDCL	1	-	-
s3v70c800	6	WMaxCDCL	1	-	-
setcover-rail_zhendong	6	WMaxCDCL	3	-	-
treewidth	6	WMaxCDCL	1	-	0.00%
treewidth-computation	6	WMaxCDCL	1	-	0.00%
unw	6	WMaxCDCL	1	-	-
vcsp24	6	CGSS2	2	-	100.00%
vcsp34	6	CGSS2	2	-	100.00%
vcsp36	6	CGSS2	1	-	100.00%
warehouses	6	WMaxCDCL	2	-	-
extension	5	CASHWMAXSAT_CorePlus	1	-	100.00%
judgment	5	CGSS2	1	-	-
kbtree9	5	CASHWMAXSAT_CorePlus	1	-	-78.25%
ms	5	CASHWMAXSAT_CorePlus	1	-	-336.48%
normalized	5	CGSS2	2	-	100.00%
p	5	CGSS2	2	-	100.00%
protein_ins	5	CGSS2	2	-	62.86%
s2v100c1500	5	CASHWMAXSAT_CorePlus	1	-	-
s3v110c800	5	WMaxCDCL	1	-	-
s3v110c900	5	WMaxCDCL	0	-	-
s3v70c1000	5	WMaxCDCL	1	-	-
s3v70c1300	5	WMaxCDCL	1	-	-
s3v90c1000	5	WMaxCDCL	1	-	-
s3v90c1100	5	WMaxCDCL	1	-	-
vcsp33	5	CGSS2	1	-	100.00%

Continued on next page

Table A.2 – Continued

Family	Train Size	Step 2	Test Size	Accuracy	Gap Closure
MinimumWeightDominatingSetProblem	4	WMaxCDCL	1	-	-

Bibliography

Ansótegui, C., Malitsky, Y., and Sellmann, M. (2014). MaxSAT by Improved Instance-Specific Algorithm Configuration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28.

Ansótegui, C., Pon Farreny, J., Sellmann, M., and Tierney, K. (2017). Reactive Dialectic Search Portfolios for MaxSAT. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31.

Bach, J., Iser, M., and Böhm, K. (2022). A Comprehensive Study of k-Portfolios of Recent SAT Solvers. In *LIPIcs, Volume 236, SAT 2022*, volume 236, pages 2:1–2:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Artwork Size: 18 pages, 894689 bytes ISSN: 1868-8969 Medium: application/pdf.

Bennett, K., Bradley, P., and Demiriz, A. (2000). Constrained K-Means Clustering. Technical Report MSR-TR-2000-65.

Coll, J., Li, S., Li, C.-M., Manyà, F., Habet, D., Cherif, M. S., and He, K. (2023). WMax-CDCL in MaxSAT Evaluation 2023. In Berg, J., Järvisalo, M., Martins, R., and Niskanen, A., editors, *MaxSAT Evaluation 2023: Solver and Benchmark Descriptions*, volume B-2023-2 of *Department of Computer Science Series of Publications B*, pages 16–17, Helsinki, Finland. University of Helsinki.

Collautti, M. et al. (2013). SNNAP: Solver-based nearest neighbor for algorithm portfolios. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*, volume 8190 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg.

Du, D., Gu, J., and Pardalos, P. M. (1997). Approximate solution of weighted MAX-SAT problems using GRASP. In *Satisfiability Problem: Theory and Applications : DIMACS Workshop, March 11-13, 1996*, pages 299–306. American Mathematical Soc. Google-Books-ID: _GOVQRL50kcC.

Heinen, P. F., Iser, M., and Bach, J. (2022). *Cluster Analysis for SAT Instances*. Bachelor Thesis, Karlsruher Institut für Technologie (KIT).

Ihalainen, H. (2022). Refined core relaxations for core-guided maximum satisfiability algorithms. Master's thesis, MSc thesis, University of Helsinki, 2022, http://hdl. handle. net/10138/351207.

Ihalainen, H. E., Berg, J., and Järvisalo, M. (2021). Refined core relaxation for core-guided MaxSAT solving. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, pages 28–1. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Iser, M. and Jabs, C. (2024). Global Benchmark Database. In Chakraborty, S. and Jiang, J.-H. R., editors, *27th International Conference on Theory and Applications of Satisfiability Testing (SAT 2024)*, volume 305 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:10, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISSN: 1868-8969.

Jeremias Berg, Matti Järvisalo, Ruben Martins, Tobias Paxian, and Andreas Niskanen (2024). MaxSAT Evaluation 2024. `https://maxsat-evaluations.github.io/2024/`.

Kadioglu, S. et al. (2010). ISAC - instance-specific algorithm configuration. In *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI 2010)*. IOS Press.

Levy-Kramer, J. (2018). k-means-constrained.

Malitsky, Y., Sabharwal, A., Samulowitz, H., and Sellmann, M. (2013). Algorithm Portfolios Based on Cost-Sensitive Hierarchical Clustering.

Marques-Silva, J., Lynce, I., and Malik, S. (2021). Conflict-driven clause learning SAT solvers. In *Handbook of satisfiability*, pages 133–182. ios Press.

Matos, P., Planes, J., Letombe, F., and Marques-Silva, J. (2008). A MAX-SAT Algorithm Portfolio. pages 911–912.

Matti Järvisalo, Jeremias Berg, Ruben Martins, and Andreas Niskanen (2024). MaxSAT Evaluation 2023. `https://maxsat-evaluations.github.io/2023/`.

Pan, S., Wang, Y., Lei, Z., Cai, S., and Yin, M. (2023). CASHWMaxSAT-CorePlus: Solver Description. In Berg, J., Järvisalo, M., Martins, R., and Niskanen, A., editors, *MaxSAT Evaluation 2023: Solver and Benchmark Descriptions*, volume B-2023-2 of *Department of Computer Science Series of Publications B*, pages 8–8, Helsinki, Finland. University of Helsinki.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rice, J. R. (1976). The algorithm selection problem. In Rubinoff, M. and Yovits, M. C., editors, *Advances in Computers*, volume 15, pages 65–118. Elsevier.

scikit-learn (2024a). 1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking. `https://scikit-learn.org/1.6/modules/ensemble.html#forest`.

scikit-learn (2024b). 2.3. Clustering. `https://scikit-learn.org/1.6/modules/clustering.html`.

scikit-learn (2024c). 3.1. cross-validation: evaluating estimator performance. `https://scikit-learn.org/1.6/modules/cross_validation.html#cross-validation-evaluating-estimator-performance`.

scikit-learn (2024d). 3.4.4.13. matthews correlation coefficient. `https://scikit-learn.org/1.6/modules/model_evaluation.html#matthews-correlation-coefficient`.

scikit-learn (2024e). sklearn.preprocessing.StandardScaler. `https://scikit-learn.org/1.6/modules/generated/sklearn.preprocessing.StandardScaler.html`.

Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2008). SATzilla: Portfolio-based Algorithm Selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606.